

TTPA

1.1

Généré par Doxygen 1.7.6.1

Vendredi 8 Juin 2018 06 :19 :10

Table des matières

1	Page principale du projet TTPA (Table Tennis Performance Analyser)	1
1.1	Introduction	1
1.2	Table des matières	1
2	Changelog	1
3	Configuration	11
4	Manuel d'installation	12
5	Recette IR	12
6	Base de données	12
7	A propos	12
8	Licence GPL	13
9	Documentation des classes	13
9.1	Référence de la classe Clhm	13
9.1.1	Description détaillée	16
9.1.2	Documentation des constructeurs et destructeur	16
9.1.3	Documentation des fonctions membres	18
9.1.4	Documentation des données membres	31
9.2	Référence de la classe CommunicationBluetooth	32
9.2.1	Documentation des énumérations membres	33
9.2.2	Documentation des constructeurs et destructeur	34
9.2.3	Documentation des fonctions membres	35
9.2.4	Documentation des données membres	41
9.3	Référence de la classe CTable	41
9.3.1	Documentation des constructeurs et destructeur	43
9.3.2	Documentation des fonctions membres	44
9.3.3	Documentation des données membres	51
9.4	Référence de la classe CTrame	53
9.4.1	Documentation des constructeurs et destructeur	54
9.4.2	Documentation des fonctions membres	54
9.5	Référence de la classe com.ttpa.iris.ttpamobile.IHMEcranPrincipal	60
9.5.1	Description détaillée	63
9.5.2	Documentation des fonctions membres	63
9.5.3	Documentation des données membres	90

9.6	Référence de la classe <code>com.ttpa.iris.ttpamobile.IHMHistoriqueSeances</code>	98
9.6.1	Description détaillée	100
9.6.2	Documentation des fonctions membres	100
9.6.3	Documentation des données membres	106
9.7	Référence de la classe <code>com.ttpa.iris.ttpamobile.Joueur</code>	107
9.7.1	Description détaillée	108
9.7.2	Documentation des constructeurs et destructeur	108
9.7.3	Documentation des fonctions membres	108
9.7.4	Documentation des données membres	109
9.8	Référence de la classe <code>com.ttpa.iris.ttpamobile.ParametreSeance</code>	109
9.8.1	Description détaillée	110
9.8.2	Documentation des constructeurs et destructeur	110
9.8.3	Documentation des fonctions membres	111
9.8.4	Documentation des données membres	117
9.9	Référence de la classe <code>com.ttpa.iris.ttpamobile.PeripheriqueBluetooth</code>	119
9.9.1	Description détaillée	120
9.9.2	Documentation des constructeurs et destructeur	120
9.9.3	Documentation des fonctions membres	121
9.9.4	Documentation des données membres	123
9.10	Référence de la classe <code>com.ttpa.iris.ttpamobile.ReceveurBluetooth</code>	125
9.10.1	Documentation des constructeurs et destructeur	126
9.10.2	Documentation des fonctions membres	126
9.11	Référence de la classe <code>com.ttpa.iris.ttpamobile.Seance</code>	126
9.11.1	Description détaillée	127
9.11.2	Documentation des constructeurs et destructeur	127
9.11.3	Documentation des fonctions membres	128
9.11.4	Documentation des données membres	134
9.12	Référence de la classe <code>com.ttpa.iris.ttpamobile.ServeurBDD</code>	136
9.12.1	Description détaillée	137
9.12.2	Documentation des constructeurs et destructeur	137
9.12.3	Documentation des fonctions membres	137
9.12.4	Documentation des données membres	146
9.13	Référence de la classe <code>com.ttpa.iris.ttpamobile.ServeurSQLite</code>	146
9.13.1	Description détaillée	148
9.13.2	Documentation des constructeurs et destructeur	148
9.13.3	Documentation des fonctions membres	148
9.13.4	Documentation des données membres	150
9.14	Référence de la classe <code>com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception</code>	155

9.14.1	Documentation des constructeurs et destructeur	155
9.14.2	Documentation des fonctions membres	156
9.14.3	Documentation des données membres	157
10	Documentation des fichiers	157
10.1	Référence du fichier AndroidManifest.xml	157
10.2	Référence du fichier communicationbluetooth.cpp	157
10.2.1	Description détaillée	158
10.3	Référence du fichier communicationbluetooth.h	158
10.3.1	Description détaillée	158
10.3.2	Documentation des macros	158
10.4	Référence du fichier const.h	158
10.4.1	Documentation des macros	159
10.4.2	Documentation du type de l'énumération	162
10.5	Référence du fichier ihm.cpp	163
10.6	Référence du fichier ihm.h	163
10.6.1	Description détaillée	164
10.6.2	Documentation des macros	164
10.7	Référence du fichier IHMEcranPrincipal.java	164
10.8	Référence du fichier IHMHistoriqueSeances.java	164
10.9	Référence du fichier Joueur.java	164
10.10	Référence du fichier main.cpp	165
10.10.1	Documentation des fonctions	165
10.11	Référence du fichier ParametreSeance.java	165
10.12	Référence du fichier PeripheriqueBluetooth.java	165
10.13	Référence du fichier ReceveurBluetooth.java	165
10.14	Référence du fichier Seance.java	166
10.15	Référence du fichier ServeurBDD.java	166
10.16	Référence du fichier ServeurSQLite.java	166
10.17	Référence du fichier table.cpp	166
10.18	Référence du fichier table.h	166
10.18.1	Description détaillée	167
10.19	Référence du fichier trame.cpp	167
10.20	Référence du fichier trame.h	167
10.20.1	Description détaillée	168

1.1 Introduction

Le système doit permettre une analyse des performances du joueur (côté relanceur). Il doit proposer une phase d'entraînement adaptée au niveau du joueur, puis de détecter l'impact des balles afin d'afficher le rythme de jeu, la précision, le pourcentage de réussite. La zone d'impact (côté distributeur) est identifiée sur un écran de télévision en fin d'exercice. Le pourcentage de balles dans chacune des zones, le rythme de jeu et le pourcentage de réussite sont disponibles en fin d'exercice. Le joueur lance un exercice spécifique et pourra connaître son évolution individuelle.

Module Écran : Racamond Adrien Module Application Mobile : Smaniotto Nathan

1.2 Table des matières

- [Configuration](#)
- [Manuel d'installation](#)
- [Changelog](#)
- [Recette IR](#)
- [Base de données](#)
- [A propos](#)
- [Licence GPL](#)

Dépôt SVN : <https://svn.riouxsvn.com/ttpa>

2 Changelog

r117 | nsmaniotto | 2018-06-07 19 :06 :53 +0200 (jeu. 07 juin 2018) | 1 ligne

Tag de la version 1.1

r116 | nsmaniotto | 2018-06-07 19 :05 :17 +0200 (jeu. 07 juin 2018) | 1 ligne

Modification des fichiers Doxygen des modules Ecran et Terminal Mobile pour la version 1.1, ajout des documentations générées

r115 | aracamond | 2018-06-06 11 :03 :25 +0200 (mer. 06 juin 2018) | 1 ligne

Correction du correctif précédent lié a la taille du texte

r114 | nsmaniotto | 2018-06-05 15 :49 :54 +0200 (mar. 05 juin 2018) | 1 ligne

[TTPA Mobile] Implémentation de voyants permettant de visualiser l'état individuel de chacun des appareils bluetooth (table, lanceur, écran) du système

r113 | nsmaniotto | 2018-06-05 12 :03 :45 +0200 (mar. 05 juin 2018) | 1 ligne

[TTPA Mobile] Lorsqu'une séance est en cours (démarrée ou en pause), il n'est désormais plus possible d'interagir avec les sliders

r112 | nsmaniotto | 2018-06-05 10 :23 :04 +0200 (mar. 05 juin 2018) | 1 ligne

[TTPA Mobile] Modification de la sélection des zones

r111 | aracamond | 2018-06-05 10 :19 :36 +0200 (mar. 05 juin 2018) | 1 ligne

Correction du resize, la table causait des problemes dans le layer RECAP, correction de la font sur celle-ci

r110 | aracamond | 2018-06-04 15 :29 :03 +0200 (lun. 04 juin 2018) | 1 ligne

Ajout de l'affichage du nom du peripherique bluetooth connecte en dessous du nom du joueur, remplacement du nom du joueur sur l'affichage LOGO par le nom du peripherique

r109 | nsmaniotto | 2018-06-04 12 :39 :38 +0200 (lun. 04 juin 2018) | 1 ligne

[TTPA Mobile] Modifications graphiques pour la sélection des zones

r108 | nsmaniotto | 2018-06-04 10 :02 :52 +0200 (lun. 04 juin 2018) | 1 ligne

[TTPA Mobile] Modification du manifest Android afin que les activités soient forcées en mode portrait

r107 | nsmaniotto | 2018-06-04 09 :54 :38 +0200 (lun. 04 juin 2018) | 1 ligne

[TTPA Mobile] Implémentation d'un bouton d'arrêt de séance, modifications graphiques dans la sélection de la zone

r106 | nsmaniotto | 2018-06-02 15 :33 :40 +0200 (sam. 02 juin 2018) | 1 ligne

[TTPA Mobile] Calcul de l'intensité de l'effet à afficher : intensiteEffet allant de 1 à 9, on affiche un effet de 10 à 90%, sauf quand il n'y a pas d'effet

r105 | nsmaniotto | 2018-06-02 15 :30 :34 +0200 (sam. 02 juin 2018) | 1 ligne

[TTPA Mobile] Implémentation du paramètre de séance intensiteEffet

r104 | nsmaniotto | 2018-05-31 16 :40 :50 +0200 (jeu. 31 mai 2018) | 1 ligne

[TTPA Mobile] Implémentation de la méthode incrementerBallesJouees() de la classe IHMEcranPrincipal permettant de déterminer la fin d'une séance, même lorsque la dernière balle jouée est hors table

r103 | nsmaniotto | 2018-05-30 16 :19 :33 +0200 (mer. 30 mai 2018) | 1 ligne

[TTPA Mobile] Ajout de fichiers layout/xml permettant la portabilité sur mobile de la sélection des zones robot et objectif

r102 | nsmaniotto | 2018-05-30 15 :19 :23 +0200 (mer. 30 mai 2018) | 1 ligne

[TTPA Mobile] Ajout de fichiers layout/xml permettant la portabilité sur mobile de l'activité IHMHistorique-Seances

r101 | nsmaniotto | 2018-05-30 13 :55 :18 +0200 (mer. 30 mai 2018) | 1 ligne

[TTPA Mobile] Ajout de fichiers drawable/xml pour la dimension d'écran par défaut

r100 | nsmaniotto | 2018-05-30 13 :53 :37 +0200 (mer. 30 mai 2018) | 1 ligne

[TTPA Mobile] Rendre l'application utilisable sur mobile

r99 | nsmaniotto | 2018-05-28 15 :18 :08 +0200 (lun. 28 mai 2018) | 1 ligne

[TTPA Mobile] Correctif du crash généré à la sélection d'une zone

r98 | nsmaniotto | 2018-05-25 16 :48 :35 +0200 (ven. 25 mai 2018) | 1 ligne

Tag de la version 1.0

r97 | nsmaniotto | 2018-05-25 14 :12 :03 +0200 (ven. 25 mai 2018) | 1 ligne

[TTPA Mobile] Suppression de l'ancienne documentation générée à partir de Android Studio Javadoc

r96 | aracamond | 2018-05-25 14 :09 :24 +0200 (ven. 25 mai 2018) | 1 ligne

ajout de documentation pour Doxygen, déplacement du TARGET du .pro vers /bin

r95 | nsmaniotto | 2018-05-25 14 :09 :07 +0200 (ven. 25 mai 2018) | 1 ligne

[TTPA Mobile] Mise à jour de dossier sql/ contenant la documentation de la base de données pour le - Terminal Mobile

r94 | aracamond | 2018-05-25 12 :22 :35 +0200 (ven. 25 mai 2018) | 1 ligne

Suppression de fonctions redondantes, déplacement des variables string CSS en tant que Define, correction de bugs mineurs lié aux fonctions redondantes

r93 | nsmaniotto | 2018-05-25 11 :06 :55 +0200 (ven. 25 mai 2018) | 1 ligne

[TTPA Mobile] Ajout d'une capture de l'écran principal de l'application, à l'accueil de la documentation html Doxygen

r92 | nsmaniotto | 2018-05-25 10 :26 :33 +0200 (ven. 25 mai 2018) | 1 ligne

[TTPA Mobile] Ajout d'images de la doc dans le svn

r91 | nsmaniotto | 2018-05-25 10 :21 :42 +0200 (ven. 25 mai 2018) | 1 ligne

[TTPA Mobile] modifications apportées à la documentation

r90 | nsmaniotto | 2018-05-25 09 :56 :39 +0200 (ven. 25 mai 2018) | 1 ligne

[TTPA Mobile] Génération de la documentation Doxygen pour la version 1.0

r89 | nsmaniotto | 2018-05-24 17 :14 :44 +0200 (jeu. 24 mai 2018) | 1 ligne

[TTPA Mobile] Mise à jour du protocole des trames envoyées, la puissance (vitesse) des balles se fait désormais de 1 à 9

r88 | aracamond | 2018-05-24 16 :58 :24 +0200 (jeu. 24 mai 2018) | 1 ligne
correction du mode DEMO pour utiliser le nouveau systeme de HORS-TABLE

r87 | aracamond | 2018-05-24 16 :15 :06 +0200 (jeu. 24 mai 2018) | 1 ligne
Implementation du compteur d'enchainement maximum par seance

r86 | aracamond | 2018-05-24 16 :06 :23 +0200 (jeu. 24 mai 2018) | 1 ligne
Modification de l'aspect de la table pour etre plus visible

r85 | aracamond | 2018-05-24 15 :39 :39 +0200 (jeu. 24 mai 2018) | 1 ligne
Refonte du systeme 'Hors-Table', recoloration de la table pour une couleur plus vraie que nature

r84 | aracamond | 2018-05-23 17 :26 :46 +0200 (mer. 23 mai 2018) | 1 ligne
Correction de la zone robot ou dans certains cas ROBOT serait remplacé par le nombre de balles dans cette zone

r83 | aracamond | 2018-05-23 17 :15 :42 +0200 (mer. 23 mai 2018) | 1 ligne
Ajout de la table dans le tableau recapitulatif (deplacement du widget entier), correction pour void [Clhm- : :commencerSeance\(\)](#) les parametres ne sont plus reset

r82 | aracamond | 2018-05-22 16 :00 :10 +0200 (mar. 22 mai 2018) | 1 ligne
Découpage de bool [CTrame : :gererTrame\(QString\)](#) en sous parties

r81 | nsmaniotto | 2018-04-23 16 :59 :43 +0200 (lun. 23 avril 2018) | 1 ligne
[TTPA Mobile] Ajout de la méthode envoyerTrameArretPeripheriqueBluetoothTable() envoyant une trame de réinitialisation à la table

r80 | nsmaniotto | 2018-04-19 17 :23 :50 +0200 (jeu. 19 avril 2018) | 1 ligne
[TTPA Mobile] Ajout de la documentation Doxygen

r79 | nsmaniotto | 2018-04-19 17 :09 :38 +0200 (jeu. 19 avril 2018) | 1 ligne
[TTPA Mobile] Calcule du taux de réussite d'une séance opérationnel

r78 | nsmaniotto | 2018-04-19 14 :41 :24 +0200 (jeu. 19 avril 2018) | 1 ligne
[TTPA Mobile] Ajout du bouton permettant de visualiser des informations non affichées de base, dans l'activité d'historique de séances, grâce à une boîte de dialogue

r77 | nsmaniotto | 2018-04-19 13 :38 :56 +0200 (jeu. 19 avril 2018) | 1 ligne
[TTPA Mobile] Les zones rentrent désormais correctement dans leur boîte de dialogue

r76 | nsmaniotto | 2018-04-19 12 :14 :56 +0200 (jeu. 19 avril 2018) | 1 ligne

[TTPA Mobile] Lors du placmenet du robot ou de l'objectif, la zone déjà occupée est bloquée et son apparence ne change plus. On peut aussi désormais désélectionner une zone

r75 | aracamond | 2018-04-18 17 :38 :57 +0200 (mer. 18 avril 2018) | 1 ligne

changement de taille du logo PAUSE

r74 | aracamond | 2018-04-18 15 :47 :56 +0200 (mer. 18 avril 2018) | 1 ligne

update de [const.h](#)

r73 | aracamond | 2018-04-18 15 :34 :55 +0200 (mer. 18 avril 2018) | 1 ligne

ajout de la possibilité de faire une pause et de reprendre, correction d'un bug lié a la lecture des trames à 1 argument

r72 | nsmaniotto | 2018-04-18 15 :22 :06 +0200 (mer. 18 avril 2018) | 1 ligne

[TTPA Mobile] Implémeentation de la pause d'une séance ainsi que de sa reprise, ajout d'images pour le bouton d'action de séance pour chaque état, démarrer, mettre en pause, reprendre une séance

r71 | nsmaniotto | 2018-04-18 10 :38 :56 +0200 (mer. 18 avril 2018) | 1 ligne

[TTPA Mobile] Suppression d'anciens layout, suppression de l'icône de base qui n'était jamais utilisée, implémentation de la gestion de la couleur de fond du bouton de paramètres de zones

r70 | nsmaniotto | 2018-04-18 09 :59 :49 +0200 (mer. 18 avril 2018) | 1 ligne

[TTPA Mobile] Modification des classes IHMHistoriqueSeances et ServeurBDD , la purge des séances d'un joueur est désormais opérationnelle

r69 | nsmaniotto | 2018-04-18 09 :24 :49 +0200 (mer. 18 avril 2018) | 1 ligne

[TTPA Mobile] Ajout de l'icône pour démarrer une séance, finition de la méthode selectionnerZone() qui est désormais opérationnelle

r68 | tvaira | 2018-04-16 14 :51 :35 +0200 (lun. 16 avril 2018) | 1 ligne

Verification des TODO

r67 | tvaira | 2018-04-14 16 :37 :27 +0200 (sam. 14 avril 2018) | 1 ligne

Exemple boite de dialogue pour la selection

r66 | nsmaniotto | 2018-04-12 22 :25 :38 +0200 (jeu. 12 avril 2018) | 1 ligne

[TTPA Mobile] Implémentation de la classe IHMHistoriqueSeances et du layout ecran_historique_seances permettant d'afficher l'historique des séances du joueur sélectionné

r65 | aracamond | 2018-04-12 22 :22 :55 +0200 (jeu. 12 avril 2018) | 1 ligne

ajout de commentaires pour doxygen

r64 | aracamond | 2018-04-12 21 :39 :51 +0200 (jeu. 12 avril 2018) | 1 ligne

améliorations au mode -demo, ajout de -norobot pour desactiver le robot dans ce mode

r63 | aracamond | 2018-04-12 21 :02 :50 +0200 (jeu. 12 avril 2018) | 1 ligne

Remplacement de int en uint8_t pour les zones du fait qu'il est inutile d'avoir plus

r62 | aracamond | 2018-04-12 19 :15 :16 +0200 (jeu. 12 avril 2018) | 1 ligne

ajout de la fenetre de recapitulatif de la séance ainsi que de la connexion du slot a celle ci

r61 | nsmaniotto | 2018-04-12 17 :34 :18 +0200 (jeu. 12 avril 2018) | 1 ligne

[TTPA Mobile] Ajout de méthodes pour le déroulement d'une séance, implémentation de la rotation du lanceur

r60 | tvaira | 2018-04-11 18 :50 :35 +0200 (mer. 11 avril 2018) | 1 ligne

Modification SQL

r59 | tvaira | 2018-04-11 17 :18 :29 +0200 (mer. 11 avril 2018) | 1 ligne

Modification README (SQL)

r58 | nsmaniotto | 2018-04-11 13 :26 :22 +0200 (mer. 11 avril 2018) | 1 ligne

[TTPA Mobile] Implémentation des méthodes permettant le traitement des trames reçues par les périphériques Bluetooth Lanceur et Table

r57 | nsmaniotto | 2018-04-11 11 :40 :17 +0200 (mer. 11 avril 2018) | 1 ligne

[TTPA Mobile] Le système est fonctionnel avec un seul des trois appareils

r56 | tvaira | 2018-04-10 21 :17 :48 +0200 (mar. 10 avril 2018) | 1 ligne

Retour sur la revue 2

r55 | tvaira | 2018-04-10 11 :43 :07 +0200 (mar. 10 avril 2018) | 1 ligne

Ajout simulateurs Lanceur et Table (ESP32)

r54 | aracamond | 2018-04-09 16 :00 :09 +0200 (lun. 09 avril 2018) | 1 ligne

Ajout de la gestion du découpage de trame en cas de trames concaténés

r53 | nsmaniotto | 2018-04-09 16 :00 :01 +0200 (lun. 09 avril 2018) | 1 ligne

[TTPA Mobile] Traitement des valeurs des barres de progression, aspect des barres de progression modifié, évènement de la modification d'une valeur d'une barre de progression implémenté

r52 | aracamond | 2018-04-05 17 :56 :55 +0200 (jeu. 05 avril 2018) | 1 ligne

changement de trames, correction de bugs relatif au trames, restructuration de l'interface TABLE, ajout du timer de seance

r51 | nsmaniotto | 2018-04-05 17 :44 :58 +0200 (jeu. 05 avril 2018) | 1 ligne

[TTPA Mobile] Refonte de l'IHM

r50 | nsmaniotto | 2018-04-04 17 :53 :58 +0200 (mer. 04 avril 2018) | 1 ligne

[TTPA Mobile] Implémentation de la communication Bluetooth, ajout des classes PeripheriqueBluetooth et ReceveurBluetooth

r49 | aracamond | 2018-04-04 16 :25 :36 +0200 (mer. 04 avril 2018) | 1 ligne

Adaptation au nouveau protocole d'impact, correction de bugs lié au nombre de balles

r48 | tvaira | 2018-03-30 09 :38 :09 +0200 (ven. 30 mars 2018) | 1 ligne

Ajout de la gestion des signaux dans le thread de communication Bluetooth

r47 | nsmaniotto | 2018-03-29 09 :26 :06 +0200 (jeu. 29 mars 2018) | 1 ligne

Tag version 0.8 pour la revue 2

r46 | nsmaniotto | 2018-03-29 09 :20 :03 +0200 (jeu. 29 mars 2018) | 1 ligne

[TTPA Mobile] Méthode de validation de la fréquence fonctionnelle

r45 | nsmaniotto | 2018-03-29 08 :59 :49 +0200 (jeu. 29 mars 2018) | 1 ligne

[TTPA Mobile] Implémentation de la vérification des paramètres saisis

r44 | aracamond | 2018-03-28 16 :24 :00 +0200 (mer. 28 mars 2018) | 1 ligne

Correction fermeture/ouverture ports, slot deconnexion

r43 | nsmaniotto | 2018-03-28 09 :23 :38 +0200 (mer. 28 mars 2018) | 1 ligne

[TTPA Mobile] Ajout de la documentation JavaDoc du projet sous Android Studio

r42 | nsmaniotto | 2018-03-28 08 :37 :54 +0200 (mer. 28 mars 2018) | 1 ligne

[TTPA Mobile] Début de l'implémentation de la sauvegarde automatique des paramètres de séance actuelle lorsqu'ils sont modifiés (à l'enregistrement pour l'instant)

r41 | tvaira | 2018-03-25 14 :23 :38 +0200 (dim. 25 mars 2018) | 1 ligne

Ajout du Thread pour la communication Bluetooth

r40 | tvaira | 2018-03-24 11 :02 :52 +0100 (sam. 24 mars 2018) | 1 ligne

Ajout parametrage Doxygen (Mobile)

r39 | tvaira | 2018-03-24 10 :51 :17 +0100 (sam. 24 mars 2018) | 1 ligne

Ajout parametrage Doxygen

r38 | aracamond | 2018-03-22 15 :58 :13 +0100 (jeu. 22 mars 2018) | 1 ligne

Refonte de l'interface en termes de couleurs, correction de problemes liées à l'affichage, finalisation des trames

r37 | aracamond | 2018-03-21 17 :59 :36 +0100 (mer. 21 mars 2018) | 1 ligne

Finission des trames, TODO : Fixer le X de connection, thread pour la verification de la connexion

r36 | nsmaniotto | 2018-03-21 17 :50 :44 +0100 (mer. 21 mars 2018) | 1 ligne

[TTPA Mobile] Implémentation des paramètres actuels de séance, enregistrés dans une table de la base de données

r35 | nsmaniotto | 2018-03-21 14 :03 :13 +0100 (mer. 21 mars 2018) | 1 ligne

[TTPA Mobile] Fragmentation du code source de la classe MainActivity sous formes de méthodes

r34 | aracamond | 2018-03-19 18 :06 :23 +0100 (lun. 19 mars 2018) | 1 ligne

Correction du problem lié a la deconnection du port

r33 | nsmaniotto | 2018-03-19 16 :13 :48 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Ajout d'un raccourci dans le menu pour naviguer vers l'activité des paramètres de la séance

r32 | aracamond | 2018-03-19 15 :18 :31 +0100 (lun. 19 mars 2018) | 1 ligne

MAJ Documentation

r31 | nsmaniotto | 2018-03-19 15 :17 :05 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Boutons permettant d'appliquer individuellement des paramétrages enregistrés désormais opérationnels

r30 | nsmaniotto | 2018-03-19 14 :51 :45 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Implémentation d'un bouton dans l'activité ParametresSeance permettant d'enregistrer dans

la base de données les paramètres actuels

r29 | nsmaniotto | 2018-03-19 14 :13 :11 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Boutons de suppression individuelle de séance/paramètres opérationnels

r28 | nsmaniotto | 2018-03-19 13 :57 :28 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Purge des paramètres de séance dans la base de données désormais opérationnelle

r27 | nsmaniotto | 2018-03-19 13 :54 :09 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Purge des séances dans la base de données désormais opérationnelle

r26 | nsmaniotto | 2018-03-19 12 :23 :52 +0100 (lun. 19 mars 2018) | 1 ligne

Méthode afficherParametre() opérationnelle, les paramètres enregistrés dans la base de données sont désormais affichées au démarrage de l'activité

r25 | aracamond | 2018-03-19 11 :36 :20 +0100 (lun. 19 mars 2018) | 1 ligne

Deplacement des fonctions de gestion de trames dans [CTrame](#)

r24 | nsmaniotto | 2018-03-19 09 :01 :37 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Ajout de la classe ParametreSeance définissant les caractéristiques d'un paramétrage

r23 | nsmaniotto | 2018-03-19 08 :44 :02 +0100 (lun. 19 mars 2018) | 1 ligne

[TTPA Mobile] Méthode afficherSeance() opérationnelle, les séances présentes dans la base de données sont désormais affichées au démarrage de l'activité

r22 | nsmaniotto | 2018-03-17 11 :41 :45 +0100 (sam. 17 mars 2018) | 1 ligne

[TTPA Mobile] Implémentation des classes ServeurBDD et ServeurSQLite permettant la création et l'utilisation de bases de données

r21 | nsmaniotto | 2018-03-16 12 :26 :29 +0100 (ven. 16 mars 2018) | 1 ligne

[TTPA Mobile] Ajout de la classe Seance définissant les caractéristiques d'une séance

r20 | nsmaniotto | 2018-03-16 11 :18 :25 +0100 (ven. 16 mars 2018) | 1 ligne

[TTPA Mobile] Implémentation des boutons permettant l'édition et la suppression de paramètres enregistrés un par un

r19 | aracamond | 2018-03-15 16 :57 :29 +0100 (jeu. 15 mars 2018) | 1 ligne

Commencement de la documentation Doxygen, apparition d'un bug causé par l'ouverture/fermeture du bluetooth

r18 | nsmaniotto | 2018-03-15 16 :54 :54 +0100 (jeu. 15 mars 2018) | 1 ligne

[TTPA Mobile] Implémentation des boutons permettant de supprimer les séances une par une

r17 | nsmaniotto | 2018-03-15 12 :04 :44 +0100 (jeu. 15 mars 2018) | 1 ligne

[TTPA Mobile] Implémentation des boutons permettant de pruger les séances et les paramètres enregistrés

r16 | nsmaniotto | 2018-03-15 11 :54 :26 +0100 (jeu. 15 mars 2018) | 1 ligne

Fragmentation du layout des paramètres de séance en deux parties : main et content

r15 | nsmaniotto | 2018-03-12 19 :22 :28 +0100 (lun. 12 mars 2018) | 1 ligne

Début de l'IHM de l'application mobile

r14 | aracamond | 2018-02-22 18 :05 :10 +0100 (jeu. 22 févr. 2018) | 1 ligne

ajout du QextSerialPort

r13 | aracamond | 2018-02-22 17 :54 :43 +0100 (jeu. 22 févr. 2018) | 1 ligne

Ajout de la zone objectif, refonte de l'interface de la table pour la zone objectif + couleurs, ajout de la capture du port serie (fonctionnel mais non terminé).

r12 | aracamond | 2018-02-15 17 :37 :29 +0100 (jeu. 15 févr. 2018) | 1 ligne

Changement des statistiques, reussite globale a faire

r11 | aracamond | 2018-02-15 15 :51 :05 +0100 (jeu. 15 févr. 2018) | 1 ligne

ajout du batch pour lancement depuis un SSH

r10 | aracamond | 2018-02-15 15 :23 :48 +0100 (jeu. 15 févr. 2018) | 1 ligne

correction des images

r9 | aracamond | 2018-02-15 15 :10 :18 +0100 (jeu. 15 févr. 2018) | 1 ligne

restructuration de l'affichage, debut de l'IHM a gauche, changement du logo

r8 | aracamond | 2018-02-14 17 :57 :18 +0100 (mer. 14 févr. 2018) | 1 ligne

ajout des pourcentages, refonte de la table, ajout du mode -dev

r7 | aracamond | 2018-02-14 10 :50 :06 +0100 (mer. 14 févr. 2018) | 1 ligne

ajout du fichier filet.jpg, fonction rafraichirCSS pour IHM

r6 | tvaira | 2018-02-08 22 :45 :03 +0100 (jeu. 08 févr. 2018) | 1 ligne

Ajout d'un QStackedWidget comme centralWidget

r5 | aracamond | 2018-02-08 17 :54 :55 +0100 (jeu. 08 févr. 2018) | 1 ligne

Ajout de fonctionalitees a Ecran-TTPA

r4 | aracamond | 2018-02-07 18 :37 :07 +0100 (mer. 07 févr. 2018) | 1 ligne

ajout de la base du programme Ecran-TTPA

r3 | tvaira | 2018-02-03 11 :21 :32 +0100 (sam. 03 févr. 2018) | 1 ligne

Ajout script pour Android (tv)

r2 | tvaira | 2018-02-03 11 :12 :59 +0100 (sam. 03 févr. 2018) | 1 ligne

Ajout initial (tv)

r1 | www-data | 2018-02-03 11 :03 :09 +0100 (sam. 03 févr. 2018) | 1 ligne

Creating initial repository structure

3 Configuration

Poste de développement :

- Distribution : Ubuntu 12.04.5 LTS
- OS : GNU/Linux
- Noyau : Linux
- Version : 3.8.0-44-generic
- Machine : x86_64
- Processeur : Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
- Mémoire RAM : 8129984 kB

Informations de version sur les outils

- Android Studio 2.3
- java 1.8.0
- SDK Android API 25 : Android 7.1.1 (Nougat)
- svn, version 1.6.17 (r1128011)
- doxygen 1.7.6.1
- bouml Bouml 7.4

Liste des paquets Qt nécessaires :

- libqtgui4 libqtcore4 libqt4-svg

4 Manuel d'installation

Fabrication de l'exécutable :

- qmake
- make

5 Recette IR

Étudiant 3 : Racamond Adrien

- Le système d'exploitation est installé et fonctionnel
- L'écran est configuré en mode "kiosque"
- La zone d'impact est identifiée et affichée en temps réel
- Les données de la séance sont affichées en temps réel
- Les liaisons sans fil sont opérationnelles
- Les informations sont affichées en fin de séquence

Étudiant 4 : Smaniotto Nathan

- La base de données est fonctionnelle et complétée
- Le système est paramétrable
- La liaison Bluetooth est fonctionnelle
- Les informations de paramétrages sont transmises
- L'application mobile est déployée

6 Base de données

```
pragma foreign_keys = on ;
```

```
CREATE TABLE table_joueurs ( "ID_JOUEUR" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
"NOM" VARCHAR(255) NOT NULL ) ;
```

```
INSERT INTO table_joueurs(NOM) VALUES('LEGOUT Christophe') ; INSERT INTO table_joueurs(NOM)  
VALUES('MARTINEZ Michel') ; INSERT INTO table_joueurs(NOM) VALUES('LEBESSON Emmanuel') ; IN-  
SERT INTO table_joueurs(NOM) VALUES('ELOI Damien') ; INSERT INTO table_joueurs(NOM) VALUES('-  
MATTENET Adrien') ; INSERT INTO table_joueurs(NOM) VALUES('CHILA Patrick') ; INSERT INTO table_  
joueurs(NOM) VALUES('BEAUMONT Jérôme') ;
```

```
CREATE TABLE table_seances ( "ID_SEANCE" INTEGER PRIMARY KEY AUTOINCREMENT, "NOMB-  
RE_BALLES" INTEGER NOT NULL, "FREQUENCE" INTEGER NOT NULL, "EFFET" VARCHAR(1) NOT  
NULL, "PUISSANCE" INTEGER NOT NULL, "ROTATION" INTEGER NOT NULL, "ZONE_OBJECTIF" IN-  
TEGER NOT NULL, "ZONE_ROBOT" INTEGER NOT NULL, "TAUX_REUSSITE" REAL NOT NULL, "DA-  
TE_DEBUT" DATETIME NOT NULL, "DATE_FIN" DATETIME NOT NULL, "ID_JOUEUR" INTEGER NOT  
NULL, CONSTRAINT fk_seances_1 FOREIGN KEY (ID_JOUEUR) REFERENCES table_joueurs (ID_JO-  
UEUR) ON DELETE CASCADE ) ;
```

```
CREATE TABLE table_parametres ( "ID_PARAMETRE" INTEGER PRIMARY KEY CHECK (ID_PARAME-  
TRE = 1), "ID_JOUEUR" INTEGER NOT NULL, CONSTRAINT fk_parametres_1 FOREIGN KEY (ID_JOU-  
EUR) REFERENCES table_joueurs (ID_JOUEUR) ) ;
```

```
INSERT INTO table_parametres(ID_PARAMETRE, ID_JOUEUR) VALUES(1, 1) ;
```

7 A propos

Auteur

Racamond Adrien <adrien.racamond.lasalle@gmail.com>

Smaniotto Nathan <smaniotto.nathan@gmail.com>

Version

1.1

Date

2018

8 Licence GPL

This program is free software ; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation ; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY ; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program ; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

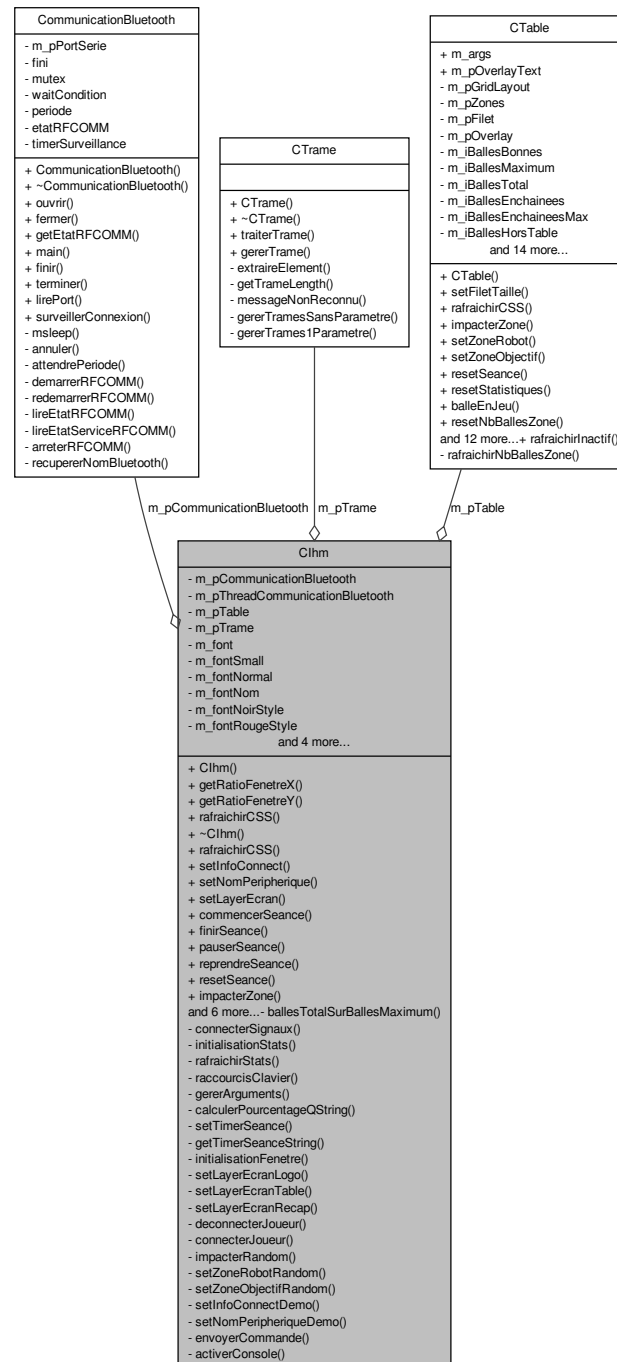
9 Documentation des classes

9.1 Référence de la classe Clhm

Classe principale de l'application (IHM)

```
#include <ihm.h>
```

Graphe de collaboration de Clhm :



Connecteurs publics

- void **rafraichirCSS** ()
Rafraichit le CSS lié à l'affichage (fontes, couleurs) utilisant **getRatioFenetreY()**
- void **setInfoConnect** (QString nom)
Affiche le nom du joueur sur la table.

- void `setNomPeripherique` (QString nom)
Affiche le nom du périphérique connecté
- void `setLayerEcran` (uint8_t layer)
Change d'écran.
- void `commencerSeance` ()
Commencer la seance.
- void `finirSeance` ()
Finir la seance et afficher la fenêtre Recap.
- void `pauserSeance` ()
- void `reprendreSeance` ()
Reprendre la seance suite a une pause.
- void `resetSeance` ()
Reset des statistiques et de la configuration.
- void `impacterZone` (uint8_t zone)
Calcul et affiche l'impact sur l'IHM et la table.
- void `balleEnJeu` ()
La balle a été capté sur le capteur coté joueur, ajout de la balle.
- void `setZoneRobot` (uint8_t zone)
Place le robot sur la table.
- void `setZoneObjectif` (uint8_t zone)
Place la zone objectif sur la table.
- void `setBallesMaximum` (int balles)
Définit le nombre de balles maximum pour la seance.
- void `rafraichirHeure` ()
Rafraichit l'heure sur l'IHM (Logo et Table)
- void `rafraichirTimerSeance` ()
Rafraichit le timer de la seance.
- void `quitter` ()
Quitte l'application (utilisé par le raccourci CTRL+Q)

Fonctions membres publiques

- `Clhm` (QWidget *parent=0)
- float `getRatioFenetreX` ()
Récupere le ratio du width par rapport à la résolution par default 960x540.
- float `getRatioFenetreY` ()
Récupere le ratio du height par rapport à la résolution par default 960x540.
- void `rafraichirCSS` (float ratio)
- `~Clhm` ()

Connecteurs privés

- void `initialisationFenetre` ()
Initialise la taille des fenetres en fonction de la résolution de l'écran en mode plein-ecran.
- void `setLayerEcranLogo` ()
Bascule sur le menu d'attente.
- void `setLayerEcranTable` ()
Bascule sur l'interface de la table.
- void `setLayerEcranRecap` ()
Bascule sur la récapitulation de la séance.
- void `deconnecterJoueur` ()
Actions nécessaires a la deconnexion du joueur.
- void `connecterJoueur` ()
Actions nécessaires a la connexion du joueur.
- void `impacterRandom` ()
[DEBUG] Envoi une balle aléatoire sur la table coté robot
- void `setZoneRobotRandom` ()
[DEBUG] Définir une position du robot aléatoire (executé depuis un bouton sur l'IHM)
- void `setZoneObjectifRandom` ()
[DEBUG] Définir un objectif aléatoire (executé depuis un bouton sur l'IHM)
- void `setInfoConnectDemo` ()
[DEBUG] Execute `setInfoConnect()` avec le nom de demonstration
- void `setNomPeripheriqueDemo` ()
[DEBUG] Execute `setNomPeripherique()` avec le nom de demonstration
- void `envoyerCommande` ()

- void [activerConsole](#) ()
[DEBUG] Envoi une commande de la console, accepte aussi les trames
- void [resetConsole](#) ()
[DEBUG] Reset le CSS de la console (nécessaire a cause du delai)

Fonctions membres privées

- QString [ballesTotalSurBallesMaximum](#) ()
String affichant le l'état des balles renvoyées sur le nombre de balles paramétrée pour la seance.
- void [connecterSignaux](#) ()
Réalise la connexion des slots/signaux.
- void [initialisationStats](#) ()
Initialisation des statistiques de la seance.
- void [rafraichirStats](#) ()
Rafraichissement des statistiques de la seance exécuté a chaque impact/changement de parametre.
- void [raccourcisClavier](#) ()
Implémentation des raccourcis clavier.
- void [gererArguments](#) ()
Verification des options de lancement.
- QString [calculerPourcentageQString](#) (int x, int y)
Récupere sous forme de QString un pourcentage : "(X%)" utilisé pour les statistiques.
- void [setTimerSeance](#) (unsigned int iTemps=0)
Ecriture du temps dans m_pQLabelTimerSeance.
- QString [getTimerSeanceString](#) (unsigned int iTemps)

Attributs privés

- [CommunicationBluetooth](#) * [m_pCommunicationBluetooth](#)
Gestion de la communication Bluetooth.
- QThread * [m_pThreadCommunicationBluetooth](#)
Thread pour la classe [CommunicationBluetooth](#).
- [CTable](#) * [m_pTable](#)
Association vers la classe [CTable](#).
- [CTrame](#) * [m_pTrame](#)
- QFont [m_font](#)
- QFont [m_fontSmall](#)
- QFont [m_fontNormal](#)
- QFont [m_fontNom](#)
- QString [m_fontNoirStyle](#)
- QString [m_fontRougeStyle](#)
- QString [m_fontVertStyle](#)
- QString [m_fontTitreStyle](#)
- unsigned int [m_iTempsSeance](#)
Temps en seconde de la seance.
- QTimer * [m_pTimerHeure](#)
Timer rafraichissant l'Heure.
- QTimer * [m_pTimerSeance](#)
Timer incrémentant le temps de la seance.

9.1.1 Description détaillée

Auteur

Racamond Adrien

Version

0.9

9.1.2 Documentation des constructeurs et destructeur

9.1.2.1 Clhm : :Clhm (QWidget * parent = 0) [explicit]

Références [connecterSignaux\(\)](#), [DELAI_FIXFENETRE](#), [CommunicationBluetooth : :Evenement](#), [gererArguments\(\)](#), [initialisationFenetre\(\)](#), [LAYER_LOGO](#), [m_fontNoirStyle](#), [m_fontRougeStyle](#), [m_fontTitreStyle](#), [m_fontVertStyle](#), [m_iTempsSeance](#), [m_pCommunicationBluetooth](#), [m_pTable](#), [m_pThreadCommunicationBluetooth](#), [m_pTimerHeure](#), [m_pTimerSeance](#), [m_pTrame](#), [PORT_BLUETOOTH](#), [raccourcisClavier\(\)](#), et [setTimerSeance\(\)](#).

```

        : QWidget (parent)
{
    setupUi (this);
    raccourcisClavier ();

#ifdef QT_NO_DEBUG
    qDebug () << Q_FUNC_INFO << "PID : " << (int)qApp->applicationPid () << "TID
        : " << QApplication::instance ()->thread ()->currentThreadId () << qApp->thread ();
#endif

    // Gestion de la communication Bluetooth
    m_pCommunicationBluetooth = new CommunicationBluetooth (PORT_BLUETOOTH,
        CommunicationBluetooth::Evenement);
    // ou :
    //communicationBluetooth = new CommunicationBluetooth (PORT_BLUETOOTH,
        CommunicationBluetooth::Scrutation);
    m_pThreadCommunicationBluetooth = new QThread;
    m_pCommunicationBluetooth->moveToThread (m_pThreadCommunicationBluetooth);

    m_pTrame = new CTrame (this);

    m_fontNoirStyle = QString::fromUtf8 ("QLabel\\n{\\n        color: #000000;\\n}");
    m_fontRougeStyle = QString::fromUtf8 ("QLabel\\n{\\n        color: #FF0000;\\n}");
    m_fontVertStyle = QString::fromUtf8 ("QLabel\\n{\\n        color: #006500;\\n}");
    m_fontTitreStyle = QString::fromUtf8 ("QLabel\\n{\\n        color: #000000;\\n
        background: qlineargradient (spread:reflect, x1:0.5, y1:0, x2:1, y2:0, stop:0
        rgba(255, 255, 255, 255), stop:1 rgba(0, 0, 0, 0)); \\n}");

    m_pTable = new CTable (this);
    QTimer::singleShot (DELAI_FIXFENETRE, this, SLOT (initialisationFenetre ()));
    QTimer::singleShot (DELAI_FIXFENETRE*4, this, SLOT (initialisationFenetre ()))
    ;
    m_pHLayoutTable->addWidget (m_pTable);

    m_pTimerHeure = new QTimer (this);
    m_pTimerHeure->setInterval (3000);

    m_pTimerSeance = new QTimer (this);
    m_pTimerSeance->setInterval (1000);
    m_iTempsSeance = 0;
    setTimerSeance (0);

    gererArguments ();

    m_pFenetres->setCurrentIndex (LAYER_LOGO);

    connecterSignaux ();

    m_pTimerHeure->start ();

    // démarre la communication Bluetooth
    m_pThreadCommunicationBluetooth->start ();
}

```

9.1.2.2 Clhm : :~Clhm ()

Références [CommunicationBluetooth : :finir\(\)](#), [m_pCommunicationBluetooth](#), et [m_pThreadCommunicationBluetooth](#).

```

{
    // Ferme la communication Bluetooth
    m_pCommunicationBluetooth->finir ();
    m_pThreadCommunicationBluetooth->quit ();
    m_pThreadCommunicationBluetooth->wait ();
    delete m_pCommunicationBluetooth;
    delete m_pThreadCommunicationBluetooth;
}

```

```

    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << "fin";
    #endif
}

```

9.1.3 Documentation des fonctions membres

9.1.3.1 void Clhm : :activerConsole () [private, slot]

Référencé par [envoyerCommande\(\)](#).

```

{
    m_pConsole->setStyleSheet("QLineEdit#m_pConsole\\n{\\n\\n}");
}

```

9.1.3.2 Clhm : :balleEnJeu () [slot]

Références [CTable : :balleEnJeu\(\)](#), [ballesTotalSurBallesMaximum\(\)](#), [m_fontNormal](#), [m_pTable](#), et [rafraichirStats\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    m_pTable->balleEnJeu();
    m_pQLabelTopMid->setFont(m_fontNormal);
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());
    rafraichirStats();
}

```

9.1.3.3 QString Clhm : :ballesTotalSurBallesMaximum () [private]

Références [CTable : :getBallesMaximum\(\)](#), [CTable : :getBallesTotal\(\)](#), et [m_pTable](#).

Référencé par [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [impacterZone\(\)](#), [rafraichirCSS\(\)](#), et [setBallesMaximum\(\)](#).

```

{
    QString ballesMax = QString::number(m_pTable->getBallesMaximum());

    if (!m_pTable->getBallesMaximum())
        ballesMax = QString::fromUtf8("");

    return /*QString::fromUtf8(IHM_BALLESENVOYEES) */ QString::number(m_pTable
->getBallesTotal()) + " / " + ballesMax;
}

```

9.1.3.4 QString Clhm : :calculerPourcentageQString (int x, int y) [private]

Référencé par [finirSeance\(\)](#), et [rafraichirStats\(\)](#).

```

{
    if (!y)
        return "(0%)";

    return "(" + QString::number((double(x) / double(y))*100, 'f', 0) + "%)";
}

```

9.1.3.5 Clhm : :commencerSeance () [slot]

Références [ballesTotalSurBallesMaximum\(\)](#), [CSS_TIMER_ON](#), [LAYER_TABLE](#), [m_iTempsSeance](#), [m_pTable](#), [m_pTimerSeance](#), [rafraichirStats\(\)](#), [CTable : :resetStatistiques\(\)](#), [setLayerEcran\(\)](#), et [setTimerSeance\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;

    //    if (m_pFenetres->currentIndex() != LAYER_TABLE)
    //        return;

    setLayerEcran(LAYER_TABLE);

    m_iTempsSeance = 0;
    m_pTimerSeance->start(1000);

    m_pQLabelTimerSeance->setStyleSheet(CSS_TIMER_ON);

    m_pHLayoutTable->addWidget(m_pTable);
    m_pTable->resetStatistiques();
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());

    setTimerSeance(0);
    rafraichirStats();
}

```

9.1.3.6 void Clhm::connecterJoueur() [private, slot]

Références [LAYER_LOGO](#), [LOGO_ATTENTECONFIGURATION](#), et [setLayerEcran\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    setLayerEcran(LAYER_LOGO);
    //    m_pQLabelLogoTexte->setText(LOGO_ATTENTEIDENTIFICATION); Ancien Message
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONFIGURATION);
}

```

9.1.3.7 void Clhm::connecterSignaux() [private]

Références [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [connecterJoueur\(\)](#), [deconnecterJoueur\(\)](#), [finirSeance\(\)](#), [impacterZone\(\)](#), [m_pCommunicationBluetooth](#), [m_pThreadCommunicationBluetooth](#), [m_pTimerHeure](#), [m_pTimerSeance](#), [m_pTrame](#), [main\(\)](#), [pauserSeance\(\)](#), [quitter\(\)](#), [rafraichirCSS\(\)](#), [rafraichirHeure\(\)](#), [rafraichirTimerSeance\(\)](#), [reprendreSeance\(\)](#), [resetSeance\(\)](#), [setBallesMaximum\(\)](#), [setInfoConnect\(\)](#), [setLayerEcran\(\)](#), [setNomPeripherique\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

Référencé par [Clhm\(\)](#).

```

{
    // Thread Bluetooth
    connect(m_pThreadCommunicationBluetooth, SIGNAL(started()),
           m_pCommunicationBluetooth, SLOT(main()));
    connect(m_pThreadCommunicationBluetooth, SIGNAL(finished()),
           m_pCommunicationBluetooth, SLOT(terminer()));

    // Timers
    connect(m_pTimerHeure, SIGNAL(timeout()), this, SLOT(rafraichirHeure()));
    connect(m_pTimerSeance, SIGNAL(timeout()), this, SLOT(rafraichirTimerSeance(
    )));

    // Gestion des trames
    connect(m_pCommunicationBluetooth, SIGNAL(nouvellesDonneesRecues(QString)),
           m_pTrame, SLOT(traiterTrame(QString)));
    connect(m_pTrame, SIGNAL(setLayerEcran(uint8_t)), this, SLOT(setLayerEcran(
    uint8_t)));
    connect(m_pTrame, SIGNAL(setInfoConnect(QString)), this, SLOT(setInfoConnect(
    QString)));
    connect(m_pTrame, SIGNAL(resetSeance()), this, SLOT(resetSeance()));
    connect(m_pTrame, SIGNAL(commencerSeance()), this, SLOT(commencerSeance()));
    ;
    connect(m_pTrame, SIGNAL(pauserSeance()), this, SLOT(pauserSeance()));
    connect(m_pTrame, SIGNAL(reprendreSeance()), this, SLOT(reprendreSeance()));
    ;
    connect(m_pTrame, SIGNAL(finirSeance()), this, SLOT(finirSeance()));
    connect(m_pTrame, SIGNAL(impacterZone(uint8_t)), this, SLOT(impacterZone(
    uint8_t)));
    connect(m_pTrame, SIGNAL(balleEnJeu()), this, SLOT(balleEnJeu()));
    connect(m_pTrame, SIGNAL(setZoneRobot(uint8_t)), this, SLOT(setZoneRobot(

```

```

    uint8_t));
connect(m_pTrame, SIGNAL(setZoneObjectif(uint8_t)), this, SLOT(
    setZoneObjectif(uint8_t)));
connect(m_pTrame, SIGNAL(setBallesMaximum(int)), this, SLOT(setBallesMaximum
    (int)));
connect(m_pTrame, SIGNAL(rafraichirCSS()), this, SLOT(rafraichirCSS()));
connect(m_pTrame, SIGNAL(quitte()), this, SLOT(quitte()));

connect(m_pCommunicationBluetooth, SIGNAL(deconnecterJoueur()), this, SLOT(
    deconnecterJoueur()));
connect(m_pCommunicationBluetooth, SIGNAL(connecterJoueur()), this, SLOT(
    connecterJoueur()));

connect(m_pCommunicationBluetooth, SIGNAL(setNomPeripherique(QString)),
    this, SLOT(setNomPeripherique(QString)));
}

```

9.1.3.8 void Clhm : :deconnecterJoueur() [private, slot]

Références [LAYER_LOGO](#), [LOGO_ATTENTECONNECTION](#), [resetSeance\(\)](#), et [setLayerEcran\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    setLayerEcran(LAYER_LOGO);
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONNECTION);
    m_pQLabelLogoNom->setText("");
    m_pQLabelLogoNomMessage->setText("");
    resetSeance();
}

```

9.1.3.9 void Clhm : :envoyerCommande() [private, slot]

Références [activerConsole\(\)](#), [m_pTrame](#), [quitte\(\)](#), et [CTrame : :traiterTrame\(\)](#).

Référencé par [gererArguments\(\)](#).

```

{
    QString texte = m_pConsole->text();

    QTimer::singleShot(300, this, SLOT(activerConsole()));
    m_pConsole->setStyleSheet("QLineEdit#m_pConsole\n"
        "{\n"
        "    background-color: #00FF00;\n"
        "}");

    if(texte.startsWith("$TPA:"))
    {
        bool retour = m_pTrame->traiterTrame(texte);

        if (!retour)
            m_pConsole->setStyleSheet("QLineEdit#m_pConsole\n{\n"
                "background-color: #FF0000;\n}");

        return;
    }
    if(texte.startsWith("hide "))
    {
        QStringList args = texte.split(" ");
        if(args.length() == 2)
        {
            m_pConsole->hide();
            QTimer::singleShot(1000 * args.at(1).toInt(), m_pConsole, SLOT(show
                ()));
        }
        return;
    }
    if (texte.toLowerCase() == "quit")
    {
        quitte();
        return;
    }
    m_pConsole->setStyleSheet("QLineEdit#m_pConsole\n{\nbackground-color:
        #FF0000;\n}");
}

```


9.1.3.10 Clhm :finirSeance () [slot]

Références [calculerPourcentageQString\(\)](#), [CTable :finirSeance\(\)](#), [CTable :getBallesBonnes\(\)](#), [CTable :getBallesEnchainees\(\)](#), [CTable :getBallesObjectif\(\)](#), [CTable :getBallesTotal\(\)](#), [CTable :getZoneObjectif\(\)](#), [LAYER_RECAP](#), [m_pTable](#), [m_pTimerSeance](#), [rafraichirStats\(\)](#), [RECAP_STAT1](#), [RECAP_STAT1_ALT](#), [RECAP_STAT2](#), [RECAP_STAT3](#), [setLayerEcran\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    setLayerEcran(LAYER_RECAP);
    QString decalage = " ";
    m_pTimerSeance->stop();
    m_pTable->finirSeance();

    //=====
    // STAT 1
    if(m_pTable->getZoneObjectif() == ZONE_AUCUNE)
    {
        m_pQLabelLeftStat1TexteRecap->setText(decalage+ RECAP_STAT1_ALT);
        m_pQLabelLeftStat1NbRecap->setText(QString::number(m_pTable->
getBallesBonnes()) + " / " + QString::number(m_pTable->getBallesTotal()) );
        m_pQLabelLeftStat1PerRecap->setText(calculerPourcentageQString(m_pTable
->getBallesBonnes(),m_pTable->getBallesTotal()) );
    }
    else
    {
        m_pQLabelLeftStat1TexteRecap->setText(decalage+ RECAP_STAT1);
        m_pQLabelLeftStat1NbRecap->setText(QString::number(m_pTable->
getBallesObjectif()) + " / " + QString::number(m_pTable->getBallesTotal()) );
        m_pQLabelLeftStat1PerRecap->setText(calculerPourcentageQString(m_pTable
->getBallesObjectif(),m_pTable->getBallesTotal()) );
    }
    //=====
    // STAT 2

    m_pQLabelLeftStat2TexteRecap->setText(decalage+ RECAP_STAT2);
    rafraichirStats();

    //=====
    // STAT 3

    m_pQLabelLeftStat3TexteRecap->setText(decalage+ RECAP_STAT3);
    m_pQLabelLeftStat3NbRecap->setText(QString::number(m_pTable->
getBallesEnchainees()));

    //=====
    // STAT 4

    // TODO

    m_pHLayoutTableRecap->addWidget(m_pTable);
}
```

9.1.3.11 void Clhm :gererArguments () [private]

Références [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [connecterJoueur\(\)](#), [deconnecterJoueur\(\)](#), [DEV_BALLES-MAX](#), [envoyerCommande\(\)](#), [finirSeance\(\)](#), [impacterRandom\(\)](#), [CTable :m_args](#), [m_fontRougeStyle](#), [m_pTable](#), [CTable :setBallesMaximum\(\)](#), [setBallesMaximum\(\)](#), [setInfoConnectDemo\(\)](#), [setLayerEcranTable\(\)](#), [setNomPeripheriqueDemo\(\)](#), [setZoneObjectifRandom\(\)](#), et [setZoneRobotRandom\(\)](#).

Référencé par [Clhm\(\)](#).

```
{
    QStringList args = QCoreApplication::arguments(); // Verification des
arguments

    srand(QTime::currentTime().msec());

    m_pTable->m_args = args;
    //=====
    // DEV
```

```

if (!args.contains("-dev",Qt::CaseInsensitive)) // Suppression des boutons
pour les tests sans materiel
{
    m_pQLabelLogoNom->setText("");
    m_pQLabelLogoNomMessage->setText("");

    delete m_pTestZoneRobotRandom;
    delete m_pTestZoneObjectifRandom;
    delete m_pTestZoneRandom;
    delete m_pTestZoneReset;
}
else
{
    m_pQLabelTopLeftNomRecap->setText("MODE DEVELOPPEMENT");
    m_pQLabelTopLeftNomRecap->setStyleSheet(m_fontRougeStyle);

    m_pQLabelTopLeftNom->setText("MODE DEVELOPPEMENT");
    m_pQLabelTopLeftNom->setStyleSheet(m_fontRougeStyle);

    m_pQLabelLogoNom->setText("MODE DEVELOPPEMENT");
    m_pQLabelLogoNom->setStyleSheet(m_fontRougeStyle);
    m_pQLabelLogoNomMessage->setText("");

    m_pTable->setBallesMaximum(DEV_BALLESMAX);
}
if (!args.contains("-dev",Qt::CaseInsensitive) && !args.contains("-console"
,Qt::CaseInsensitive))
{
    delete m_pTestOutrepasser;
    delete m_pTestStartRecap;
}
else
{
    connect(m_pTestOutrepasser, SIGNAL(pressed()), this, SLOT(
commencerSeance()));
    connect(m_pTestStartRecap, SIGNAL(pressed()), this, SLOT(commencerSeance
()));
}
//=====
// WINDOWED
if (!args.contains("-windowed",Qt::CaseInsensitive)) // mode fenêtré
{
    this->showFullScreen();
}
//=====
// MODE DEMO
if (args.contains("-demo",Qt::CaseInsensitive)) // Mode de demonstration
{
    setBallesMaximum(DEV_BALLESMAX);

    QTimer::singleShot(2000, this, SLOT(connecterJoueur()));
    QTimer::singleShot(2000, this, SLOT(setInfoConnectDemo()));
    QTimer::singleShot(2000, this, SLOT(setNomPeripheriqueDemo()));
    QTimer::singleShot(3500, this, SLOT(setZoneObjectifRandom()));
    if (!args.contains("-norobot",Qt::CaseInsensitive))
        QTimer::singleShot(4000, this, SLOT(setZoneRobotRandom()));
    QTimer::singleShot(5000, this, SLOT(commencerSeance()));
    QTimer::singleShot(4000, this, SLOT(setLayerEcranTable()));

    for(int i=0 ; i < DEV_BALLESMAX ; i++)
    {
        QTimer::singleShot((6500) + i*1000, this, SLOT(balleEnJeu()));
        if(rand() % 10) // 1 chance sur 10 de rater
            QTimer::singleShot((7000) + i*1000, this, SLOT(impacterRandom()
));
    }
    QTimer::singleShot((7000) + (1+DEV_BALLESMAX)*1000, this, SLOT(
finirSeance()));
    QTimer::singleShot((15000) + (DEV_BALLESMAX)*1000, this, SLOT(
deconnecterJoueur()));
}
//=====
// CONSOLE
if (!args.contains("-console",Qt::CaseInsensitive))
{
    delete m_pConsole;
}
else
{
    connect(m_pConsole, SIGNAL(returnPressed()), this, SLOT(envoyerCommande
()));
}
}
}

```

9.1.3.12 float Clhm : :getRatioFenetreX ()

Références [TAILLE_FENETRE_DEFAULT_WIDTH](#).

```
{
    return ( (float)this->width()/TAILLE_FENETRE_DEFAULT_WIDTH );
}
```

9.1.3.13 float Clhm : :getRatioFenetreY ()

Références [TAILLE_FENETRE_DEFAULT_HEIGHT](#).

Référencé par [initialisationFenetre\(\)](#), [rafraichirCSS\(\)](#), [resetSeance\(\)](#), [setInfoConnect\(\)](#), et [setLayerEcran\(\)](#).

```
{
    return ( (float)this->height()/TAILLE_FENETRE_DEFAULT_HEIGHT );
}
```

9.1.3.14 QString Clhm : :getTimerSeanceString (unsigned int iTemps) [private]

Référencé par [pauserSeance\(\)](#), [reprendreSeance\(\)](#), et [setTimerSeance\(\)](#).

```
{
    unsigned int iMinutes = (unsigned int)((float)iTemps / (float)60);
    unsigned int iSecondes = iTemps - (iMinutes*60);

    QString zeroSeconde = ""; // CORRECTION LIEE AU ZERO

    if (iSecondes < 10)
        zeroSeconde = "0";

    return QString::number(iMinutes) + ":" + zeroSeconde + QString::number(
        iSecondes) + " ";
}
```

9.1.3.15 void Clhm : :impacterRandom () [private, slot]

Références [impacterZone\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{
    qDebug() << ">> DEV! " << Q_FUNC_INFO;
    impacterZone(rand() % 9);
}
```

9.1.3.16 Clhm : :impacterZone (uint8_t zone) [slot]

Paramètres

zone	enum voir const.h
------	-----------------------------------

Références [ballesTotalSurBallesMaximum\(\)](#), [CTable : :impacterZone\(\)](#), [m_pTable](#), et [rafraichirStats\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [impacterRandom\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    m_pTable->impacterZone(zone);
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());
    rafraichirStats();
}
```

9.1.3.17 void Clhm : :initialisationFenetre() [private, slot]

Références [getRatioFenetreY\(\)](#), [initialisationStats\(\)](#), [LOGO_ATTENTECONNECTION](#), [m_fontNoirStyle](#), [m_fontNormal](#), [m_fontTitreStyle](#), [m_pTable](#), [rafraichirCSS\(\)](#), [rafraichirHeure\(\)](#), [RATIO_ENTETE](#), [RECAP_TITRE_TEXTE](#), et [CTable : :setFiletTaille\(\)](#).

Référencé par [Clhm\(\)](#).

```
{
    rafraichirCSS(getRatioFenetreY());

    //=====
    // LAYER LOGO
    m_pQLabelLogo->setFixedHeight(this->height() / 2);
    m_pQLabelLogo->setFixedWidth(this->width() / 2);
    m_pQLabelLogoTexte->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelLogoHeure->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONNECTION);

    //=====
    // LAYER TABLE
    qDebug() << Q_FUNC_INFO << QTime::currentTime().toString();
    m_pQLabelTopRightHeure->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTimerSeance->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTimerSeanceRecap->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopMid->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopMid->setStyleSheet(m_fontNoirStyle);
    m_pTable->setFiletTaille(getRatioFenetreY());
    initialisationStats();

    //=====
    // LAYER RECAP
    m_pQLabelTopRightHeureRecap->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopTexteRecap->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopTexteRecap->setStyleSheet(m_fontTitreStyle);
    m_pQLabelTopTexteRecap->setFont(m_fontNormal);
    m_pQLabelTopTexteRecap->setText(RECAP_TITRE_TEXTE);

    //=====
    // HEURE
    rafraichirHeure();
}
```

9.1.3.18 void Clhm : :initialisationStats() [private]

Références [m_font](#), [m_fontNoirStyle](#), [m_fontNormal](#), [rafraichirStats\(\)](#), et [TABLE_STAT1](#).

Référencé par [initialisationFenetre\(\)](#).

```
{
    m_pQLabelLeftStat1Texte->setFont(m_font);
    m_pQLabelLeftStat1Nb->setFont(m_font);

    m_pQLabelLeftStat1Texte->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLeftStat1Nb->setStyleSheet(m_fontNoirStyle);

    m_pQLabelLeftStat1Texte->setText(TABLE_STAT1);

    //RECAP
    m_pQLabelLeftStat1TexteRecap->setFont(m_font);
    m_pQLabelLeftStat1TexteRecap->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLeftStat2TexteRecap->setFont(m_font);
    m_pQLabelLeftStat2TexteRecap->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLeftStat3TexteRecap->setFont(m_font);
    m_pQLabelLeftStat3TexteRecap->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLeftStat4TexteRecap->setFont(m_font);
    m_pQLabelLeftStat4TexteRecap->setStyleSheet(m_fontNoirStyle);

    m_pQLabelLeftStat1PerRecap->setFont(m_font);
    m_pQLabelLeftStat1PerRecap->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLeftStat2PerRecap->setFont(m_font);
    m_pQLabelLeftStat2PerRecap->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLeftStat3PerRecap->setFont(m_font);
    m_pQLabelLeftStat3PerRecap->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLeftStat4PerRecap->setFont(m_font);
    m_pQLabelLeftStat4PerRecap->setStyleSheet(m_fontNoirStyle);
}
```

```

m_pQLabelLeftStat1NbRecap->setFont (m_fontNormal);
m_pQLabelLeftStat1NbRecap->setStyleSheet (m_fontNoirStyle);
m_pQLabelLeftStat2NbRecap->setFont (m_fontNormal);
m_pQLabelLeftStat2NbRecap->setStyleSheet (m_fontNoirStyle);
m_pQLabelLeftStat3NbRecap->setFont (m_fontNormal);
m_pQLabelLeftStat3NbRecap->setStyleSheet (m_fontNoirStyle);
m_pQLabelLeftStat4NbRecap->setFont (m_fontNormal);
m_pQLabelLeftStat4NbRecap->setStyleSheet (m_fontNoirStyle);

m_pQLabelLeftStat1TexteRecap->clear();
m_pQLabelLeftStat2TexteRecap->clear();
m_pQLabelLeftStat3TexteRecap->clear();
m_pQLabelLeftStat4TexteRecap->clear();

m_pQLabelLeftStat1PerRecap->clear();
m_pQLabelLeftStat2PerRecap->clear();
m_pQLabelLeftStat3PerRecap->clear();
m_pQLabelLeftStat4PerRecap->clear();

m_pQLabelLeftStat1NbRecap->clear();
m_pQLabelLeftStat2NbRecap->clear();
m_pQLabelLeftStat3NbRecap->clear();
m_pQLabelLeftStat4NbRecap->clear();

    rafraichirStats();
}

```

9.1.3.19 void Clhm : :pauserSeance () [slot]

Références [CSS_TIMER_OFF](#), [getTimerSeanceString\(\)](#), [m_iTempsSeance](#), et [m_pTimerSeance](#).

Référencé par [connecterSignaux\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;

    m_pTimerSeance->stop();

    m_pQLabelTimerSeance->setStyleSheet (CSS_TIMER_OFF);
    m_pQLabelTimerSeance->setText (QString::fromUtf8(" ") +
        getTimerSeanceString(m_iTempsSeance));
}

```

9.1.3.20 void Clhm : :quitter () [slot]

Référencé par [connecterSignaux\(\)](#), [envoyerCommande\(\)](#), et [raccourcisClavier\(\)](#).

```

{
    close();
}

```

9.1.3.21 void Clhm : :raccourcisClavier () [private]

Références [quitter\(\)](#).

Référencé par [Clhm\(\)](#).

```

{
    // CTRL+Q
    QAction *actionQuitter = new QAction("&Quitter", this);
    actionQuitter->setShortcut (QKeySequence (Qt::CTRL + Qt::Key_Q));
    //actionQuitter->setShortcut (QKeySequence (QKeySequence::Quit)); // Ctrl+Q,
    // NE FONCTIONNE PAS SUR PI
    addAction(actionQuitter);
    connect(actionQuitter, SIGNAL(triggered()), this, SLOT(quitter()));
}

```

9.1.3.22 void Clhm : :rafraichirCSS (float ratio)

Références [ballesTotalSurBallesMaximum\(\)](#), [m_font](#), [m_fontNoirStyle](#), [m_fontNom](#), [m_fontNormal](#), [m_fontSmall](#), [m_fontTitreStyle](#), [m_pTable](#), [CTable : :rafraichirCSS\(\)](#), [TAILLE_TEXTE](#), [TAILLE_TEXTE_NOM](#), [TAILLE_TEXTE_NORMAL](#), et [TAILLE_TEXTE_SMALL](#).

```

{
    //=====
    //  SPECIFIQUE AU LOGO
    //=====

    m_font.setBold(true);
    m_fontSmall.setBold(true);
    m_fontNormal.setBold(true);
    m_fontNom.setBold(true);

    m_font.setPointSize((int) (TAILLE_TEXTE*ratio));
    m_fontSmall.setPointSize((int) (TAILLE_TEXTE_SMALL*ratio));
    m_fontNormal.setPointSize((int) (TAILLE_TEXTE_NORMAL*ratio));
    m_fontNom.setPointSize((int) (TAILLE_TEXTE_NOM*ratio));
    m_pQLabelLogoTexte->setFont(m_font);
    m_pQLabelLogoTexte->setStyleSheet(m_fontTitreStyle);
    m_pQLabelLogoHeure->setFont(m_fontNormal);
    m_pQLabelLogoHeure->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLogoNom->setFont(m_font);
    m_pQLabelLogoNomMessage->setFont(m_font);
    m_pQLabelLogoNomMessage->setStyleSheet(m_fontNoirStyle);

    //=====
    //  SPECIFIQUE A LA TABLE
    //=====

    m_pQLabelTopMid->setFont(m_fontNormal);
    m_pQLabelTopLeftNom->setFont(m_fontNom);
    m_pQLabelTopLeftNomPeripherique->setFont(m_fontSmall);
    m_pQLabelTopRightHeure->setFont(m_fontNormal);
    m_pQLabelTopRightHeure->setStyleSheet(m_fontNoirStyle);
    m_pQLabelTimerSeance->setFont(m_fontNormal);
    m_pQLabelTimerSeanceRecap->setFont(m_fontNormal);
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());

    m_pTable->rafraichirCSS(ratio);

    //=====
    //  SPECIFIQUE AU RECAP
    //=====
    m_pQLabelTopRightHeureRecap->setFont(m_fontNormal);
    m_pQLabelTopRightHeureRecap->setStyleSheet(m_fontNoirStyle);
    m_pQLabelTopLeftNomRecap->setFont(m_fontNom);
    m_pQLabelTopLeftNomPeripheriqueRecap->setFont(m_fontSmall);
}

```

9.1.3.23 Clhm : :rafraichirCSS() [slot]

Rafraichit le CSS lié à l'affichage (fontes, couleurs)

Paramètres

<i>ratio</i>	
--------------	--

Références [getRatioFenetreY\(\)](#).

Référencé par [connecterSignaux\(\)](#), [initialisationFenetre\(\)](#), et [resetSeance\(\)](#).

```
{ rafraichirCSS(getRatioFenetreY()); }
```

9.1.3.24 void Clhm : :rafraichirHeure() [slot]

Référencé par [connecterSignaux\(\)](#), et [initialisationFenetre\(\)](#).

```

{
    QString zeroMinute = "", zeroHeure = "";    // CORRECTION LIEE AU ZERO
    if (QTime::currentTime().hour() < 10)
        zeroHeure = "0";
    if (QTime::currentTime().minute() < 10)
        zeroMinute = "0";

    QString format = zeroHeure + QString::number(QTime::currentTime().hour()) +
        ":" + zeroMinute + QString::number(QTime::currentTime().minute()) + " ";
}

```

```

// EASTER EGG ( ° °)

    if (QTime::currentTime().hour() == 13 &&
        QTime::currentTime().minute() == 37){ format = "L3:3T";}
    m_pQLabelTopRightHeureRecap->setText(format);
    m_pQLabelTopRightHeure->setText(format);
    m_pQLabelLogoHeure->setText(format);
}

```

9.1.3.25 void Clhm : :rafraichirStats () [private]

Références [calculerPourcentageQString\(\)](#), [CTable : :getBallesHorsTable\(\)](#), [CTable : :getBallesTotal\(\)](#), et [m_pTable](#).

Référencé par [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [finirSeance\(\)](#), [impacterZone\(\)](#), [initialisationStats\(\)](#), [resetSeance\(\)](#), [setZoneObjectif\(\)](#), et [setZoneObjectifRandom\(\)](#).

```

{
    m_pQLabelLeftStat1Nb->setText (QString::number(m_pTable->getBallesHorsTable(
    )) + " " + calculerPourcentageQString(m_pTable->getBallesHorsTable(), m_pTable->
    getBallesTotal() - 1));
    m_pQLabelLeftStat2NbRecap->setText (QString::number(m_pTable->
    getBallesHorsTable()) + " / " + QString::number(m_pTable->getBallesTotal()) );
    m_pQLabelLeftStat2PerRecap->setText (calculerPourcentageQString(m_pTable->
    getBallesHorsTable(), m_pTable->getBallesTotal()) );
}

```

9.1.3.26 void Clhm : :rafraichirTimerSeance () [slot]

Références [CSS_TIMER_ON](#), [m_iTempsSeance](#), et [setTimerSeance\(\)](#).

Référencé par [connecterSignaux\(\)](#).

```

{
    //    qDebug() << Q_FUNC_INFO;          //!< Spam
    m_iTempsSeance++;

    m_pQLabelTimerSeance->setStyleSheet(CSS_TIMER_ON);
    setTimerSeance(m_iTempsSeance);
}

```

9.1.3.27 Clhm : :reprenreSeance () [slot]

Références [CSS_TIMER_RES](#), [getTimerSeanceString\(\)](#), [m_iTempsSeance](#), et [m_pTimerSeance](#).

Référencé par [connecterSignaux\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;

    m_pTimerSeance->start(1000);
    m_pQLabelTimerSeance->setStyleSheet(CSS_TIMER_RES);
    m_pQLabelTimerSeance->setText (QString::fromUtf8(" ") +
    getTimerSeanceString(m_iTempsSeance));
}

```

9.1.3.28 Clhm : :resetSeance () [slot]

Références [getRatioFenetreY\(\)](#), [m_iTempsSeance](#), [m_pTable](#), [m_pTimerSeance](#), [rafraichirCSS\(\)](#), [rafraichirStats\(\)](#), [CTable : :resetSeance\(\)](#), [setInfoConnect\(\)](#), et [setTimerSeance\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [deconnecterJoueur\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    m_pTable->resetSeance();
}

```

```

    rafraichirCSS(getRatioFenetreY());
    rafraichirStats();
    setInfoConnect(m_pQLabelLogoNom->text());

    m_iTempsSeance = 0;
    m_pTimerSeance->stop();
    setTimerSeance(0);
}

```

9.1.3.29 Clhm : :setBallesMaximum (int *balles*) [slot]

Paramètres

<i>balles</i>	int
---------------	-----

Références [ballesTotalSurBallesMaximum\(\)](#), [m_fontNormal](#), [m_pTable](#), et [CTable : :setBallesMaximum\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO << "BALLES: " << balles;
    m_pTable->setBallesMaximum(balles);
    // rafraichirCSS(getRatioFenetreY()); // nécessaire pour refresh l'IHM car
    // ballemax est utilisé par elle aussi
    m_pQLabelTopMid->setFont(m_fontNormal);
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());
}

```

9.1.3.30 Clhm : :setInfoConnect (QString *nom*) [slot]

Paramètres

<i>nom</i>	QString nom du joueur
------------	-----------------------

Références [getRatioFenetreY\(\)](#), [LOGO_ATTENTECONFIGURATION](#), [m_fontNom](#), et [TAILLE_TEXTE_NOM](#).

Référencé par [connecterSignaux\(\)](#), [resetSeance\(\)](#), et [setInfoConnectDemo\(\)](#).

```

{
    if (nom == "")
        return;

    m_pQLabelTopLeftNomRecap->setText(nom);
    m_pQLabelTopLeftNom->setText(nom);
    // m_pQLabelLogoNom->setText(nom); affiche le nom du peripherique depuis
    // 1.1
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONFIGURATION);
    qDebug() << Q_FUNC_INFO << " nom : " << nom;

    float tailleNomRatio;
    if (nom.length() > 20)
        tailleNomRatio = (4.0 / (float) (nom.length())) + (1-4.0/20);
    else
        tailleNomRatio = 1.0;

    m_fontNom.setPointSize((int) (TAILLE_TEXTE_NOM*getRatioFenetreY()) *
        tailleNomRatio);
    // m_pQLabelTopLeftNom->setFont(m_fontNom); affiche le nom du
    // peripherique depuis 1.1
    m_pQLabelTopLeftNomRecap->setFont(m_fontNom);
}

```

9.1.3.31 void Clhm : :setInfoConnectDemo () [private, slot]

Références [IHM_NOMDETEST](#), et [setInfoConnect\(\)](#).

Référencé par [gererArguments\(\)](#).

```

{ setInfoConnect(IHM_NOMDETEST); }

```


9.1.3.32 Clhm : :setLayerEcran (uint8_t layer) [slot]

Paramètres

<i>layer</i>	enum voir const.h
--------------	-----------------------------------

Références [getRatioFenetreY\(\)](#), [m_pTable](#), et [CTable : :setLayerEcran\(\)](#).

Référencé par [commencerSeance\(\)](#), [connecterJoueur\(\)](#), [connecterSignaux\(\)](#), [deconnecterJoueur\(\)](#), [finirSeance\(\)](#), [setLayerEcranLogo\(\)](#), [setLayerEcranRecap\(\)](#), et [setLayerEcranTable\(\)](#).

```
{
    m_pTable->setLayerEcran(layer, getRatioFenetreY());
    m_pFenetres->setCurrentIndex(layer);
    qDebug() << Q_FUNC_INFO;
}
```

9.1.3.33 void Clhm : :setLayerEcranLogo () [private, slot]

Références [LAYER_LOGO](#), et [setLayerEcran\(\)](#).

```
{ setLayerEcran(LAYER_LOGO); }
```

9.1.3.34 void Clhm : :setLayerEcranRecap () [private, slot]

Références [LAYER_RECAP](#), et [setLayerEcran\(\)](#).

```
{ setLayerEcran(LAYER_RECAP); }
```

9.1.3.35 void Clhm : :setLayerEcranTable () [private, slot]

Références [LAYER_TABLE](#), et [setLayerEcran\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{ setLayerEcran(LAYER_TABLE); }
```

9.1.3.36 Clhm : :setNomPeripherique (QString nom = "Peripherique DEMO") [slot]

Paramètres

<i>nom</i>	QString nom du périphérique
------------	-----------------------------

Références [LOGO_JOUEUR_CONNECTE](#), et [m_fontVertStyle](#).

Référencé par [connecterSignaux\(\)](#), et [setNomPeripheriqueDemo\(\)](#).

```
{
    m_pQLabelLogoNom->setText(nom);
    m_pQLabelTopLeftNomPeripherique->setText("(" + nom + ")");
    m_pQLabelTopLeftNomPeripheriqueRecap->setText("(" + nom + ")");

    m_pQLabelLogoNom->setStyleSheet(m_fontVertStyle);
    m_pQLabelTopLeftNomPeripherique->setStyleSheet(m_fontVertStyle);
    m_pQLabelTopLeftNomPeripheriqueRecap->setStyleSheet(m_fontVertStyle);

    m_pQLabelLogoNomMessage->setText(LOGO_JOUEUR_CONNECTE);
}
```

9.1.3.37 void Clhm : :setNomPeripheriqueDemo () [private, slot]

Références [IHM_PERIPHERIQUEDETEST](#), et [setNomPeripherique\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{ setNomPeripherique(IHM_PERIPHERIQUEDETEST); }
```

9.1.3.38 void Clhm : :setTimerSeance (unsigned int iTemps = 0) [private]

Paramètres

<i>Minutes,- Secondes</i>	
-------------------------------	--

Références [getTimerSeanceString\(\)](#).

Référencé par [Clhm\(\)](#), [commencerSeance\(\)](#), [rafraichirTimerSeance\(\)](#), et [resetSeance\(\)](#).

```
{
    m_pQLabelTimerSeance->setText( getTimerSeanceString(iTemps) );
    m_pQLabelTimerSeanceRecap->setText( getTimerSeanceString(iTemps) );
}
```

9.1.3.39 Clhm : :setZoneObjectif (uint8_t zone) [slot]

Paramètres

<i>zone</i>	enum voir const.h
-------------	-----------------------------------

Références [m_pTable](#), [rafraichirStats\(\)](#), et [CTable : :setZoneObjectif\(\)](#).

Référencé par [connecterSignaux\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO << "ZONE: " << zone;
    m_pTable->setZoneObjectif(zone);
    rafraichirStats();
}
```

9.1.3.40 void Clhm : :setZoneObjectifRandom () [private, slot]

Références [m_pTable](#), [rafraichirStats\(\)](#), et [CTable : :setZoneObjectif\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{
    qDebug() << ">> DEV! " << Q_FUNC_INFO;
    m_pTable->setZoneObjectif(rand() % 9);
    rafraichirStats();
}
```

9.1.3.41 Clhm : :setZoneRobot (uint8_t zone) [slot]

Paramètres

<i>zone</i>	enum voir const.h
-------------	-----------------------------------

Références [m_pTable](#), et [CTable : :setZoneRobot\(\)](#).

Référencé par [connecterSignaux\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO << "ZONE: " << zone;
    m_pTable->setZoneRobot(zone);
}
```

9.1.3.42 void Clhm : :setZoneRobotRandom () [private, slot]

Références [m_pTable](#), et [CTable : :setZoneRobot\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{
    qDebug() << ">> DEV! " << Q_FUNC_INFO;
    int random = -1;

    while (random < 0 || random == m_pTable->getZoneObjectif())
    {
        srand(QTime::currentTime().msec());
        random = rand() % 9;
        m_pTable->setZoneRobot(random);
    }
}
```

9.1.4 Documentation des données membres

9.1.4.1 QFont Clhm : :m_font [private]

Référencé par [initialisationStats\(\)](#), et [rafraichirCSS\(\)](#).

9.1.4.2 QString Clhm : :m_fontNoirStyle [private]

Référencé par [Clhm\(\)](#), [initialisationFenetre\(\)](#), [initialisationStats\(\)](#), et [rafraichirCSS\(\)](#).

9.1.4.3 QFont Clhm : :m_fontNom [private]

Référencé par [rafraichirCSS\(\)](#), et [setInfoConnect\(\)](#).

9.1.4.4 QFont Clhm : :m_fontNormal [private]

Référencé par [balleEnJeu\(\)](#), [initialisationFenetre\(\)](#), [initialisationStats\(\)](#), [rafraichirCSS\(\)](#), et [setBalles-Maximum\(\)](#).

9.1.4.5 QString Clhm : :m_fontRougeStyle [private]

Référencé par [Clhm\(\)](#), et [gererArguments\(\)](#).

9.1.4.6 QFont Clhm : :m_fontSmall [private]

Référencé par [rafraichirCSS\(\)](#).

9.1.4.7 QString Clhm : :m_fontTitreStyle [private]

Référencé par [Clhm\(\)](#), [initialisationFenetre\(\)](#), et [rafraichirCSS\(\)](#).

9.1.4.8 QString Clhm : :m_fontVertStyle [private]

Référencé par [Clhm\(\)](#), et [setNomPeripherique\(\)](#).

9.1.4.9 unsigned int Clhm : :m_iTempsSeance [private]

Référencé par [Clhm\(\)](#), [commencerSeance\(\)](#), [pauserSeance\(\)](#), [rafraichirTimerSeance\(\)](#), [reprendreSeance\(\)](#), et [resetSeance\(\)](#).

9.1.4.10 CommunicationBluetooth* Clhm : :m_pCommunicationBluetooth [private]

Référencé par [Clhm\(\)](#), [connecterSignaux\(\)](#), et [~Clhm\(\)](#).

9.1.4.11 CTable* Clhm : :m_pTable [private]

Référencé par [balleEnJeu\(\)](#), [ballesTotalSurBallesMaximum\(\)](#), [Clhm\(\)](#), [commencerSeance\(\)](#), [finirSeance\(\)](#), [generArguments\(\)](#), [impacterZone\(\)](#), [initialisationFenetre\(\)](#), [rafraichirCSS\(\)](#), [rafraichirStats\(\)](#), [resetSeance\(\)](#), [setBallesMaximum\(\)](#), [setLayerEcran\(\)](#), [setZoneObjectif\(\)](#), [setZoneObjectifRandom\(\)](#), [setZoneRobot\(\)](#), et [setZoneRobotRandom\(\)](#).

9.1.4.12 QThread* Clhm : :m_pThreadCommunicationBluetooth [private]

Référencé par [Clhm\(\)](#), [connecterSignaux\(\)](#), et [~Clhm\(\)](#).

9.1.4.13 QTimer* Clhm : :m_pTimerHeure [private]

Référencé par [Clhm\(\)](#), et [connecterSignaux\(\)](#).

9.1.4.14 QTimer* Clhm : :m_pTimerSeance [private]

Référencé par [Clhm\(\)](#), [commencerSeance\(\)](#), [connecterSignaux\(\)](#), [finirSeance\(\)](#), [pauserSeance\(\)](#), [reprendreSeance\(\)](#), et [resetSeance\(\)](#).

9.1.4.15 CTrame* Clhm : :m_pTrame [private]

Référencé par [Clhm\(\)](#), [connecterSignaux\(\)](#), et [envoyerCommande\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

- [ihm.h](#)
- [ihm.cpp](#)

9.2 Référence de la classe CommunicationBluetooth

Assure la réception des trames via le Bluetooth.

```
#include <communicationbluetooth.h>
```

Types publics

- enum [Mode](#) { [Scrutation](#), [Evenement](#) }
Mode de gestion du port de communication.

Connecteurs publics

- void [main](#) ()
Le corps du thread assurant la réception des données via le Bluetooth.
- void [finir](#) ()
Met fin au Thread.
- void [terminer](#) ()
Termine le thread.
- void [lirePort](#) ()
Lit les données disponibles sur le port Bluetooth.
- void [surveillerConnexion](#) ()
Surveille l'état du service RFCOMM.

Signaux

- void [nouvellesDonneesRecues](#) (QString donneesRecues)
Signale les données reçues sur le port Bluetooth.
- void [deconnecterJoueur](#) ()
Signale une déconnexion du Bluetooth.
- void [connecterJoueur](#) ()

- *Signale une connexion du Bluetooth.*
– void **setNomPeripherique** (QString nom)
– *Affiche le nom du périphérique connecté*

Fonctions membres publiques

- **CommunicationBluetooth** (QString nomPort="rfcomm0", Mode mode=**Scrutation**, QObject *parent=0)
– *Constructeur.*
- **~CommunicationBluetooth** ()
– *Destructeur.*
- void **ouvrir** ()
– *Ouvre et configure le port série.*
- void **fermer** ()
– *Ferme le port série.*
- int **getEtatRFCOMM** ()
– *Retourne l'état du port RFCOMM.*

Fonctions membres privées

- void **msleep** (unsigned long sleepMS)
– *Temporise l'exécution du Thread.*
- void **annuler** ()
– *Termine la temporisation en cours.*
- void **attendrePeriode** ()
– *Attend une durée PERIODE_SURVEILLANCE.*
- void **demarrerRFCOMM** ()
– *Démarre le service RFCOMM.*
- void **redemarrerRFCOMM** ()
– *Redémarre le service RFCOMM.*
- QString **lireEtatRFCOMM** ()
– *Lit l'état du service RFCOMM.*
- QString **lireEtatServiceRFCOMM** ()
– void **arreterRFCOMM** ()
– *Arrête le service RFCOMM.*
- void **recupererNomBluetooth** ()
– *Récupère le nom de l'appareil connecté*

Attributs privés

- QTextSerialPort * **m_pPortSerie**
– *Agrégation vers la classe QTextSerialPort.*
- bool **fini**
– *état du Thread.*
- QMutex **mutex**
– *pour gérer les temporisations*
- QWaitCondition **waitCondition**
– *pour les temporisations*
- int **periode**
– *la périodicité du thread*
- int **etatRFCOMM**
– *état du service RFCOMM.*
- QTimer * **timerSurveillance**
– *pour surveiller périodiquement l'état de la connexion Bluetooth*

9.2.1 Documentation des énumérations membres

9.2.1.1 enum CommunicationBluetooth : :Mode

Valeurs énumérées :

Scrutation

Evenement

```
{
    Scrutation,
    Evenement
};
```

9.2.2 Documentation des constructeurs et destructeur

**9.2.2.1 CommunicationBluetooth : :CommunicationBluetooth (QString *nomPort* = "rfcomm0",
CommunicationBluetooth : :Mode *mode* = Scrutation, QObject * *parent* = 0) [explicit]**

Paramètres

<i>nomPort</i>	QString le nom du fichier de périphérique Bluetooth (par défaut rfcomm0)
<i>parent</i>	QObject Adresse de l'objet Qt parent (par défaut 0)

Références [demarrerRFCOMM\(\)](#), [m_pPortSerie](#), [periode](#), [Scrutation](#), [surveillerConnexion\(\)](#), et [timer-Surveillance](#).

```
        : QObject(parent), fini(false), periode(
    PERIODE_SURVEILLANCE), etatRFCOMM(RFCOMM_ARRETE)
{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    demarrerRFCOMM();

    // cf. méthode main()
    if(mode == CommunicationBluetooth::Scrutation)
    {
        m_pPortSerie = new QextSerialPort(QextSerialPort::Polling, this);
        m_pPortSerie->setPortName(QString("/dev/") + nomPort);
        surveillerConnexion();
        #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Mode Polling (par
        scrutation)");
        #endif
    }
    else /* CommunicationBluetooth::Evenement */
    {
        m_pPortSerie = new QextSerialPort(QextSerialPort::EventDriven, this);
        m_pPortSerie->setPortName(QString("/dev/") + nomPort);
        surveillerConnexion();

        timerSurveillance = new QTimer(this);
        timerSurveillance->setInterval(periode);
        connect(timerSurveillance, SIGNAL(timeout()), this, SLOT(
        surveillerConnexion()));
        timerSurveillance->start();
        #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Mode EventDriven (par
        évènement)");
        #endif
    }
}
```

9.2.2.2 CommunicationBluetooth : :~CommunicationBluetooth ()

Références [arreterRFCOMM\(\)](#), et [fermer\(\)](#).

```
{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    fermer();
    arreterRFCOMM();
}
```

9.2.3 Documentation des fonctions membres

9.2.3.1 CommunicationBluetooth : :annuler() [private]

Référencé par [finir\(\)](#).

```
{
    waitCondition.wakeAll();
}
```

9.2.3.2 CommunicationBluetooth : :arreterRFCOMM() [private]

Référencé par [~CommunicationBluetooth\(\)](#).

```
{
    FILE *resultat;
    resultat = popen("sudo systemctl stop rfcomm.service 2> /dev/null", "r");
    pclose(resultat);
}
```

9.2.3.3 CommunicationBluetooth : :attendrePeriode() [private]

Références [msleep\(\)](#), et [periode](#).

Référencé par [surveillerConnexion\(\)](#).

```
{
    this->msleep(periode); // en ms
}
```

9.2.3.4 CommunicationBluetooth : :connecterJoueur() [signal]

Référencé par [surveillerConnexion\(\)](#).

9.2.3.5 CommunicationBluetooth : :deconnecterJoueur() [signal]

Référencé par [surveillerConnexion\(\)](#).

9.2.3.6 CommunicationBluetooth : :demarrerRFCOMM() [private]

Référencé par [CommunicationBluetooth\(\)](#), et [surveillerConnexion\(\)](#).

```
{
    FILE *resultat;
    resultat = popen("sudo systemctl start rfcomm.service 2> /dev/null", "r");
    pclose(resultat);
}
```

9.2.3.7 CommunicationBluetooth : :fermer()

Références [lirePort\(\)](#), et [m_pPortSerie](#).

Référencé par [surveillerConnexion\(\)](#), et [~CommunicationBluetooth\(\)](#).

```
{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    if (m_pPortSerie->isOpen())
    {
        if (m_pPortSerie->queryMode() == QextSerialPort::EventDriven)
            disconnect(m_pPortSerie, SIGNAL(readyRead()), this, SLOT(lirePort()));
        m_pPortSerie->close();
    }
}
```

9.2.3.8 CommunicationBluetooth :finir () [slot]

Références [annuler\(\)](#), et [fini](#).

Référencé par [Clhm : ~Clhm\(\)](#).

```
{  
    fini = true;  
    annuler();  
}
```

9.2.3.9 CommunicationBluetooth :getEtatRFCOMM ()

Références [etatRFCOMM](#).

```
{ return etatRFCOMM; }
```

9.2.3.10 CommunicationBluetooth :lireEtatRFCOMM () [private]

Renvoie

QString

Référencé par [surveillerConnexion\(\)](#).

```
{  
    FILE *resultat;  
    char ligne[1024];  
    QString reponse;  
  
    // lit l'état de la connexion  
    resultat = popen("rfcomm -a", "r");  
    fgets(ligne, 1024, resultat);  
    while (!feof(resultat))  
    {  
        reponse += ligne;  
        fgets(ligne, 1024, resultat);  
    }  
    pclose(resultat);  
  
    return reponse;  
}
```

9.2.3.11 QString CommunicationBluetooth :lireEtatServiceRFCOMM () [private]

Référencé par [surveillerConnexion\(\)](#).

```
{  
    FILE *resultat;  
    char ligne[1024];  
    QString reponse;  
  
    // lit l'état de la connexion  
    resultat = popen("sudo systemctl status rfcomm.service 2> /dev/null", "r");  
    fgets(ligne, 1024, resultat);  
    while (!feof(resultat))  
    {  
        reponse += ligne;  
        fgets(ligne, 1024, resultat);  
    }  
    pclose(resultat);  
  
    return reponse;  
}
```


9.2.3.12 CommunicationBluetooth : lirePort () [slot]

Références [m_pPortSerie](#), [msleep\(\)](#), et [nouvellesDonneesRecues\(\)](#).

Référencé par [fermer\(\)](#), [main\(\)](#), et [ouvrir\(\)](#).

```
{
    if (!m_pPortSerie->isOpen())
        return;

    QByteArray donnees;

    // lecture des données disponibles
    while (m_pPortSerie->bytesAvailable())
    {
        donnees += m_pPortSerie->readAll();
        this->msleep(20); // en ms
        //::usleep(100000); // cf. timeout
    }

    QString donneesRecues = QString(QString::fromUtf8(donnees.data()));
    if(!donneesRecues.isEmpty())
    {
        #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
        QString::fromUtf8("Données reçues : ") << donneesRecues;
        #endif
        emit nouvellesDonneesRecues(donneesRecues);
    }
}
```

9.2.3.13 CommunicationBluetooth : main () [slot]

Références [fini](#), [lirePort\(\)](#), [m_pPortSerie](#), et [surveillerConnexion\(\)](#).

```
{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
    QString::fromUtf8("Communication Bluetooth démarrée");
    #endif

    if(m_pPortSerie->queryMode() == QextSerialPort::Polling)
    {
        // On interroge périodiquement le port de communication et on
        // surveille l'état de la (connexion)
        while(!fini)
        {
            lirePort();

            surveillerConnexion();
        }
    }
    else
    {
        // On crée une boucle d'événements nécessaire pour gérer les signaux
        // (readyRead() et timeout())
        while(!fini)
        {
            QApplication::processEvents();
            //QApplication::processEvents(QEventLoop::ExcludeUserInputEvents,
            //periode);
        }
    }
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
    QString::fromUtf8("Communication Bluetooth arrêtée");
    #endif
}
```

9.2.3.14 CommunicationBluetooth : msleep (unsigned long sleepMS) [private]

Paramètres

<i>sleepMS</i>	unsigned long durée de la temporisation en millisecondes
----------------	--

Référencé par [attendrePeriode\(\)](#), [lirePort\(\)](#), et [surveillerConnexion\(\)](#).

```
{
    waitCondition.wait(&mutex, sleepMS);
}
```

9.2.3.15 CommunicationBluetooth : :nouvellesDonneesRecues (QString *donneesRecues*) [signal]

Paramètres

<i>donnees- Recues</i>	
----------------------------	--

Référencé par [lirePort\(\)](#).

9.2.3.16 CommunicationBluetooth : :ouvrir ()

Références [lirePort\(\)](#), [m_pPortSerie](#), et [recupererNomBluetooth\(\)](#).

Référencé par [surveillerConnexion\(\)](#).

```
{
    if (m_pPortSerie->isOpen())
    {
        #ifndef QT_NO_DEBUG
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Le port %1 est ouvert").
            arg(m_pPortSerie->portName());
        #endif
        return;
    }
    #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    // ouverture du port
    m_pPortSerie->open(QIODevice::ReadWrite);
    if (!m_pPortSerie->isOpen())
    {
        #ifndef QT_NO_DEBUG
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Le port %1 n'est pas
            ouvert").arg(m_pPortSerie->portName());
        #endif
        return;
    }
    else
    {
        #ifndef QT_NO_DEBUG
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Ouverture du port %1
            réussie").arg(m_pPortSerie->portName());
        #endif
        // configuration du port
        m_pPortSerie->setBaudRate(BAUD9600);
        m_pPortSerie->setDataBits(DATA_8);
        m_pPortSerie->setParity(PAR_NONE);
        m_pPortSerie->setStopBits(STOP_1);
        m_pPortSerie->setFlowControl(FLOW_OFF);
        if (m_pPortSerie->queryMode() == QextSerialPort::EventDriven)
        {
            if (m_pPortSerie->isOpen())
            {
                connect(m_pPortSerie, SIGNAL(readyRead()), this, SLOT(lirePort()));
                recupererNomBluetooth();
            }
        }
    }
}
```

9.2.3.17 CommunicationBluetooth : :recupererNomBluetooth () [private]

Références [setNomPeripherique\(\)](#).

Référencé par [ouvrir\(\)](#).

```
{
    FILE *resultat;
```

```

char ligne[1024];
QString reponse;

resultat = popen("rfcomm -a", "r");    // récupération de l'adresse
fgets(ligne, 1024, resultat);
while (!feof(resultat))
{
    reponse += ligne;
    fgets(ligne, 1024, resultat);
}
pclose(resultat);

QString adresseMAC;

if (reponse.size() == 0)
{
    qDebug() << Q_FUNC_INFO << "!!\\ NE PEUT PAS RECUPERER L'ADRESSE MAC /!
    \\";
    return;
}

adresseMAC = reponse.mid(30, 17);

QString commande = "hcitool name " + adresseMAC;

reponse = "";

resultat = popen(commande.toAscii(), "r");    // récupération du nom
fgets(ligne, 1024, resultat);
while (!feof(resultat))
{
    reponse += ligne;
    fgets(ligne, 1024, resultat);
}
pclose(resultat);

if (reponse.size() == 0)
{
    qDebug() << Q_FUNC_INFO << "!!\\ NE PEUT PAS RECUPERER LE NOM DE
    L'APPAREIL /!\\\\";
    return;
}

qDebug() << Q_FUNC_INFO << "Nom de l'appareil connecte : " << reponse.
    trimmed();

emit setNomPeripherique(reponse.trimmed());
}

```

9.2.3.18 CommunicationBluetooth : :redemarrerRFCOMM() [private]

Référencé par [surveillerConnexion\(\)](#).

```

{
    FILE *resultat;
    resultat = popen("sudo systemctl restart rfcomm.service 2> /dev/null", "r")
    ;
    pclose(resultat);
}

```

9.2.3.19 CommunicationBluetooth : :setNomPeripherique (QString nom) [signal]

Paramètres

<i>nom</i>	QString nom du périphérique
------------	-----------------------------

Référencé par [recupererNomBluetooth\(\)](#).

9.2.3.20 CommunicationBluetooth : :surveillerConnexion() [slot]

Références [attendrePeriode\(\)](#), [connecterJoueur\(\)](#), [deconnecterJoueur\(\)](#), [demarrerRFCOMM\(\)](#), [etatRFCOMM-M](#), [fermer\(\)](#), [lireEtatRFCOMM\(\)](#), [lireEtatServiceRFCOMM\(\)](#), [m_pPortSerie](#), [msleep\(\)](#), [ouvrir\(\)](#), [redemarrerRFCOMM\(\)](#), [RFCOMM_ARRETE](#), [RFCOMM_CONNECTE](#), et [RFCOMM_FERME](#).

Référencé par [CommunicationBluetooth\(\)](#), et [main\(\)](#).

```
{
    if(m_pPortSerie->bytesAvailable())
        return;

    attendrePeriode();

    QString etat = lireEtatServiceRFCOMM();

    if(!etat.isEmpty())
    {
        if(etat.contains("inactive"))
        {
            #ifndef QT_NO_DEBUG
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth rfcomm0
inactif");
            #endif
            demarrerRFCOMM();
            //return;
            etat = lireEtatServiceRFCOMM();
        }
        if(etat.contains("active"))
        {
            #ifndef QT_NO_DEBUG
            //qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth rfcomm0
actif");
            #endif
            etat = lireEtatRFCOMM();

            #ifndef QT_NO_DEBUG
            //qDebug() << Q_FUNC_INFO << etat << etatRFCOMM;
            #endif
            if(!etat.isEmpty())
            {
                if(etat.contains("connected"))
                {
                    if(etatRFCOMM != RFCOMM_CONNECTE)
                    {
                        etatRFCOMM = RFCOMM_CONNECTE;
                        #ifndef QT_NO_DEBUG
                        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth
rfcomm0 connecté");
                        #endif
                        ouvrir();
                        emit connecterJoueur();
                        return;
                    }
                }
                else if(etat.contains("closed"))
                {
                    if(etatRFCOMM != RFCOMM_FERME)
                    {
                        etatRFCOMM = RFCOMM_FERME;
                        #ifndef QT_NO_DEBUG
                        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth
rfcomm0 fermé");
                        #endif
                        fermer();
                        this->msleep(100); // en ms
                        redemarrerRFCOMM();
                        emit deconnecterJoueur();
                        return;
                    }
                }
            }
        }
        else
        {
            if(etatRFCOMM != RFCOMM_ARRETE)
            {
                etatRFCOMM = RFCOMM_ARRETE;
                #ifndef QT_NO_DEBUG
                qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucune
connexion Bluetooth rfcomm0");
                #endif
            }
        }
    }
}
```

9.2.3.21 CommunicationBluetooth : :terminer () [slot]

```
{
}
```

9.2.4 Documentation des données membres

9.2.4.1 int CommunicationBluetooth : :etatRFCOMM [private]

Référencé par [getEtatRFCOMM\(\)](#), et [surveillerConnexion\(\)](#).

9.2.4.2 bool CommunicationBluetooth : :fini [private]

Référencé par [finir\(\)](#), et [main\(\)](#).

9.2.4.3 QextSerialPort* CommunicationBluetooth : :m_pPortSerie [private]

Référencé par [CommunicationBluetooth\(\)](#), [fermer\(\)](#), [lirePort\(\)](#), [main\(\)](#), [ouvrir\(\)](#), et [surveillerConnexion\(\)](#).

9.2.4.4 QMutex CommunicationBluetooth : :mutex [private]

9.2.4.5 int CommunicationBluetooth : :periode [private]

Référencé par [attendrePeriode\(\)](#), et [CommunicationBluetooth\(\)](#).

9.2.4.6 QTimer* CommunicationBluetooth : :timerSurveillance [private]

Référencé par [CommunicationBluetooth\(\)](#).

9.2.4.7 QWaitCondition CommunicationBluetooth : :waitCondition [private]

La documentation de cette classe a été générée à partir des fichiers suivants :

- [communicationbluetooth.h](#)
- [communicationbluetooth.cpp](#)

9.3 Référence de la classe CTable

```
#include <table.h>
```

Connecteurs publics

- void [rafraichirInactif](#) ()

Fonctions membres publiques

- [CTable](#) (QWidget *parent=0)
- void [setFiletTaille](#) (float ratio)
- void [rafraichirCSS](#) (float ratio)
Définis la hauteur du filet en utilisant la hauteur de la fenêtre actuelle.
- bool [impacterZone](#) (uint8_t numeroZone)
Rafraichit le CSS en utilisant la hauteur de la fenêtre actuelle.
- void [setZoneRobot](#) (uint8_t zone)
Calcul et affiche l'impact sur l'IHM et la table.
- void [setZoneObjectif](#) (uint8_t zone)
Place le robot sur la table.
- void [resetSeance](#) ()
Place la zone objectif sur la table.
- void [resetStatistiques](#) ()

- void `balleEnJeu` ()
Reset complet des variables des statistiques.
- void `resetNbBallesZone` ()
Reset des statistiques uniquement.
- void `finirSeance` ()
La balle a bien été mise en jeu par la machine, BalleTotal est incrémenté
- int `getBallesBonnes` ()
Reset du nombre de balles dans chaque zone.
- int `getBallesTotal` ()
Récupère le nombre de balles ayant été jouées et renvoyé par le joueur.
- int `getBallesMaximum` ()
Récupère le nombre de balles ayant été jouées par la machine (en comptant celles n'ayant pas atteint une zone car le joueur est mauvais (° °))
- int `getBallesObjectif` ()
Récupère le nombre de balles maximum pour la seance.
- int `getBallesEnchainees` ()
Récupère le nombre de balles correspondant à la zone objectif actuel.
- int `getBallesHorsTable` ()
Récupère le nombre de balles enchainées.
- int `getZoneToucheePrec` ()
Récupère le nombre de balles hors table.
- void `setBallesMaximum` (int nb)
Récupère la zone touchée précédente.
- uint8_t `getZoneObjectif` ()
Définis le nombre de balles maximum pour la séance.
- bool `getBalleCoteTablePrec` ()
Recupère le numero de la case Objectif.
- bool `getBalleCoteTable` ()
Récupère le coté de la table ou la balle a frappé en dernier.
- void `setLayerEcran` (uint8_t layer, float tailleFenetreY)
Récupère le coté de la table ou la balle a frappé juste avant la dernière.

Attributs publics

- QStringList `m_args`
- QLabel * `m_pOverlayText`

Fonctions membres privées

- void `rafraichirNbBallesZone` ()
Corrections lié au changement de fenetre.

Attributs privés

- QGridLayout * `m_pGridLayout`
Rafraichit le nombre de balles par zone.
- QVector< QLabel * > `m_pZones`
- QLabel * `m_pFilet`
- QLabel * `m_pOverlay`
- int `m_iBallesBonnes`
- int `m_iBallesMaximum`
- int `m_iBallesTotal`
- int `m_iBallesEnchainees`
- int `m_iBallesEnchaineesMax`
- int `m_iBallesHorsTable`
- int `m_iBallesDansZone` [NB_ZONES]
- int `m_iZoneTouchee`
- int `m_iZoneToucheePrec`
- int `m_iZoneRobot`
- int `m_iZoneObjectif`
- bool `m_bBalleCoteTable`
- bool `m_bBalleCoteTablePrec`
- QFont `m_font`
- QFont `m_fontBig`
- QFont `m_fontOverlay`

- QString m_fondInactif
- QString m_fondActif
- QString m_fondRobot
- QString m_fondObjectif
- QString m_fondRate

9.3.1 Documentation des constructeurs et destructeur

9.3.1.1 CTable : CTable (QWidget *parent = 0) [explicit]

Références [CSS_FOND_INACTIF](#), [m_args](#), [m_font](#), [m_pFilet](#), [m_pGridLayout](#), [m_pOverlay](#), [m_pZones](#), [NB_ZONES](#), [resetSeance\(\)](#), et [WIDGET_SIZE_MAX](#).

```

                                :
QWidget (parent)
{
    //=====
    //          GRAPHIQUE
    //=====

    m_pGridLayout = new QGridLayout(this);

    //=====
    //          OVERLAY TABLE
    //=====
    m_pOverlay = new QLabel(this);
    m_pOverlay->setStyleSheet(QString::fromUtf8("QLabel\n"
        "{\n"
        "    border-image: url(/:images/resources/table.jpg) 0 0 0 0 stretch\n"
        "    stretch;\n"
        "}")");
    m_pOverlay->setMinimumSize(QSize(0, WIDGET_SIZE_MAX));
    m_pOverlay->setMaximumSize(QSize(WIDGET_SIZE_MAX, WIDGET_SIZE_MAX));
    m_pGridLayout->addWidget(m_pOverlay, 0, 0, 3, 3);

    //=====
    //          ZONES
    //=====
    QLabel* pZone;
    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        pZone = new QLabel(this);
        pZone->setText("Zone " + QString::number(i+1));
        pZone->setFont(m_font);
        pZone->setStyleSheet(CSS_FOND_INACTIF);
        pZone->setAlignment(Qt::AlignCenter);

        //-----
        m_pZones.push_back(pZone);
        m_pGridLayout->addWidget(pZone, (i/3), (i%3));
    }

    //=====
    //          OVERLAY TEXT
    //=====

    /* Affichage SUR la table

    m_pOverlayText = new QLabel(this);
    m_pOverlayText->setMinimumSize(QSize(0, WIDGET_SIZE_MAX));
    m_pOverlayText->setMaximumSize(QSize(WIDGET_SIZE_MAX, WIDGET_SIZE_MAX));
    m_pOverlayText->setText("");
    m_fontOverlay.setPointSize(12);
    m_pOverlayText->setFont(m_fontOverlay);
    m_pOverlayText->setAlignment(Qt::AlignCenter);
    m_pGridLayout->addWidget(m_pOverlayText, 0, 0, 3, 3);
    m_pOverlayText->setStyleSheet(QString::fromUtf8("QLabel\n"
        "{\n"
        "    background-color: rgb(0, 0, 0, 0);\n"
        "    color: rgba(0,0,0,0);\n"
        "}")");
    */

    //=====
    //          FILET

```

```

//=====
m_pFilet = new QLabel(this);
m_pFilet->setStyleSheet(QString::fromUtf8("QLabel\n"
"\n"
"    background: url(/images/resources/filet.jpg) cover;\n"
"}"));
m_pFilet->setMinimumSize(QSize(4, WIDGET_SIZE_MAX));
m_pFilet->setMaximumSize(QSize(WIDGET_SIZE_MAX, 50));
m_pGridLayout->addWidget(m_pFilet, 4, 0, 1, 0);

//=====
setLayout(m_pGridLayout);
m_pGridLayout->setSpacing(0);

//=====
// LOGIQUE
//=====
resetSeance();

//AUTRE
m_args.clear();
}

```

9.3.2 Documentation des fonctions membres

9.3.2.1 void CTable : :balleEnJeu ()

*

Références [getBalleCoteTable\(\)](#), [getBalleCoteTablePrec\(\)](#), [m_bBalleCoteTable](#), [m_bBalleCoteTablePrec](#), [m_iBallesHorsTable](#), et [m_iBallesTotal](#).

Référencé par [Clhm : :balleEnJeu\(\)](#).

```

{
    m_iBallesTotal++;
    //    rafraichirNbBallesZone();

    m_bBalleCoteTablePrec = m_bBalleCoteTable;
    m_bBalleCoteTable = false;

    if(getBalleCoteTablePrec() == false && getBalleCoteTable() == false)
    {
        qDebug() << Q_FUNC_INFO << " La Balle n'as pas ete renvoye";
        m_iBallesHorsTable++;
    }
}

```

9.3.2.2 void CTable : :finirSeance ()

*

Références [getBalleCoteTable\(\)](#), [getBalleCoteTablePrec\(\)](#), [m_bBalleCoteTable](#), [m_bBalleCoteTablePrec](#), et [m_iBallesHorsTable](#).

Référencé par [Clhm : :finirSeance\(\)](#).

```

{
    m_bBalleCoteTablePrec = m_bBalleCoteTable;
    m_bBalleCoteTable = false;

    if(getBalleCoteTablePrec() == false && getBalleCoteTable() == false)
    {
        qDebug() << Q_FUNC_INFO << " La derniere balle n'as jamais ete renvoyé"
        ;
        m_iBallesHorsTable++;
    }
}

```


9.3.2.3 bool CTable : :getBalleCoteTable ()

Références [m_bBalleCoteTable](#).

Référencé par [balleEnJeu\(\)](#), et [finirSeance\(\)](#).

```
{ return m_bBalleCoteTable; }
```

9.3.2.4 bool CTable : :getBalleCoteTablePrec ()

*

Références [m_bBalleCoteTablePrec](#).

Référencé par [balleEnJeu\(\)](#), et [finirSeance\(\)](#).

```
{ return m_bBalleCoteTablePrec; }
```

9.3.2.5 int CTable : :getBallesBonnes ()

Références [m_iBallesBonnes](#).

Référencé par [Clhm : :finirSeance\(\)](#).

```
{ return m_iBallesBonnes; }
```

9.3.2.6 int CTable : :getBallesEnchainees ()

*

Références [m_iBallesEnchainees](#).

Référencé par [Clhm : :finirSeance\(\)](#).

```
{ return m_iBallesEnchainees; }
```

9.3.2.7 int CTable : :getBallesHorsTable ()

*

Références [m_iBallesHorsTable](#).

Référencé par [Clhm : :rafraichirStats\(\)](#).

```
{ return m_iBallesHorsTable; }
```

9.3.2.8 int CTable : :getBallesMaximum ()

*

Références [m_iBallesMaximum](#).

Référencé par [Clhm : :ballesTotalSurBallesMaximum\(\)](#).

```
{ return m_iBallesMaximum; }
```

9.3.2.9 int CTable : :getBallesObjectif ()

*

Références [m_iBallesDansZone](#), et [m_iZoneObjectif](#).Référéncé par [Clhm : :finirSeance\(\)](#).

```
{ return m_iBallesDansZone[m_iZoneObjectif]; }
```

9.3.2.10 int CTable : :getBallesTotal ()

*

Références [m_iBallesTotal](#).Référéncé par [Clhm : :ballesTotalSurBallesMaximum\(\)](#), [Clhm : :finirSeance\(\)](#), et [Clhm : :rafraichirStats\(\)](#).

```
{ return m_iBallesTotal; }
```

9.3.2.11 uint8_t CTable : :getZoneObjectif ()

*

Paramètres

<i>nombre</i>	de balles en INT
---------------	------------------

Références [m_iZoneObjectif](#).Référéncé par [Clhm : :finirSeance\(\)](#).

```
{ return m_iZoneObjectif; }
```

9.3.2.12 int CTable : :getZoneToucheePrec ()

*

9.3.2.13 bool CTable : :impacterZone (uint8_t numeroZone)

*

Paramètres

<i>usuellement</i>	getRatioFenetreY() de Clhm
--------------------	--

Références [CSS_FOND_ACTIF](#), [CSS_FOND_RATE](#), [DELAI_COUP](#), [m_bBalleCoteTable](#), [m_bBalleCoteTablePrec](#), [m_fontBig](#), [m_iBallesBonnes](#), [m_iBallesDansZone](#), [m_iBallesEnchainees](#), [m_iBallesEnchaineesMax](#), [m_iBallesHorsTable](#), [m_iBallesTotal](#), [m_iZoneObjectif](#), [m_iZoneRobot](#), [m_iZoneTouchee](#), [m_iZoneToucheePrec](#), [m_pZones](#), [NB_ZONES](#), [rafraichirInactif\(\)](#), [rafraichirNbBallesZone\(\)](#), et [ZONE_AUCUNE](#).

Référéncé par [Clhm : :impacterZone\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    if (numeroZone >= NB_ZONES || numeroZone == m_iZoneRobot)
        return false;

    if (m_iBallesBonnes + m_iBallesHorsTable >= m_iBallesTotal)
        return false;

    m_bBalleCoteTablePrec = m_bBalleCoteTable;
}
```

```

m_bBalleCoteTable = true;

qDebug() << Q_FUNC_INFO << numeroZone;
m_iZoneToucheePrec = m_iZoneTouchee;
m_iZoneTouchee = numeroZone;

m_iBallesBonnes++;
m_iBallesDansZone[numeroZone]++;

rafraichirNbBallesZone();

if (numeroZone != ZONE_AUCUNE)
{
    if (numeroZone == m_iZoneObjectif || m_iZoneObjectif == ZONE_AUCUNE)
    {
        m_pZones[numeroZone]->setStyleSheet(CSS_FOND_ACTIF);
        m_iBallesEnchainees++;
    }
    else
    {
        m_pZones[numeroZone]->setStyleSheet(CSS_FOND_RATE);
        m_iBallesEnchainees = 0;
    }

    m_pZones[numeroZone]->setFont(m_fontBig);
}
else
    m_iBallesEnchainees = 0;

if (m_iBallesEnchainees > m_iBallesEnchaineesMax)
    m_iBallesEnchaineesMax = m_iBallesEnchainees;

m_iZoneTouchee = numeroZone;
QTimer::singleShot(DELAI_COUP, this, SLOT(rafraichirInactif()));

return true;
}

```

9.3.2.14 void CTable : :rafraichirCSS (float ratio)

*

Paramètres

<i>usuellement</i>	getRatioFenetreY() de Clhm
--------------------	----------------------------

Références [CSS_FOND_INACTIF](#), [m_font](#), [m_fontBig](#), [m_fontOverlay](#), [m_pZones](#), [NB_ZONES](#), [TAILLE_OVERLAY](#), [TAILLE_TEXTE](#), [TAILLE_TEXTE_NB](#), et [TAILLE_TEXTE_NB_BIG](#).

Référencé par [Clhm : :rafraichirCSS\(\)](#).

```

{
    m_font.setPointSize((int) (TAILLE_TEXTE_NB*ratio));
    m_font.setBold(true);
    m_fontBig.setPointSize((int) (TAILLE_TEXTE_NB_BIG*ratio));
    m_fontBig.setBold(true);
    m_fontOverlay.setPointSize((int) (TAILLE_OVERLAY*ratio));
    qDebug() << Q_FUNC_INFO << (TAILLE_TEXTE*ratio);

    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        m_pZones[i]->setFont(m_font);
        m_pZones[i]->setStyleSheet(CSS_FOND_INACTIF);
    }
    //m_pOverlayText->setFont(m_fontOverlay);
}

```

9.3.2.15 void CTable : :rafraichirInactif () [slot]

Références [CSS_FOND_INACTIF](#), [CSS_FOND_OBJECTIF](#), [m_font](#), [m_iZoneObjectif](#), [m_iZoneRobot](#), [m_pZones](#), et [NB_ZONES](#).

Référencé par [impacterZone\(\)](#).

```

{

```

```

qDebug() << Q_FUNC_INFO;
for(uint8_t i=0; i < NB_ZONES; i++)
{
    if (i != m_iZoneRobot)
    {
        if (i == m_iZoneObjectif)
            m_pZones[i]->setStyleSheet(CSS_FOND_OBJECTIF);
        else
            m_pZones[i]->setStyleSheet(CSS_FOND_INACTIF);
    }
    m_pZones[i]->setFont(m_font);
}
}

```

9.3.2.16 void CTable : :rafraichirNbBallesZone () [private]

Références [m_iBallesDansZone](#), [m_iBallesTotal](#), [m_iZoneRobot](#), [m_pZones](#), et [NB_ZONES](#).

Référencé par [impacterZone\(\)](#), [resetNbBallesZone\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        if (i != m_iZoneRobot)
        {
            if (m_iBallesTotal)
                m_pZones[i]->setText(QString::number(((double(m_iBallesDansZone
[i])/double(m_iBallesTotal)*100)), 'f', 0) + '%');
            else
                m_pZones[i]->setText(QString::number(0) + '%');
        }
    }
}

```

9.3.2.17 void CTable : :resetNbBallesZone ()

*

Références [m_iBallesDansZone](#), [NB_ZONES](#), et [rafraichirNbBallesZone\(\)](#).

Référencé par [resetSeance\(\)](#), et [resetStatistiques\(\)](#).

```

{
    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        m_iBallesDansZone[i] = 0;
        // m_pZones[i]->setText(QString::number(0) + '%');
        // m_pZones[i]->setStyleSheet(CSS_FOND_INACTIF);
    }
    rafraichirNbBallesZone();
}

```

9.3.2.18 void CTable : :resetSeance ()

*

Paramètres

<i>enum</i>	zones_e, voir const.h
-------------	---------------------------------------

Références [DEV_BALLES_MAX](#), [m_args](#), [m_iBallesEnchainees](#), [m_iBallesEnchaineesMax](#), [m_iBallesMaximum](#), [m_iZoneObjectif](#), [m_iZoneRobot](#), [resetNbBallesZone\(\)](#), [resetStatistiques\(\)](#), [setBallesMaximum\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [CTable\(\)](#), et [CIhm : :resetSeance\(\)](#).

```

{
    m_iBallesMaximum = 0;
}

```

```

    m_iBallesEnchainees      = 0;
    m_iBallesEnchaineesMax   = 0;
    m_iZoneRobot = m_iZoneObjectif = ZONE_AUCUNE;
    resetNbBallesZone();

    resetStatistiques();

    if (m_args.contains("-dev"))
        setBallesMaximum(DEV_BALLES_MAX);
}

```

9.3.2.19 void CTable :resetStatistiques ()

*

Références [m_bBalleCoteTable](#), [m_bBalleCoteTablePrec](#), [m_iBallesBonnes](#), [m_iBallesEnchainees](#), [m_iBallesHorsTable](#), [m_iBallesTotal](#), [m_iZoneTouchee](#), [m_iZoneToucheePrec](#), [resetNbBallesZone\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [Clhm :commencerSeance\(\)](#), et [resetSeance\(\)](#).

```

{
    m_iZoneToucheePrec = m_iZoneTouchee = ZONE_AUCUNE;
    m_iBallesBonnes      = 0;
    m_iBallesTotal       = 0;
    m_iBallesEnchainees  = 0;
    m_iBallesHorsTable   = 0;
    m_bBalleCoteTable     = true;
    m_bBalleCoteTablePrec = true;
    resetNbBallesZone();
}

```

9.3.2.20 void CTable :setBallesMaximum (int nb)

*

Références [m_iBallesMaximum](#).

Référencé par [Clhm :gererArguments\(\)](#), [resetSeance\(\)](#), et [Clhm :setBallesMaximum\(\)](#).

```

{ m_iBallesMaximum = nb; }

```

9.3.2.21 void CTable :setFiletTaille (float ratio)

Références [HAUTEUR_FILET](#), [m_pFilet](#), et [WIDGET_SIZE_MAX](#).

Référencé par [Clhm :initialisationFenetre\(\)](#).

```

{
    m_pFilet->setMinimumSize(QSize(4, (float)HAUTEUR_FILET*ratio));
    m_pFilet->setMaximumSize(QSize(WIDGET_SIZE_MAX, (float)HAUTEUR_FILET*ratio));
    qDebug() << Q_FUNC_INFO << ((float)HAUTEUR_FILET*ratio);
}

```

9.3.2.22 void CTable :setLayerEcran (uint8_t layer, float tailleFenetreY)

Références [LAYER_RECAP](#), [m_font](#), [m_pZones](#), [NB_ZONES](#), et [TAILLE_TEXTE_NB](#).

Référencé par [Clhm :setLayerEcran\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO << " layer: " << layer << "tailleFenetreY: " <<
        tailleFenetreY;
    int tailleTexte;

    if (layer == LAYER_RECAP)

```

```

        tailleTexte = TAILLE_TEXTE_NB*(tailleFenetreY/1.75);
    else
        tailleTexte = TAILLE_TEXTE_NB*(tailleFenetreY);

    m_font.setPointSize((int)tailleTexte);

    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        m_pZones[i]->setFont(m_font);
    }
}

```

9.3.2.23 void CTable : :setZoneObjectif (uint8_t zone)

*

Paramètres

<i>enum</i>	zones_e, voir const.h
-------------	---------------------------------------

Références [CSS_FOND_INACTIF](#), [CSS_FOND_OBJECTIF](#), [m_iZoneObjectif](#), [m_iZoneTouchee](#), [m_iZoneToucheePrec](#), [m_pZones](#), [NB_ZONES](#), [rafraichirNbBallesZone\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [CIhm : :setZoneObjectif\(\)](#), et [CIhm : :setZoneObjectifRandom\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    if ( (numeroZone >= NB_ZONES && numeroZone != ZONE_AUCUNE) )
    {
        qDebug() << "!!\\ Erreur Zone /!\\";
        return;
    }
    if (m_iZoneObjectif < NB_ZONES)
        m_pZones[m_iZoneObjectif]->setStyleSheet(CSS_FOND_INACTIF);

    m_iZoneObjectif = numeroZone;

    if (m_iZoneTouchee == numeroZone)
        m_iZoneTouchee = ZONE_AUCUNE;
    if (m_iZoneToucheePrec == numeroZone)
        m_iZoneToucheePrec = ZONE_AUCUNE;

    if (m_iZoneObjectif < NB_ZONES)
    {
        m_pZones[numeroZone]->setStyleSheet(CSS_FOND_OBJECTIF);
    }

    rafraichirNbBallesZone();
}

```

9.3.2.24 void CTable : :setZoneRobot (uint8_t zone)

*

Paramètres

<i>enum</i>	zones_e, voir const.h
-------------	---------------------------------------

Références [CSS_FOND_INACTIF](#), [CSS_FOND_ROBOT](#), [m_iZoneRobot](#), [m_iZoneTouchee](#), [m_iZoneToucheePrec](#), [m_pZones](#), [NB_ZONES](#), [rafraichirNbBallesZone\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [CIhm : :setZoneRobot\(\)](#), et [CIhm : :setZoneRobotRandom\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    if ( (numeroZone >= NB_ZONES && numeroZone != ZONE_AUCUNE) )
    {
        qDebug() << "!!\\ Erreur Zone /!\\";
        return;
    }

    if (m_iZoneRobot < NB_ZONES)

```

```

        m_pZones[m_iZoneRobot]->setStyleSheet(CSS_FOND_INACTIF);

    m_iZoneRobot = numeroZone;

    if (m_iZoneTouchee == numeroZone)
        m_iZoneTouchee = ZONE_AUCUNE;
    if (m_iZoneToucheePrec == numeroZone)
        m_iZoneToucheePrec = ZONE_AUCUNE;

    if (m_iZoneRobot < NB_ZONES)
    {
        m_pZones[numeroZone]->setStyleSheet(CSS_FOND_ROBOT);
        m_pZones[numeroZone]->setText("ROBOT");
    }

    rafraichirNbBallesZone();
}

```

9.3.3 Documentation des données membres

9.3.3.1 QStringList CTable : :m_args

Référencé par [CTable\(\)](#), [Clhm : :gererArguments\(\)](#), et [resetSeance\(\)](#).

9.3.3.2 bool CTable : :m_bBalleCoteTable [private]

Référencé par [balleEnJeu\(\)](#), [finirSeance\(\)](#), [getBalleCoteTable\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

9.3.3.3 bool CTable : :m_bBalleCoteTablePrec [private]

Référencé par [balleEnJeu\(\)](#), [finirSeance\(\)](#), [getBalleCoteTablePrec\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

9.3.3.4 QString CTable : :m_fondActif [private]

9.3.3.5 QString CTable : :m_fondInactif [private]

9.3.3.6 QString CTable : :m_fondObjectif [private]

9.3.3.7 QString CTable : :m_fondRate [private]

9.3.3.8 QString CTable : :m_fondRobot [private]

9.3.3.9 QFont CTable : :m_font [private]

Référencé par [CTable\(\)](#), [rafraichirCSS\(\)](#), [rafraichirInactif\(\)](#), et [setLayerEcran\(\)](#).

9.3.3.10 QFont CTable : :m_fontBig [private]

Référencé par [impacterZone\(\)](#), et [rafraichirCSS\(\)](#).

9.3.3.11 QFont CTable : :m_fontOverlay [private]

Référencé par [rafraichirCSS\(\)](#).

9.3.3.12 int CTable : :m_iBallesBonnes [private]

Référencé par [getBallesBonnes\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

9.3.3.13 int CTable : :m_iBallesDansZone[NB_ZONES] [private]

Référencé par [getBallesObjectif\(\)](#), [impacterZone\(\)](#), [rafraichirNbBallesZone\(\)](#), et [resetNbBallesZone\(\)](#).

9.3.3.14 `int CTable : :m_iBallesEnchainees` [private]

Référencé par [getBallesEnchainees\(\)](#), [impacterZone\(\)](#), [resetSeance\(\)](#), et [resetStatistiques\(\)](#).

9.3.3.15 `int CTable : :m_iBallesEnchaineesMax` [private]

Référencé par [impacterZone\(\)](#), et [resetSeance\(\)](#).

9.3.3.16 `int CTable : :m_iBallesHorsTable` [private]

Référencé par [balleEnJeu\(\)](#), [finirSeance\(\)](#), [getBallesHorsTable\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

9.3.3.17 `int CTable : :m_iBallesMaximum` [private]

Référencé par [getBallesMaximum\(\)](#), [resetSeance\(\)](#), et [setBallesMaximum\(\)](#).

9.3.3.18 `int CTable : :m_iBallesTotal` [private]

Référencé par [balleEnJeu\(\)](#), [getBallesTotal\(\)](#), [impacterZone\(\)](#), [rafraichirNbBallesZone\(\)](#), et [resetStatistiques\(\)](#).

9.3.3.19 `int CTable : :m_iZoneObjectif` [private]

Référencé par [getBallesObjectif\(\)](#), [getZoneObjectif\(\)](#), [impacterZone\(\)](#), [rafraichirInactif\(\)](#), [resetSeance\(\)](#), et [setZoneObjectif\(\)](#).

9.3.3.20 `int CTable : :m_iZoneRobot` [private]

Référencé par [impacterZone\(\)](#), [rafraichirInactif\(\)](#), [rafraichirNbBallesZone\(\)](#), [resetSeance\(\)](#), et [setZoneRobot\(\)](#).

9.3.3.21 `int CTable : :m_iZoneTouchee` [private]

Référencé par [impacterZone\(\)](#), [resetStatistiques\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

9.3.3.22 `int CTable : :m_iZoneToucheePrec` [private]

Référencé par [impacterZone\(\)](#), [resetStatistiques\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

9.3.3.23 `QLabel* CTable : :m_pFilet` [private]

Référencé par [CTable\(\)](#), et [setFiletTaille\(\)](#).

9.3.3.24 `QGridLayout* CTable : :m_pGridLayout` [private]

*

Référencé par [CTable\(\)](#).

9.3.3.25 `QLabel* CTable : :m_pOverlay` [private]

Référencé par [CTable\(\)](#).

9.3.3.26 `QLabel* CTable : :m_pOverlayText`

9.3.3.27 `QVector<QLabel*> CTable : :m_pZones` [private]

Référencé par [CTable\(\)](#), [impacterZone\(\)](#), [rafraichirCSS\(\)](#), [rafraichirInactif\(\)](#), [rafraichirNbBallesZone\(\)](#), [setLayerEcran\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

- [table.h](#)
- [table.cpp](#)

9.4 Référence de la classe CTFrame

```
#include <trame.h>
```

Connecteurs publics

- bool [traiterTrame](#) (QString donneesRecues)
voir [quitter\(\)](#) de [Clhm](#)
- bool [gererTrame](#) (QString donneesRecues)
découpe les trames de la reception

Signaux

- void [setInfoConnect](#) (QString nom)
- void [setLayerEcran](#) (uint8_t layer)
voir [setInfoConnect\(QString nom\)](#) de [Clhm](#)
- void [commencerSeance](#) ()
voir [setLayerEcran\(int layer\)](#) de [Clhm](#)
- void [pauserSeance](#) ()
voir [commencerSeance\(\)](#) de [Clhm](#)
- void [reprendreSeance](#) ()
voir [pauserSeance\(\)](#) de [Clhm](#)
- void [finirSeance](#) ()
voir [reprendreSeance\(\)](#) de [Clhm](#)
- void [resetSeance](#) ()
voir [finirSeance\(\)](#) de [Clhm](#)
- void [impacterZone](#) (uint8_t zone)
voir [resetSeance\(\)](#) de [Clhm](#)
- void [balleEnJeu](#) ()
voir [impacterZone\(int zone\)](#) de [Clhm](#)
- void [setZoneRobot](#) (uint8_t zone)
voir [balleEnJeu\(\)](#) de [Clhm](#)
- void [setZoneObjectif](#) (uint8_t zone)
voir [setZoneRobot\(int zone\)](#) de [Clhm](#)
- void [setBallesMaximum](#) (int balles)
voir [setZoneObjectif\(int zone\)](#) de [Clhm](#)
- void [setFrequenceRobot](#) (float freq)
voir [setBalleMaximum\(int balles\)](#) de [Clhm](#)
- void [rafraichirCSS](#) ()
voir [setFrequenceRobot\(int freq\)](#) de [CTable](#)
- void [quitter](#) ()
voir [rafraichirCSS\(\)](#) de [Clhm](#)

Fonctions membres publiques

- [CTFrame](#) (QObject *parent=0)
- [~CTFrame](#) ()

Fonctions membres privées

- QString [extraireElement](#) (QString donneesRecues, const int element)
extrait les elements de la trame avec [extraireElement\(QString donneesRecues\)](#) puis effectue les signals en fonction
- int [getTrameLength](#) (QString donneesRecues)
Découpe la trame et retourne l'élément.
- void [messageNonReconnu](#) (QString donneesRecues, int element)

- *Retourne la longueur de la trame.*
– bool [gererTramesSansParametre](#) (QString donneesRecues)
– *Affiche les chars dans l'element demandé par rapport a la liste d'elements de la trame.*
- bool [gererTrames1Parametre](#) (QString donneesRecues)
– *verifie, identifie et execute les trames ne possédant pas de parametre*

9.4.1 Documentation des constructeurs et destructeur

9.4.1.1 CTrame : :CTrame (QObject * parent = 0)

```

        : QObject (parent)
    {
    }

```

9.4.1.2 CTrame : :~CTrame ()

```

{
}

```

9.4.2 Documentation des fonctions membres

9.4.2.1 void CTrame : :balleEnJeu () [signal]

*

Référencé par [gererTrames1Parametre\(\)](#), et [gererTramesSansParametre\(\)](#).

9.4.2.2 void CTrame : :commencerSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

9.4.2.3 QString CTrame : :extraireElement (QString donneesRecues, const int element) [private]

*

Paramètres

<i>Trame</i>	en QString
--------------	------------

Renvoie

validité de la trame

Référencé par [gererTrame\(\)](#), [gererTrames1Parametre\(\)](#), [gererTramesSansParametre\(\)](#), et [messageNon-Reconnu\(\)](#).

```

{
    if(donneesRecues.isEmpty())
        return QString();

    QStringList listeElements;
    listeElements = donneesRecues.split(":");

    return listeElements.at(iElement).trimmed();
}

```

9.4.2.4 void CTrame : :finirSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

9.4.2.5 bool CTFrame : :gererTrame (QString *donneesRecues*) [slot]

*

Références [extraireElement\(\)](#), [gererTrames1Parametre\(\)](#), [gererTramesSansParametre\(\)](#), [getTrameLength\(\)](#), [setBallesMaximum\(\)](#), [setZoneObjectif\(\)](#), [setZoneRobot\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [traiterTrame\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
        QString::fromUtf8("Données reçues : ") << donneesRecues;

    bool trameValide = true;

    if (getTrameLength(donneesRecues) == 2) // sans arguments
        trameValide = gererTramesSansParametre(donneesRecues);

    else if (getTrameLength(donneesRecues) == 3) // 1 arguments
        trameValide = gererTrames1Parametre(donneesRecues);

    // $TPA:SETSEANCE:[POS_ROBOT]:[POS_OBJECTIF]:[NB_BALLES_MAX]:[FREQ_ENVOI]
    else if (getTrameLength(donneesRecues) == 5 && extraireElement(
        donneesRecues,1).startsWith("SETSEANCE"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0) // AUCUN dans
            le cas d'un negatif ou zero
            emit setZoneRobot(ZONE_AUCUNE);
        else
            emit setZoneRobot((extraireElement(donneesRecues,2)).toInt() - 1);

        if ((extraireElement(donneesRecues,3)).toInt() <= 0) // AUCUN dans
            le cas d'un negatif ou zero
            emit setZoneObjectif(ZONE_AUCUNE);
        else
            emit setZoneObjectif((extraireElement(donneesRecues,3)).toInt() - 1);
    };

    emit setBallesMaximum((extraireElement(donneesRecues,4)).toInt());
}
else
{
    qDebug() << Q_FUNC_INFO << "!!\\ TRAME NON RECONNUE !!\\";
    trameValide = false;
}
return trameValide;
}
```

9.4.2.6 bool CTFrame : :gererTrames1Parametre (QString *donneesRecues*) [private]

*

Références [balleEnJeu\(\)](#), [extraireElement\(\)](#), [impacterZone\(\)](#), [messageNonReconnu\(\)](#), [setBallesMaximum\(\)](#), [setInfoConnect\(\)](#), [setZoneObjectif\(\)](#), [setZoneRobot\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [gererTrame\(\)](#).

```
{
    bool trameValide = true;

    // $TPA:CONNECT:[nom]
    if (extraireElement(donneesRecues,1).startsWith("CONNECT"))
    {
        emit setInfoConnect(extraireElement(donneesRecues,2));
    }
    // $TPA:SETROBOT:[POS_ROBOT]
    else if (extraireElement(donneesRecues,1).startsWith("SETROBOT"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0) // Adaptation
            au systeme de la table
            emit setZoneRobot(ZONE_AUCUNE);
        else
            emit setZoneRobot((extraireElement(donneesRecues,2)).toInt() - 1);
    }
    // $TPA:SETOBJECTIF:[POS_OBJECTIF]
    else if (extraireElement(donneesRecues,1).startsWith("SETOBJECTIF"))
    {

```

```

        if ((extraireElement(donneesRecues,2)).toInt() <= 0)    // Adaptation
        au systeme de la table
            emit setZoneObjectif(ZONE_AUCUNE);
        else
            emit setZoneObjectif((extraireElement(donneesRecues,2)).toInt() - 1
        );
    }
    // $TPA:SETBALLESMAX:[BALLE]
    else if (extraireElement(donneesRecues,1).startsWith("SETBALLESMAX"))
    {
        emit setBallesMaximum((extraireElement(donneesRecues,2)).toInt());
    }
    // $TPA:IMPACT:[X]
    else if (extraireElement(donneesRecues,1).startsWith("IMPACT"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0)    //
        Adaptation au systeme de la table
            emit balleEnJeu();
        else
            emit impacterZone((extraireElement(donneesRecues,2)).toInt() - 1);
    }
    else
    {
        messageNonReconnu(donneesRecues,1);
        trameValide = false;
    }
    return trameValide;
}

```

9.4.2.7 bool CTrame : :gererTramesSansParametre (QString donneesRecues) [private]

*

Paramètres

Trame	en QString, element en int
-------	----------------------------

Références [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [extraireElement\(\)](#), [finirSeance\(\)](#), [messageNonReconnu\(\)](#), [pauserSeance\(\)](#), [quitter\(\)](#), [reprendreSeance\(\)](#), et [resetSeance\(\)](#).

Référencé par [gererTrame\(\)](#).

```

{
    bool trameValide = true;
    // $TPA:JOUER
    if (extraireElement(donneesRecues,1).startsWith("JOUER"))
    {
        emit balleEnJeu();
    }
    // $TPA:START
    else if (extraireElement(donneesRecues,1).startsWith("START"))
    {
        emit commencerSeance();
    }
    // $TPA:FINSEANCE
    else if (extraireElement(donneesRecues,1).startsWith("FINSEANCE"))
    {
        emit finirSeance();
    }
    // $TPA:PAUSE
    else if (extraireElement(donneesRecues,1).startsWith("PAUSE"))
    {
        emit pauserSeance();
    }
    // $TPA:RESUME
    else if (extraireElement(donneesRecues,1).startsWith("RESUME"))
    {
        emit reprendreSeance();
    }
    // $TPA:RESET
    else if (extraireElement(donneesRecues,1).startsWith("RESET"))
    {
        emit resetSeance();
    }
    // $TPA:QUIT
    else if (extraireElement(donneesRecues,1).startsWith("QUIT"))
    {
        emit quitter();
    }
}

```

```

    }
    else
    {
        messageNonReconnu(donneesRecues,1);
        trameValide = false;
    }
    return trameValide;
}

```

9.4.2.8 int CTrame : :getTrameLength (QString *donneesRecues*) [private]

*

Paramètres

<i>Trame</i>	en QString, index de l'élément
--------------	--------------------------------

Référencé par [gererTrame\(\)](#).

```

{
    if(donneesRecues.isEmpty())
        return 0;

    QStringList listeElements;
    listeElements = donneesRecues.split(":");

    return listeElements.length();
}

```

9.4.2.9 void CTrame : :impacterZone (uint8_t *zone*) [signal]

*

Référencé par [gererTrames1Parametre\(\)](#).

9.4.2.10 void CTrame : :messageNonReconnu (QString *donneesRecues*, int *element*) [private]

*

Paramètres

<i>Trame</i>	en QString
--------------	------------

Références [extraireElement\(\)](#).

Référencé par [gererTrames1Parametre\(\)](#), et [gererTramesSansParametre\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO << "!!\\ TRAME NON RECONNUE /!\\";

    QString testChars = " [Element " + QString::number(element) + "]: [|";

    for(int i=0; i<extraireElement(donneesRecues,element).length();i++)
    {
        testChars += (extraireElement(donneesRecues,element)[i]) + "|";
    }
    testChars+="]";
    qDebug() << testChars;
}

```

9.4.2.11 void CTrame : :pauserSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

9.4.2.12 void CTFrame : :quitter () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

9.4.2.13 void CTFrame : :rafraichirCSS () [signal]

*

9.4.2.14 void CTFrame : :reprendreSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

9.4.2.15 void CTFrame : :resetSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

9.4.2.16 void CTFrame : :setBallesMaximum (int *balles*) [signal]

*

Référencé par [gererTrame\(\)](#), et [gererTrames1Parametre\(\)](#).

9.4.2.17 void CTFrame : :setFrequenceRobot (float *freq*) [signal]

*

9.4.2.18 void CTFrame : :setInfoConnect (QString *nom*) [signal]

Référencé par [gererTrames1Parametre\(\)](#).

9.4.2.19 void CTFrame : :setLayerEcran (uint8_t *layer*) [signal]

*

9.4.2.20 void CTFrame : :setZoneObjectif (uint8_t *zone*) [signal]

*

Référencé par [gererTrame\(\)](#), et [gererTrames1Parametre\(\)](#).

9.4.2.21 void CTFrame : :setZoneRobot (uint8_t *zone*) [signal]

*

Référencé par [gererTrame\(\)](#), et [gererTrames1Parametre\(\)](#).

9.4.2.22 bool CTFrame : :traiterTrame (QString *donneesRecues*) [slot]

*

Références [gererTrame\(\)](#).

Référencé par [CIhm : :envoyerCommande\(\)](#).

```
{
    if(donneesRecues.isEmpty())
        return false;

    bool retour = false;
```

```
QStringList listeElements;  
listeElements = donneesRecues.split("$",QString::SkipEmptyParts);  
  
for(uint8_t i = 0; i < listeElements.length(); i++)  
{  
    if (listeElements.at(i).startsWith("TTPA:"))  
        retour = gererTrame("$" + listeElements.at(i));  
}  
return retour;  
}
```

La documentation de cette classe a été générée à partir des fichiers suivants :

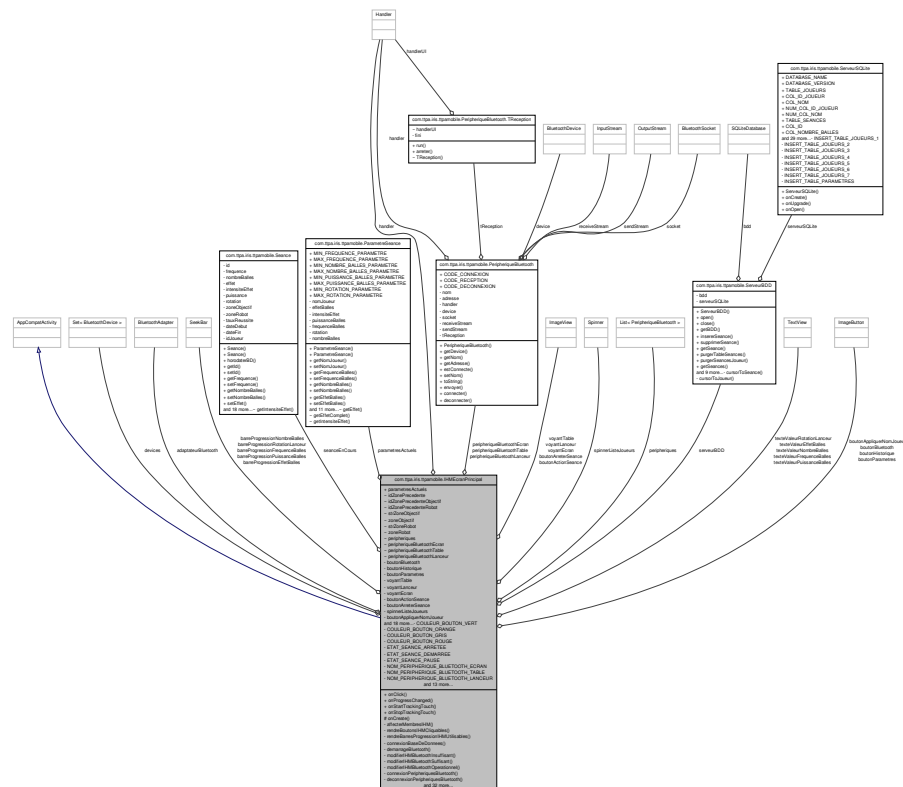
- [trame.h](#)
- [trame.cpp](#)

9.5 Référence de la classe com.tpa.iris.ttpamobile.IHMEcranPrincipal

Graphe d'héritage de com.tpa.iris.ttpamobile.IHMEcranPrincipal :



Graphe de collaboration de com.ttpa.iris.ttpamobile.IHMEcranPrincipal :



Fonctions membres publiques

- void [onClick](#) (View element)
- void [onProgressChanged](#) (SeekBar seekBar, int progress, boolean fromUser)
- void [onStartTrackingTouch](#) (SeekBar seekBar)
- void [onStopTrackingTouch](#) (SeekBar seekBar)

Attributs publics

- ParametreSeance parametresActuels = new ParametreSeance()

Fonctions membres protégées

- void onCreate (Bundle savedInstanceState)

Attributs de packaging

- ```

- int idZonePrecedente = -1
- int idZonePrecedenteObjectif = -1
- int idZonePrecedenteRobot = -1
- String strZoneObjectif = new String("ZONE 0")
- int zoneObjectif = 0
- String strZoneRobot = new String("ZONE 0")
- int zoneRobot = 0
- List< PeripheriqueBluetooth > peripheriques
- PeripheriqueBluetooth peripheriqueBluetoothEcran = null
- PeripheriqueBluetooth peripheriqueBluetoothTable = null
- PeripheriqueBluetooth peripheriqueBluetoothLanceur = null

```

## Fonctions membres privées

- void [affecterMembresIHM](#) ()
- void [rendreBoutonsIHMClickables](#) ()
- void [rendreBarresProgressionIHMUtilisables](#) ()
- void [connexionBaseDeDonnees](#) ()
- void [demarrageBluetooth](#) ()
- void [modifierIHMBluetoothInsuffisant](#) ()
- void [modifierIHMBluetoothSuffisant](#) ()
- void [modifierIHMBluetoothOperationnel](#) ()
- void [connexionPeripheriquesBluetooth](#) ()
- void [deconnexionPeripheriquesBluetooth](#) ()
- void [connexionPeripheriqueBluetoothEcran](#) (BluetoothDevice appareilBluetooth)
- void [connexionPeripheriqueBluetoothTable](#) (BluetoothDevice appareilBluetooth)
- void [connexionPeripheriqueBluetoothLanceur](#) (BluetoothDevice appareilBluetooth)
- boolean [verifierConnexionAppareilsBluetoothRequis](#) ()
- void [actualiserIHMAppareilsBluetooth](#) (boolean tableEstConnectee, boolean lanceurEstConnecte, boolean ecranEstConnecte)
- void [actionnerSeance](#) ()
- void [demarrerSeance](#) ()
- void [pauserSeance](#) ()
- void [repandreSeance](#) ()
- void [arreterSeance](#) (boolean seanceAEnregistrer)
- void [appliquerParametresSeance](#) ()
- void [modifierValeursParametresIHM](#) ()
- void [attendre](#) (int tempsMillisecondes)
- void [envoyerTramePeripheriqueBluetoothEcran](#) (String trame)
- void [envoyerTrameConnexionPeripheriqueBluetoothEcran](#) ()
- void [envoyerTrameParametrageSeancePeripheriqueBluetoothEcran](#) ()
- void [envoyerTrameDebutSeancePeripheriqueBluetoothEcran](#) ()
- void [envoyerTrameArretPeripheriqueBluetoothEcran](#) ()
- void [ajouterJoueur](#) ()
- void [creerListeJoueurs](#) ()
- void [traiterDonneesRecues](#) (String nomAppareilSource, String donnees)
- void [traiterDonneesRecuesLanceur](#) (String donnees)
- void [traiterErreurRecueLanceur](#) (String erreur)
- void [traiterDonneesRecuesTable](#) (String donnees)
- void [calculerReussiteSeance](#) (int zoneTouchee)
- void [incrementerBallesJouees](#) ()
- void [envoyerTrameArretPeripheriqueBluetoothLanceur](#) ()
- void [envoyerTrameArretPeripheriqueBluetoothTable](#) ()
- void [envoyerTrameDebutSeancePeripheriqueBluetoothLanceur](#) ()
- void [envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur](#) ()
- void [envoyerTramePeripheriqueBluetoothLanceur](#) (String trame)
- void [selectionnerZone](#) (final int typeSelection, int choixObjectif, int choixRobot)
- void [redirectionActiviteHistorique](#) ()

## Attributs privés

- ImageButton [boutonBluetooth](#)
- ImageButton [boutonHistorique](#)
- ImageButton [boutonParametres](#)
- ImageView [voyantTable](#)
- ImageView [voyantLanceur](#)
- ImageView [voyantEcran](#)
- ImageView [boutonActionSeance](#)
- ImageView [boutonArreterSeance](#)
- Spinner [spinnerListeJoueurs](#)
- ImageButton [boutonAppliquerNomJoueur](#)
- SeekBar [barreProgressionNombreBalles](#)
- TextView [texteValeurNombreBalles](#)
- SeekBar [barreProgressionFrequenceBalles](#)
- TextView [texteValeurFrequenceBalles](#)
- SeekBar [barreProgressionEffetBalles](#)
- TextView [texteValeurEffetBalles](#)
- SeekBar [barreProgressionPuissanceBalles](#)
- TextView [texteValeurPuissanceBalles](#)
- SeekBar [barreProgressionRotationLanceur](#)
- TextView [texteValeurRotationLanceur](#)
- ServeurBDD [serveurBDD](#)
- int [etatSeance](#) = ETAT\_SEANCE\_ARRETEE
- Seance [seanceEnCours](#)

- int `ballesJouees`
- int `ballesReussies`
- String `nomJoueur`
- BluetoothAdapter `adaptateurBluetooth` = null
- Set< BluetoothDevice > `devices`
- final Handler `handler`

#### Attributs privés statiques

- static final int `COULEUR_BOUTON_VERT` = Color.parseColor("#5eed7b")
- static final int `COULEUR_BOUTON_ORANGE` = Color.parseColor("#f7bb31")
- static final int `COULEUR_BOUTON_GRIIS` = Color.parseColor("#c0c5c6")
- static final int `COULEUR_BOUTON_ROUGE` = Color.parseColor("#ee5e5e")
- static final int `ETAT_SEANCE_ARRETEE` = 0
- static final int `ETAT_SEANCE_DEMARREE` = 1
- static final int `ETAT_SEANCE_PAUSE` = 2
- static final String `NOM_PERIPHERIQUE_BLUETOOTH_ECRAN` = "TTPA-Ecran"
- static final String `NOM_PERIPHERIQUE_BLUETOOTH_TABLE` = "TTPA-Table"
- static final String `NOM_PERIPHERIQUE_BLUETOOTH_LANCEUR` = "TTPA-Lanceur"
- static final String `TRAME_ENTETE` = "\$TTPA"
- static final String `TRAME_FIN` = "\r\n"
- static final String `TRAME_ECRAN_DEBUT_SEANCE` = ":START"
- static final String `TRAME_ECRAN_PAUSE_SEANCE` = ":PAUSE"
- static final String `TRAME_ECRAN_REPRISE_SEANCE` = ":RESUME"
- static final String `TRAME_ECRAN_FIN_SEANCE` = ":FINSEANCE"
- static final String `TRAME_LANCEUR_PAUSE_SEANCE` = ":PAUSE:" + TRAME\_FIN
- static final String `TRAME_LANCEUR_REPRISE_SEANCE` = ":RESUME:" + TRAME\_FIN
- static final String `TRAME_LANCEUR_ARRET_SEANCE` = ":STOP:" + TRAME\_FIN
- static final String `TRAME_LANCEUR_PING` = ":PING:" + TRAME\_FIN
- static final String `TRAME_TABLE_ARRET_SEANCE` = ":RESET"
- static final int `REQUEST_CODE_ENABLE_BLUETOOTH` = 0
- static final int `SELECTION_ZONE_OBJECTIF` = 0
- static final int `SELECTION_ZONE_ROBOT` = 1

#### 9.5.1 Description détaillée

Created by smaniotto on 05/04/18. Classe `IHMEcranPrincipal` définissant le comportement du layout 'ecran\_principal'.

#### 9.5.2 Documentation des fonctions membres

##### 9.5.2.1 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance ( )` [private]

Méthode `actionnerSeance()` permettant de démarrer ou d'arrêter une séance en fonction de l'état actuel.

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_ARRETEE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_DEMARREE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_PAUSE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.idZonePrecedente`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauseSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.SELECTION_ZONE_ROBOT`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick()`.

```
{
 Log.d("IHMEcranPrincipal", "actionnerSeance() état : " + etatSeance);

 // Démarrer ou arrêter la séance selon son état actuel
 switch (etatSeance)
 {
 case ETAT_SEANCE_ARRETEE:
 // La séance n'est pas commencée, il faut la démarrer
 if(idZonePrecedente == -1) // Si les zones n'ont pas été
 sélectionnées
 selectionnerZone(SELECTION_ZONE_ROBOT, 0, 0);
 }
}
```

```

 else // Sinon, on peut démarrer la séance
 {
 demarrerSeance();
 etatSeance = ETAT_SEANCE_DEMARREE;
 }
 break;
 case ETAT_SEANCE_DEMARREE:
 // La séance est en cours, il faut la metre en pause
 pauserSeance();
 etatSeance = ETAT_SEANCE_PAUSE;
 break;
 case ETAT_SEANCE_PAUSE:
 // La séance est en pause, il faut la reprendre
 reprendreSeance();
 etatSeance = ETAT_SEANCE_DEMARREE;
 break;
 default:
 break;
}
}

```

### 9.5.2.2 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actualiserIHMAppareilsBluetooth ( boolean tableEstConnectee, boolean lanceurEstConnecte, boolean ecranEstConnecte ) [private]`

Méthode `actualiserIHMAppareilsBluetooth()` permettant de modifier les voyants des appareils Bluetooth en fonction des appareils connectés

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantEcran`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantLanceur`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantTable`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis()`.

```

{
 if (tableEstConnectee)
 voyantTable.setImageResource(R.drawable.table_connectee);
 else
 voyantTable.setImageResource(R.drawable.table_deconnectee);

 if (lanceurEstConnecte)
 voyantLanceur.setImageResource(R.drawable.lanceur_connecte);
 else
 voyantLanceur.setImageResource(R.drawable.lanceur_deconnecte);

 if (ecranEstConnecte)
 voyantEcran.setImageResource(R.drawable.ecran_connecte);
 else
 voyantEcran.setImageResource(R.drawable.ecran_deconnecte);
}

```

### 9.5.2.3 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM ( ) [private]`

Méthode `affecterMembresIHM()` permettant l'affectation des membres de l'IHM (boutons, barres de progressions, textes, ...) aux attributs correspondants.

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionEffetBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionFrequenceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionNombreBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionPuissanceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionRotationLanceur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonAppliquerNomJoueur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonArreterSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonBluetooth`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonHistorique`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonParametres`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.spinnerListeJoueurs`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurEffetBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurFrequenceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurNombreBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurPuissanceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurRotationLanceur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantEcran`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantLanceur`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantTable`.

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "affecterMembresIHM()");

 //editTextNomJoueur = (EditText) findViewById(R.id.editTextNomJoueur);
 boutonAppliquerNomJoueur = (ImageButton) findViewById(R.id.
boutonAppliquerNomJoueur);
 boutonBluetooth = (ImageButton) findViewById(R.id.boutonBluetooth);
 boutonHistorique = (ImageButton) findViewById(R.id.boutonHistorique);
 boutonParametres = (ImageButton) findViewById(R.id.boutonParametres);
 voyantTable = (ImageView) findViewById(R.id.voyantTable);
 voyantLanceur = (ImageView) findViewById(R.id.voyantLanceur);
 voyantEcran = (ImageView) findViewById(R.id.voyantEcran);
 boutonActionSeance = (ImageView) findViewById(R.id.boutonActionSeance);
 boutonArreterSeance = (ImageView) findViewById(R.id.boutonArreterSeance
);
 barreProgressionNombreBalles = (SeekBar) findViewById(R.id.
barreProgressionNombreBalles);
 texteValeurNombreBalles = (TextView) findViewById(R.id.
texteValeurNombreBalles);
 barreProgressionFrequenceBalles = (SeekBar) findViewById(R.id.
barreProgressionFrequenceBalles);
 texteValeurFrequenceBalles = (TextView) findViewById(R.id.
texteValeurFrequenceBalles);
 barreProgressionEffetBalles = (SeekBar) findViewById(R.id.
barreProgressionEffetBalles);
 texteValeurEffetBalles = (TextView) findViewById(R.id.
texteValeurEffetBalles);
 barreProgressionPuissanceBalles = (SeekBar) findViewById(R.id.
barreProgressionPuissanceBalles);
 texteValeurPuissanceBalles = (TextView) findViewById(R.id.
texteValeurPuissanceBalles);
 barreProgressionRotationLanceur = (SeekBar) findViewById(R.id.
barreProgressionRotationLanceur);
 texteValeurRotationLanceur = (TextView) findViewById(R.id.
texteValeurRotationLanceur);
 spinnerListeJoueurs = (Spinner) findViewById(R.id.spinnerListeJoueurs);
 spinnerListeJoueurs.setContentDescription("La liste des joueurs");
}
```

#### 9.5.2.4 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ajouterJoueur ( ) [private]

Méthode [ajouterJoueur\(\)](#) ajoutant un joueur saisi à la base de données.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#), [com.ttpa.iris.ttpamobile.-ServeurBDD.insererJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.nomJoueur](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.serveurBDD](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#).

```
{
 AlertDialog.Builder ajoutJoueur = new AlertDialog.Builder(this);
 LayoutInflater factory = LayoutInflater.from(this);
 final View ajoutJoueurView = factory.inflate(R.layout.ajout_joueur,
null);
 ajoutJoueur.setView(ajoutJoueurView);

 ajoutJoueur.setTitle("Ajouter un nouveau joueur");

 ajoutJoueur.setPositiveButton("Valider", new DialogInterface.
OnClickListener()
 {
 public void onClick(DialogInterface dialog, int which)
 {
 //Lorsque l'on cliquera sur le bouton "OK", on récupère
l'EditText correspondant à notre vue personnalisée (cad à alertDialogView)
 EditText nomJoueur = (EditText)ajoutJoueurView.findViewById(R.
id.editTextNom);
 Joueur joueur = new Joueur(nomJoueur.getText().toString());
 long id = serveurBDD.insererJoueur(joueur);
 Log.d("IHMEcranPrincipal", "Nom joueur : " + nomJoueur.getText(
).toString() + " - id : " + id);
 Toast.makeText(getApplicationContext(), "Joueur " + nomJoueur.
getText() + " ajouté", Toast.LENGTH_SHORT).show();
 creerListeJoueurs();
 }
 });
}
```

```

 ajoutJoueur.setNegativeButton("Annuler", new DialogInterface.
onClickListener()
{
 public void onClick(DialogInterface dialog, int which)
 {

 }

});
ajoutJoueur.show();
}

```

#### 9.5.2.5 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance ( )` [private]

Méthode `appliquerParametresSeance()` appliquant les valeurs des paramètres à l'objet `parametresActuels`.

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionEffetBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionFrequenceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionNombreBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionPuissanceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionRotationLanceur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_ARRETEE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.nomJoueur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels`, `com.ttpa.iris.ttpamobile.ParametreSeance.setEffetBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setFrequenceBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setIntensiteEffet()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setNombreBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setNomJoueur()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setPuissanceBalles()`, et `com.ttpa.iris.ttpamobile.ParametreSeance.setRotation()`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate()`.

```

{
 Log.d("IHMEcranPrincipal", "appliquerParametresSeance()");

 if(etatSeance == ETAT_SEANCE_ARRETEE)
 {
 parametresActuels.setNomJoueur(nomJoueur);
 parametresActuels.setNombreBalles((barreProgressionNombreBalles.
getProgress() * 5) + 5); // 5 balles par palier, 5 balles minimum
 parametresActuels.setFrequenceBalles((
barreProgressionFrequenceBalles.getProgress() * 5) + 30); // 5 balles par
palier, 30 balles minimum

 String effet;
 int intensiteEffet = barreProgressionEffetBalles.getProgress() - 8;

 if (barreProgressionEffetBalles.getProgress() == 8)
 {
 effet = "Aucun";
 intensiteEffet = 1;
 }
 else if (barreProgressionEffetBalles.getProgress() < 8)
 {
 effet = "Coupé";
 intensiteEffet = 0 - intensiteEffet;
 }
 else
 effet = "Lifté";

 parametresActuels.setEffetBalles(effet);
 parametresActuels.setIntensiteEffet(intensiteEffet);
 parametresActuels.setPuissanceBalles(barreProgressionPuissanceBalles
.getProgress() + 1); // 1 minimum (soit 10% minimum)
 parametresActuels.setRotation(barreProgressionRotationLanceur.
getProgress() * 5); // 5° par pallier
 }
}

```

### 9.5.2.6 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance ( boolean *seanceAEnregistrer* ) [private]

Méthode [arreterSeance\(\)](#) permettant d'envoyer les trames correspondantes aux appareils Bluetooth du projet.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionEffetBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionFrequenceBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionNombreBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionPuissanceBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionRotationLanceur](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonArreterSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothTable\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT\\_SEANCE\\_ARRETEE](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance](#), [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.seanceEnCours](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.serveurBDD](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementsBallesJouees\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "arreterSeance()");
 // Envoyer la trame d'arrêt à l'écran
 envoyerTrameArretPeripheriqueBluetoothEcran();

 // Envoyer la trame d'arrêt au lanceur
 envoyerTrameArretPeripheriqueBluetoothLanceur();

 // Envoyer la trame d'arrêt à la table
 envoyerTrameArretPeripheriqueBluetoothTable();

 if (seanceAEnregistrer)
 {
 // Enregistrer la séance dans la base de données
 serveurBDD.insererSeance(seanceEnCours);
 }

 // Changer l'icône du bouton d'action
 boutonActionSeance.setImageResource(R.drawable.bouton_demarrer);
 boutonActionSeance.setEnabled(true);

 // Changer l'état et la visibilité du bouton d'arrêt de séance
 boutonArreterSeance.setEnabled(false);
 boutonArreterSeance.setVisibility(View.INVISIBLE);

 // Changer l'état de la séance
 etatSeance = ETAT_SEANCE_ARRETEE;

 // Changer les états des barres de paramétrage
 barreProgressionNombreBalles.setEnabled(true);
 barreProgressionFrequenceBalles.setEnabled(true);
 barreProgressionEffetBalles.setEnabled(true);
 barreProgressionPuissanceBalles.setEnabled(true);
 barreProgressionRotationLanceur.setEnabled(true);
}
```

### 9.5.2.7 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.attendre ( int *tempsMillisecondes* ) [private]

Méthode [attendre\(\)](#) permettant d'attendre un temps données.

Paramètres

|                                 |  |
|---------------------------------|--|
| <i>temps-<br/>Millisecondes</i> |  |
|---------------------------------|--|

```
{
```

```

Log.d("IHMEcranPrincipal", "attendre()");

try
{
 Thread.sleep(tempsMillisecondes);
}
catch (InterruptedException e)
{
 e.printStackTrace();
}
}

```

#### 9.5.2.8 void `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance ( int zoneTouchee )` [private]

Méthode `calculerReussiteSeance()` calculant le taux de réussite de la séance selon le nombre de balles ayant déjà touché l'objectif.

##### Paramètres

|                          |                            |
|--------------------------|----------------------------|
| <code>zoneTouchee</code> | étant la zone de l'impacte |
|--------------------------|----------------------------|

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ballesJouees`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ballesReussies`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_DEMARREE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance`, `com.ttpa.iris.ttpamobile.Seance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.Seance.getZoneObjectif()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.seanceEnCours`, et `com.ttpa.iris.ttpamobile.Seance.setTauxReussite()`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesTable()`.

```

{
 Log.d("IHMEcranPrincipal", "calculerReussiteSeance() zone touchée : " +
zoneTouchee);

 if(etatSeance == ETAT_SEANCE_DEMARREE)
 {
 if (zoneTouchee != -1) // Si la balle a bien touché la table
 {
 if ((seanceEnCours.getZoneObjectif() != -1) && (seanceEnCours.
getZoneObjectif() != 0)) { // Si l'objectif a été défini
 if (zoneTouchee == seanceEnCours.getZoneObjectif()) // Si
la zone touchée est la même que l'objectif
 ballesReussies++;
 } else // Si aucun objectif n'a été défini: la table entière
est l'objectif
 ballesReussies++;
 }

 seanceEnCours.setTauxReussite((float) ballesReussies / (float)
seanceEnCours.getNombreBalles() * 100);

 if (ballesJouees == seanceEnCours.getNombreBalles())
 arreterSeance(true);
 }
}

```

#### 9.5.2.9 void `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionBaseDeDonnees ( )` [private]

Méthode `connexionBaseDeDonnees()` permettant la création puis la connexion à la base de données.

Références `com.ttpa.iris.ttpamobile.ServeurBDD.open()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.serveurBDD`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate()`.

```

{
 Log.d("IHMEcranPrincipal", "connexionBaseDeDonnees()");
}

```



```

 serveurBDD = new ServeurBDD(this);
 serveurBDD.open();
}

```

#### 9.5.2.10 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothEcran (BluetoothDevice *appareilBluetooth*) [private]

Méthode [connexionPeripheriqueBluetoothEcran\(\)](#) permettant la connexion Bluetooth à l'écran.

##### Paramètres

|                           |                               |
|---------------------------|-------------------------------|
| <i>appareil-Bluetooth</i> | étant le Bluetooth de l'écran |
|---------------------------|-------------------------------|

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.handler](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothEcran](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "connexionPeripheriqueBluetoothEcran()");

 peripheriqueBluetoothEcran = new PeripheriqueBluetooth(
 appareilBluetooth, handler);

 //Toast.makeText(getApplicationContext(), "Connexion à l'écran ...",
 Toast.LENGTH_SHORT).show();

 peripheriqueBluetoothEcran.connecter();

 //attendre(2000);

 if (peripheriqueBluetoothEcran.estConnecte())
 Toast.makeText(getApplicationContext(), "Ecran connecté !", Toast.
 LENGTH_SHORT).show();
}

```

#### 9.5.2.11 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothLanceur (BluetoothDevice *appareilBluetooth*) [private]

Méthode [connexionPeripheriqueBluetoothLanceur\(\)](#) permettant la connexion Bluetooth du lanceur.

##### Paramètres

|                           |                               |
|---------------------------|-------------------------------|
| <i>appareil-Bluetooth</i> | étant le Bluetooth du lanceur |
|---------------------------|-------------------------------|

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.handler](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "connexionPeripheriqueBluetoothLanceur()");

 peripheriqueBluetoothLanceur = new PeripheriqueBluetooth(
 appareilBluetooth, handler);

 //Toast.makeText(getApplicationContext(), "Connexion au lanceur ...",
 Toast.LENGTH_SHORT).show();

 peripheriqueBluetoothLanceur.connecter();

 //attendre(2000);

 if (peripheriqueBluetoothLanceur.estConnecte())

```

```

 Toast.makeText(getApplicationContext(), "Lanceur connecté !", Toast
 .LENGTH_SHORT).show();
 }

```

#### 9.5.2.12 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothTable (BluetoothDevice *appareilBluetooth* ) [private]

Méthode [connexionPeripheriqueBluetoothTable\(\)](#) permettant la connexion Bluetooth à la table.

##### Paramètres

|                           |                                |
|---------------------------|--------------------------------|
| <i>appareil-Bluetooth</i> | étant le Bluetooth de la table |
|---------------------------|--------------------------------|

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.handler](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothTable](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "connexionPeripheriqueBluetoothTable()");

 peripheriqueBluetoothTable = new PeripheriqueBluetooth(
 appareilBluetooth, handler);

 //Toast.makeText(getApplicationContext(), "Connexion à la table ...",
 Toast.LENGTH_SHORT).show();

 peripheriqueBluetoothTable.connecter();

 //attendre(2000);

 if (peripheriqueBluetoothTable.estConnecte())
 Toast.makeText(getApplicationContext(), "Table connectée !", Toast.
 LENGTH_SHORT).show();
}

```

#### 9.5.2.13 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth ( ) [private]

Méthode [connexionPeripheriquesBluetooth\(\)](#) permettant la connexion aux appareils Bluetooth du projet détectés.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothTable\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.deconnexionPeripheriquesBluetooth\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.devices](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.handler](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM\\_PERIPHERIQUE\\_BLUETOOTH\\_ECRAN](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM\\_PERIPHERIQUE\\_BLUETOOTH\\_LANCEUR](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM\\_PERIPHERIQUE\\_BLUETOOTH\\_TABLE](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriques](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "connexionPeripheriquesBluetooth()");

 // Déconnexion de tous les appareils avant une possible connexion
 deconnexionPeripheriquesBluetooth();

 for (BluetoothDevice appareilBluetooth : devices)
 {
 //Toast.makeText(getApplicationContext(), "Périphérique = " +
 appareilBluetooth.getName(), Toast.LENGTH_SHORT).show();
 peripheriques.add(new PeripheriqueBluetooth(appareilBluetooth,
 handler));
 }
}

```

```

switch (appareilBluetooth.getName())
{
 case NOM_PERIPHERIQUE_BLUETOOTH_ECRAN:
 connexionPeripheriqueBluetoothEcran(appareilBluetooth);
 break;
 case NOM_PERIPHERIQUE_BLUETOOTH_TABLE:
 connexionPeripheriqueBluetoothTable(appareilBluetooth);
 break;
 case NOM_PERIPHERIQUE_BLUETOOTH_LANCEUR:
 connexionPeripheriqueBluetoothLanceur(appareilBluetooth);
 break;
 default:
 break;
}

if(peripheriques.size() == 0)
{
 Toast.makeText(getApplicationContext(), "Aucun périphérique détecté
! ", Toast.LENGTH_SHORT).show();
}
}

```

#### 9.5.2.14 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs ( ) [private]

Méthode `creerListeJoueurs()` créant la liste des joueurs présents dans la base de données

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameConnexionPeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_ARRETEE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance`, `com.ttpa.iris.ttpamobile.Joueur.getId()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getIdJoueurParametres()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueur()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueurs()`, `com.ttpa.iris.ttpamobile.Joueur.getNom()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.nomJoueur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.serveurBDD`, `com.ttpa.iris.ttpamobile.ServeurBDD.setIdJoueurParametres()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setNomJoueur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.spinnerListeJoueurs`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ajouterJoueur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate()`.

```

{
 final List<Joueur> listeJoueurs = serveurBDD.getJoueurs();
 final List<String> noms = new ArrayList<String>();

 // le dernier joueur a avoir utilisé l'application
 int idJoueur = serveurBDD.getIdJoueurParametres();
 for(int i = 0; i < listeJoueurs.size(); i++)
 {
 Joueur joueur = listeJoueurs.get(i);
 if(joueur.getId() == idJoueur)
 {
 noms.add(joueur.getNom());
 break;
 }
 }
 for(int i = 0; i < listeJoueurs.size(); i++)
 {
 Joueur joueur = listeJoueurs.get(i);
 if(joueur.getId() == idJoueur)
 continue;
 noms.add(joueur.getNom());
 }

 ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, noms);
 adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
 spinnerListeJoueurs.setAdapter(adapter);

 spinnerListeJoueurs.setOnItemClickListener(new AdapterView.OnItemClickListener()
 {
 @Override
 public void onItemClick(AdapterView<?> arg0, View arg1, int position, long id)
 }
}

```

```

 {
 Joueur joueur = serveurBDD.getJoueur(noms.get(position));
 Log.d("IHMEcranPrincipal", "Nom joueur sélectionné : " + noms.
get(position));
 // On conserve son id pour la prochaine session
 serveurBDD.setIdJoueurParametres(joueur.getId());
 nomJoueur = joueur.getNom();
 if(etatSeance == ETAT_SEANCE_ARRETEE)
 parametresActuels.setNomJoueur(nomJoueur);

 // On envoi l'information à l'écran, si la séance n'a pas
 encore été démarrée
 envoyerTrameConnexionPeripheriqueBluetoothEcran();
 }

 @Override
 public void onNothingSelected(AdapterView<?> arg0)
 {
 }
 });
}

```

#### 9.5.2.15 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.deconnexionPeripheriquesBluetooth ( ) [private]

Méthode [deconnexionPeripheriquesBluetooth\(\)](#) permettant la déconnexion des appareils Bluetooth du projet connectés.

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.deconnecter\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothEcran](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothTable](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "deconnexionPeripheriquesBluetooth()");

 if(peripheriqueBluetoothEcran != null)
 peripheriqueBluetoothEcran.deconnecter(true);

 if(peripheriqueBluetoothTable != null)
 peripheriqueBluetoothTable.deconnecter(true);

 if(peripheriqueBluetoothLanceur != null)
 peripheriqueBluetoothLanceur.deconnecter(true);

 modifierIHMBluetoothInsuffisant();
}

```

#### 9.5.2.16 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth ( ) [private]

Méthode [demarrageBluetooth\(\)](#) permettant le démarrage du Bluetooth puis la connexion automatique aux appareils du projet.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.adaptateurBluetooth](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.devices](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriques](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.REQUEST\\_CODE\\_ENABLE\\_BLUETOOTH](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "demarrageBluetooth()");

 adaptateurBluetooth = BluetoothAdapter.getDefaultAdapter();
 if (adaptateurBluetooth == null)
 {

```

```

 Toast.makeText(getApplicationContext(), "Bluetooth non activé !",
 Toast.LENGTH_SHORT).show();
 modifierIHMBluetoothInsuffisant();
 }
 else
 {
 if (!adaptateurBluetooth.isEnabled())
 {
 Toast.makeText(getApplicationContext(), "Bluetooth non activé !
 ", Toast.LENGTH_SHORT).show();

 modifierIHMBluetoothInsuffisant();

 Intent activeBlueTooth = new Intent (BluetoothAdapter.
 ACTION_REQUEST_ENABLE);
 startActivityForResult (activeBlueTooth,
 REQUEST_CODE_ENABLE_BLUETOOTH);
 //bluetoothAdapter.enable();
 }
 else
 {
 Toast.makeText(getApplicationContext(), "Bluetooth activé,
 recherche en cours...", Toast.LENGTH_LONG).show();

 // Recherche des périphériques connus
 peripheriques = new ArrayList<PeripheriqueBluetooth>();
 devices = adaptateurBluetooth.getBondedDevices();

 // Connexion aux appareils Bluetooth du projet détectés
 connexionPeripheriquesBluetooth();
 }
 }
}

```

#### 9.5.2.17 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance ( ) [private]

Méthode `demarrerSeance()` permettant d'envoyer les trames correspondantes aux appareils Bluetooth du projet.

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ballesJouees`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ballesReussies`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionEffetBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionFrequenceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionPuisseanceBalles`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionRotationLanceur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonArreterSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getEffetCompleet()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getFrequenceBalles()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getIdJoueurParametres()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getIntensiteEffet()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getPuisseanceBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getRotation()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.seanceEnCours`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.serveurBDD`, `com.ttpa.iris.ttpamobile.Seance.setldJoueur()`, `com.ttpa.iris.ttpamobile.Seance.setZoneObjectif()`, `com.ttpa.iris.ttpamobile.Seance.setZoneRobot()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneObjectif`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneRobot`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`.

```

{
 Log.d("IHMEcranPrincipal", "demarrerSeance()");

 // Lire les valeurs des paramètres actuels afin de les appliquer aux
 paramètres de la séance
 appliquerParametresSeance();

 // Envoyer la trame de paramétrage à l'écran, puis la trame de début de
 séance
}

```

```

envoyerTrameDebutSeancePeripheriqueBluetoothEcran();

envoyerTrameParametrageSeancePeripheriqueBluetoothEcran();

// On applique les paramètres actuels à la séance en cours
seanceEnCours = new Seance(parametresActuels.getFrequenceBalles(),
parametresActuels.getNombreBalles(), parametresActuels.getEffetComplet(),
parametresActuels.getIntensiteEffet(), parametresActuels.getPuissanceBalles(),
parametresActuels.getRotation());
seanceEnCours.setIdJoueur(serveurBDD.getIdJoueurParametres());
seanceEnCours.setZoneObjectif(zoneObjectif);
seanceEnCours.setZoneRobot(zoneRobot);

// Mettre à zéro les statistiques
ballesJouees = 0;
ballesReussies = 0;

// Envoyer la trame de départ au lanceur
envoyerTrameDebutSeancePeripheriqueBluetoothLanceur();

// Changer l'icône du bouton d'action
boutonActionSeance.setImageResource(R.drawable.bouton_pause);

// Changer l'état et la visibilité du bouton d'arrêt de séance
boutonArreterSeance.setEnabled(true);
boutonArreterSeance.setVisibility(View.VISIBLE);

// Changer les états des barres de paramétrage
barreProgressionNombreBalles.setEnabled(false);
barreProgressionFrequenceBalles.setEnabled(false);
barreProgressionEffetBalles.setEnabled(false);
barreProgressionPuissanceBalles.setEnabled(false);
barreProgressionRotationLanceur.setEnabled(false);
}

```

#### 9.5.2.18 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothEcran()` [private]

Méthode `envoyerTrameArretPeripheriqueBluetoothEcran()` envoyant la trame de fin de séance à l'écran

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ECRAN_FIN_SEANCE`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`.

```

{
 Log.d("IHMEcranPrincipal", "
envoyerTrameArretPeripheriqueBluetoothEcran()");

 envoyerTramePeripheriqueBluetoothEcran(TRAME_ECRAN_FIN_SEANCE);
}

```

#### 9.5.2.19 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothLanceur()` [private]

Méthode `envoyerTrameArretPeripheriqueBluetoothLanceur()` envoyant la trame de fin de séance au lanceur.

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_LANCEUR_ARRET_SEANCE`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`.

```

{
 Log.d("IHMEcranPrincipal", "
envoyerTrameArretPeripheriqueBluetoothLanceur()");

 if (peripheriqueBluetoothLanceur != null)
 {
 if (peripheriqueBluetoothLanceur.estConnecte())
 {
 peripheriqueBluetoothLanceur.envoyer(TRAME_LANCEUR_ARRET_SEANCE);
 }
 }
}

```

```

 Log.d("IHMEcranPrincipal", "Trame arrêt séance Lanceur : " +
TRAME_LANCEUR_ARRET_SEANCE);
 }
}

```

#### 9.5.2.20 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothTable ( ) [private]

Méthode [envoyerTrameArretPeripheriqueBluetoothTable\(\)](#) envoyant la trame de fin de séance au lanceur.

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothTable](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_TABLE\\_ARRET\\_SEANCE](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "
envoyerTrameArretPeripheriqueBluetoothTable ()");

 if (peripheriqueBluetoothTable != null)
 {
 if (peripheriqueBluetoothTable.estConnecte())
 {
 peripheriqueBluetoothTable.envoyer(TRAME_TABLE_ARRET_SEANCE);
 Log.d("IHMEcranPrincipal", "Trame arrêt séance Table : " +
TRAME_TABLE_ARRET_SEANCE);
 }
 }
}

```

#### 9.5.2.21 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameConnexionPeripheriqueBluetoothEcran ( ) [private]

Méthode [envoyerTrameConnexionPeripheriqueBluetoothEcran\(\)](#) envoyant la trame de connexion à l'écran.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.getNomJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_ENTETE](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "
envoyerTrameConnexionPeripheriqueBluetoothEcran ()");

 String trame = TRAME_ENTETE + ":CONNECT:" + parametresActuels.
getNomJoueur ();

 envoyerTramePeripheriqueBluetoothEcran(trame);
}

```

#### 9.5.2.22 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothEcran ( ) [private]

Méthode [envoyerTrameFinParametrageSeancePeripheriqueBluetoothEcran\(\)](#) envoyant la trame de fin de paramétrage à l'écran.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_ECRAN\\_DEBUT\\_SEANCE](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#).

```

{

```

```

 Log.d("IHMEcranPrincipal", "
 envoyerTrameDebutSeancePeripheriqueBluetoothEcran()");

 envoyerTramePeripheriqueBluetoothEcran(TRAME_ECRAN_DEBUT_SEANCE);
 }

```

#### 9.5.2.23 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur( ) [private]

Méthode `envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()` envoyant la trame de début de séance au lanceur.

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getEffet()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getFrequenceBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getIntensiteEffet()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getPuissanceBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getRotation()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ENTETE`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_FIN`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`.

```

{
 Log.d("IHMEcranPrincipal", "
 envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()");

 if (peripheriqueBluetoothLanceur != null)
 {
 if (peripheriqueBluetoothLanceur.estConnecte())
 {
 String trame = TRAME_ENTETE + ":SET:" + parametresActuels.
 getEffet() + ":" + parametresActuels.getIntensiteEffet() + ":" +
 parametresActuels.getPuissanceBalles() + ":" + parametresActuels.
 getFrequenceBalles() + ":" + parametresActuels.getRotation() + ":" +
 parametresActuels.getNombreBalles() + TRAME_FIN;
 peripheriqueBluetoothLanceur.envoyer(trame);
 Log.d("IHMEcranPrincipal", "Trame départ séance Lanceur : " +
 trame);
 }
 }
}

```

#### 9.5.2.24 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran( ) [private]

Méthode `envoyerTrameParametrageSeancePeripheriqueBluetoothEcran()` envoyant la trame de paramétrage à l'écran.

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ENTETE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneObjectif`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneRobot`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`.

```

{
 Log.d("IHMEcranPrincipal", "
 envoyerTrameParametrageSeancePeripheriqueBluetoothEcran()");

 String trame = TRAME_ENTETE + ":SETSEANCE:" + zoneRobot + ":" +
 zoneObjectif + ":" + parametresActuels.getNombreBalles();

 envoyerTramePeripheriqueBluetoothEcran(trame);
}

```



**9.5.2.25** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran (String trame) [private]

Méthode [envoyerTramePeripheriqueBluetoothEcran\(\)](#) envoyant la trame à l'écran, si l'écran est connecté.

#### Paramètres

|              |                          |
|--------------|--------------------------|
| <i>trame</i> | étant la trame à envoyer |
|--------------|--------------------------|

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothEcran](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameConnexionPeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesTable\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "envoyerTramePeripheriqueBluetoothEcran()");
 if(peripheriqueBluetoothEcran != null)
 {
 if (peripheriqueBluetoothEcran.estConnecte())
 peripheriqueBluetoothEcran.envoyer(trame);
 }
}
```

**9.5.2.26** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothLanceur (String trame) [private]

Méthode [envoyerTramePeripheriqueBluetoothLanceur\(\)](#) envoyant la trame au lanceur, si le lanceur est connecté.

#### Paramètres

|              |                          |
|--------------|--------------------------|
| <i>trame</i> | étant la trame à envoyer |
|--------------|--------------------------|

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "envoyerTramePeripheriqueBluetoothLanceur()");
 if(peripheriqueBluetoothLanceur != null)
 {
 if (peripheriqueBluetoothLanceur.estConnecte())
 peripheriqueBluetoothLanceur.envoyer(trame);
 }
}
```

**9.5.2.27** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur ( ) [private]

Méthode [envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur\(\)](#) envoyant la trame de reprise de séance au lanceur.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ballesJouees](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte\(\)](#), [com.ttpa.iris.ttpamobile.](#)

[ParametreSeance.getEffet\(\)](#), [com.tpa.iris.ttpamobile.ParametreSeance.getFrequenceBalles\(\)](#), [com.tpa.iris.ttpamobile.ParametreSeance.getIntensiteEffet\(\)](#), [com.tpa.iris.ttpamobile.ParametreSeance.getNombreBalles\(\)](#), [com.tpa.iris.ttpamobile.ParametreSeance.getPuissanceBalles\(\)](#), [com.tpa.iris.ttpamobile.ParametreSeance.getRotation\(\)](#), [com.tpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels](#), [com.tpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur](#), [com.tpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_ENTETE](#), et [com.tpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_FIN](#).

```
{
 Log.d("IHMEcranPrincipal", "
envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()");

 if (peripheriqueBluetoothLanceur != null)
 {
 if (peripheriqueBluetoothLanceur.estConnecte())
 {
 String trame = TRAME_ENTETE + ":SET:" + parametresActuels.
getEffet() + ":" + parametresActuels.getIntensiteEffet() + ":" +
parametresActuels.getPuissanceBalles() + ":" + parametresActuels.
getFrequenceBalles() + ":" + parametresActuels.getRotation() + ":" + (
parametresActuels.getNombreBalles() - ballesJouees) + TRAME_FIN;
 peripheriqueBluetoothLanceur.envoyer(trame);
 Log.d("IHMEcranPrincipal", "Trame reprise séance Lanceur : " +
trame);
 }
 }
}
```

#### 9.5.2.28 void com.tpa.iris.ttpamobile.IHMEcranPrincipal.incrementsBallesJouees ( ) [private]

Méthode [incrementsBallesJouees\(\)](#) permettant de traiter les nombre de balles jouées de la séance, et d'arrêter la séance si la dernière balle est en dehors.

Références [com.tpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance\(\)](#), [com.tpa.iris.ttpamobile.IHMEcranPrincipal.ballesJouees](#), [com.tpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance](#), [com.tpa.iris.ttpamobile.IHMEcranPrincipal.ETAT\\_SEANCE\\_DEMARREE](#), [com.tpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance](#), [com.tpa.iris.ttpamobile.Seance.getNombreBalles\(\)](#), et [com.tpa.iris.ttpamobile.IHMEcranPrincipal.seanceEnCours](#).

Référencé par [com.tpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesTable\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "incrementsBallesJouees()");

 if(etatSeance == ETAT_SEANCE_DEMARREE)
 {
 ++ballesJouees;

 if (ballesJouees == seanceEnCours.getNombreBalles())
 {
 boutonActionSeance.setEnabled(false);

 final Timer timerAsync = new Timer();
 final TimerTask timerTaskAsync = new TimerTask() {
 @Override
 public void run() {
 runOnUiThread(new Runnable() {
 @Override public void run() {
 if(etatSeance == ETAT_SEANCE_DEMARREE)
 arreterSeance(true);

 timerAsync.cancel();
 }
 });
 }
 };
 timerAsync.schedule(timerTaskAsync, 2000, 2000);
 }
 }
}
```

#### 9.5.2.29 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant ( ) [private]

Méthode `modifierIHMBluetoothInsuffisant()` appelée lorsque le Bluetooth n'est pas activé ou que les appareils nécessaires ne sont pas présents/connectés

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonBluetooth`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonParametres`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR_BOUTON_ROUGE`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.deconnexionPeripheriquesBluetooth()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis()`.

```
{
 Log.d("IHMEcranPrincipal", "modifierIHMBluetoothInsuffisant()");

 // Arrêter la séance en cours, sans enregistrer
 arreterSeance(false);

 // Modifier la couleur du bouton Bluetooth
 boutonBluetooth.setBackgroundColor(COULEUR_BOUTON_ROUGE);
 // Modifier la couleur du bouton réglages des zones
 boutonParametres.setBackgroundColor(COULEUR_BOUTON_ROUGE);
 // Autoriser le réglage des zones
 boutonParametres.setEnabled(false);
 // Changer l'icône du bouton d'action
 boutonActionSeance.setImageResource(R.drawable.
bouton_demarrer_desactive);
 // Empêcher une séance d'être jouée
 boutonActionSeance.setEnabled(false);
}
```

#### 9.5.2.30 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothOperationnel ( ) [private]

Méthode `modifierIHMBluetoothOperationnel()` appelée lorsque tous les appareils Bluetooth sont connectés.

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonBluetooth`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonParametres`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR_BOUTON_VERT`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis()`.

```
{
 Log.d("IHMEcranPrincipal", "modifierIHMBluetoothOperationnel()");

 // Modifier la couleur du bouton Bluetooth
 boutonBluetooth.setBackgroundColor(COULEUR_BOUTON_VERT);
 // Modifier la couleur du bouton réglages des zones
 boutonParametres.setBackgroundColor(COULEUR_BOUTON_VERT);
 // Autoriser le réglage des zones
 boutonParametres.setEnabled(true);
 // Changer l'icône du bouton d'action
 boutonActionSeance.setImageResource(R.drawable.bouton_demarrer);
 // Autoriser le déroulement d'une séance
 boutonActionSeance.setEnabled(true);
}
```

#### 9.5.2.31 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothSuffisant ( ) [private]

Méthode `modifierIHMBluetoothSuffisant()` appelée lorsque les appareils Bluetooth nécessaires sont connectés.

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonBluetooth`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonParametres`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR_BOUTON_VERT`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR_BOUTON_ROUGE`.

[iris.ttpamobile.IHMEcranPrincipal.COULEUR\\_BOUTON\\_ORANGE](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR\\_BOUTON\\_VERT](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "modifierIHMBluetoothSuffisant()");

 // Modifier la couleur du bouton Bluetooth
 boutonBluetooth.setBackgroundColor(COULEUR_BOUTON_ORANGE);
 // Modifier la couleur du bouton réglages des zones
 boutonParametres.setBackgroundColor(COULEUR_BOUTON_VERT);
 // Autoriser le réglage des zones
 boutonParametres.setEnabled(true);
 // Changer l'icône du bouton d'action
 boutonActionSeance.setImageResource(R.drawable.bouton_demarrer);
 // Autoriser le déroulement d'une séance
 boutonActionSeance.setEnabled(true);
}
```

#### 9.5.2.32 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM ( ) [private]

Méthode [modifierValeursParametresIHM\(\)](#) modifiant les valeurs des champs de paramètre en fonction de leur barre de progression correspondante.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionEffetBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionFrequenceBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionNombreBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionPuissanceBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionRotationLanceur](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurEffetBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurFrequenceBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurNombreBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurPuissanceBalles](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurRotationLanceur](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onProgressChanged\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "modifierValeursParametresIHM()");

 texteValeurNombreBalles.setText(((barreProgressionNombreBalles.
getProgress() * 5) + 5) + " balles"); // 5 balles par palier, 5 balles minimum
 texteValeurFrequenceBalles.setText(((barreProgressionFrequenceBalles.
getProgress() * 5) + 30) + " balles/min"); // 5 balles par palier, 30 balles
 minimum

 String effet;
 int intensiteEffet = barreProgressionEffetBalles.getProgress() - 8;

 if (barreProgressionEffetBalles.getProgress() == 8)
 effet = "Aucun";
 else if (barreProgressionEffetBalles.getProgress() < 8)
 {
 effet = "Coupé";
 intensiteEffet = 0 - intensiteEffet;
 }
 else
 effet = "Lifté";

 switch (effet)
 {
 case "Aucun":
 texteValeurEffetBalles.setText (effet);
 break;
 default:
 texteValeurEffetBalles.setText (effet + " " + intensiteEffet);
 } // 10% par palier

 texteValeurPuissanceBalles.setText(((barreProgressionPuissanceBalles.
getProgress() * 10) + 10) + "%"); // 10% par palier, 10% minimum
```

```

String stringRotation;
int rotationActuelle = barreProgressionRotationLanceur.getProgress() *
5; // 5° par pallier

if(rotationActuelle > 45)
 stringRotation = (rotationActuelle - 45) + "° à droite";
else if(rotationActuelle < 45)
 stringRotation = (45 - rotationActuelle) + "° à gauche";
else
 stringRotation = (rotationActuelle - 45) + "°";

texteValeurRotationLanceur.setText(stringRotation);
}

```

### 9.5.2.33 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick ( View element )

Méthode onClick pour la gestion de l'évènement d'un click.

#### Paramètres

|                |                                                   |
|----------------|---------------------------------------------------|
| <i>element</i> | étant la vue sur laquelle l'utilisateur a cliqué. |
|----------------|---------------------------------------------------|

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ajouterJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonAppliquerNomJoueur](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonArreterSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonBluetooth](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonHistorique](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonParametres](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameConnexionPeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.nomJoueur](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.redirectionActiviteHistorique\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.SELECTION\\_ZONE\\_ROBOT](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.setNomJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis\(\)](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ajouterJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone\(\)](#).

```

{
 Log.d("IHMEcranPrincipal", "onClick()");

 if(element == boutonAppliquerNomJoueur)
 {
 ajouterJoueur();
 }
 else if(element == boutonBluetooth)
 {
 if(!verifierConnexionAppareilsBluetoothRequis()) // Si tous les
appareils Bluetooth ne sont pas connectés
 {
 //Toast.makeText(getApplicationContext(), "Attendre..",
Toast.LENGTH_SHORT).show();
 demarrageBluetooth();
 }
 }
 else if(element == boutonHistorique)
 {
 Log.d("IHMEcranPrincipal", "onClick() boutonHistorique");

 // Redirection vers l'activité de l'historique des séances
redirectionActiviteHistorique();
 }
 else if(element == boutonParametres)
 {
 // Gérer le click sur le bouton paramètres (engrenage)
selectionnerZone(SELECTION_ZONE_ROBOT, 0, 0);
 }
 else if(element == boutonActionSeance)
 {
 Log.d("IHMEcranPrincipal", "onClick() boutonActionSeance");
 parametresActuels.setNomJoueur(nomJoueur);
 envoyerTrameConnexionPeripheriqueBluetoothEcran();
 }
}

```

```

 actionnerSeance();
 }
 else if(element == boutonArreterSeance)
 {
 Log.d("IHMEcranPrincipal", "onClick() boutonArreterSeance");
 arreterSeance(false);
 }
}

```

#### 9.5.2.34 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate ( Bundle savedInstanceState )` [protected]

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionBaseDeDonnees()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBarresProgressionIHMUtilisables()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables()`.

```

{
 Log.d("IHMEcranPrincipal", "onCreate()");

 super.onCreate(savedInstanceState);
 setContentView(R.layout.ecran_principal);

 // Affectation des membres de l'IHM
 affecterMembresIHM();

 // Modifier les valeurs des champs de paramètre de l'IHM en fonction
 // des états des barres de progression
 modifierValeursParametresIHM();

 // Rendre les boutons présents dans l'IHM cliquables
 rendreBoutonsIHMClickables();

 // Rendre les barres de progression présents dans l'IHM utilisables
 rendreBarresProgressionIHMUtilisables();

 // Connexion à la base de données
 connexionBaseDeDonnees();

 // Crée une liste des joueurs enregistrés
 creerListeJoueurs();

 // Appliquer les paramètres de séance actuels
 appliquerParametresSeance();

 // Démarrage du Bluetooth puis connexion aux appareils du projet
 // détectés
 demarrageBluetooth();
}

```

#### 9.5.2.35 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onProgressChanged ( SeekBar seekBar, int progress, boolean fromUser )`

Méthode `onProgressChanged()` appelée lorsqu'une barre de progression est modifiée.

##### Paramètres

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| <i>seekBar</i>  | étant la barre de progression modifiée                      |
| <i>progress</i> | étant le progrès actuel de la barre de progression modifiée |
| <i>fromUser</i> |                                                             |

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`.

```

{
 modifierValeursParametresIHM();
}

```

**9.5.2.36 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onStartTrackingTouch ( SeekBar seekBar )**

Méthode [onStartTrackingTouch\(\)](#) appelée lorsqu'une barre de progression commence à être modifiée.

**Paramètres**

|                |                                        |
|----------------|----------------------------------------|
| <i>seekBar</i> | étant la barre de progression modifiée |
|----------------|----------------------------------------|

```
{ }
```

**9.5.2.37 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onStopTrackingTouch ( SeekBar seekBar )**

Méthode [onStopTrackingTouch\(\)](#) appelée lorsqu'une barre de progression a fini d'être modifiée.

**Paramètres**

|                |                                        |
|----------------|----------------------------------------|
| <i>seekBar</i> | étant la barre de progression modifiée |
|----------------|----------------------------------------|

```
{
 //appliquerParametresSeance();
}
```

**9.5.2.38 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance ( ) [private]**

Méthode [pauserSeance\(\)](#) permettant de mettre en pause un séance en cours.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT\\_SEANCE\\_PAUSE](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_ECRAN\\_PAUSE\\_SEANCE](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_LANCEUR\\_PAUSE\\_SEANCE](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterErreurRecueLanceur\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "pauserSeance()");

 // Changer l'icône du bouton d'action
 boutonActionSeance.setImageResource(R.drawable.bouton_reprendre);

 envoyerTramePeripheriqueBluetoothEcran(TRAME_ECRAN_PAUSE_SEANCE);
 envoyerTramePeripheriqueBluetoothLanceur(TRAME_LANCEUR_PAUSE_SEANCE);

 etatSeance = ETAT_SEANCE_PAUSE;
}
```

**9.5.2.39 void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.redirectionActiviteHistorique ( ) [private]**

Méthode [redirectionActiviteHistorique\(\)](#) démarrnant l'activité [IHMHistoriqueSeances](#) permettant de visualiser l'historique des séances du joueur actuel.

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "redirectionActiviteHistorique()");

 Intent intent = new Intent(IHMEcranPrincipal.this, IHMHistoriqueSeances
 .class);
 startActivity(intent);
}
```

**9.5.2.40** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBarresProgressionIHMUtilisables ( )  
[private]

Méthode [rendreBarresProgressionIHMUtilisables\(\)](#)

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionEffetBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionFrequenceBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionNombreBalles](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionPuissanceBalles](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionRotationLanceur](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "rendreBarresProgressionIHMUtilisables()");

 barreProgressionNombreBalles.setOnSeekBarChangeListener(this);
 barreProgressionFrequenceBalles.setOnSeekBarChangeListener(this);
 barreProgressionEffetBalles.setOnSeekBarChangeListener(this);
 barreProgressionPuissanceBalles.setOnSeekBarChangeListener(this);
 barreProgressionRotationLanceur.setOnSeekBarChangeListener(this);
}
```

**9.5.2.41** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables ( )  
[private]

Méthode [rendreBoutonsIHMClickables\(\)](#)

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonAppliquerNomJoueur](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonArreterSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonBluetooth](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonHistorique](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonParametres](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onCreate\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "rendreBoutonsIHMClickables()");

 boutonAppliquerNomJoueur.setOnClickListener(this);
 boutonBluetooth.setOnClickListener(this);
 boutonHistorique.setOnClickListener(this);
 boutonParametres.setOnClickListener(this);
 boutonActionSeance.setOnClickListener(this);
 boutonArreterSeance.setOnClickListener(this);
}
```

**9.5.2.42** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance ( ) [private]

Méthode [reprendreSeance\(\)](#) permettant de reprendre une séance actuellement en pause.

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT\\_SEANCE\\_DEMARREE](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_ECRAN\\_REPRISE\\_SEANCE](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_LANCEUR\\_REPRISE\\_SEANCE](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "reprendreSeance()");

 // Changer l'icône du bouton d'action
 boutonActionSeance.setImageResource(R.drawable.bouton_pause);

 envoyerTramePeripheriqueBluetoothEcran(TRAME_ECRAN_REPRISE_SEANCE);
 envoyerTramePeripheriqueBluetoothLanceur(TRAME_LANCEUR_REPRISE_SEANCE);

 etatSeance = ETAT_SEANCE_DEMARREE;
}
```



9.5.2.43 `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone ( final int typeSelection, int choixObjectif, int choixRobot ) [private]`

Méthode `selectionnerZone()` affichant une boîte de dialogue permettant de sélectionner la zone du robot ou la zone de l'objectif.

#### Paramètres

|                      |  |
|----------------------|--|
| <i>typeSelection</i> |  |
| <i>choixObjectif</i> |  |
| <i>choixRobot</i>    |  |

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.idZonePrecedente`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.idZonePrecedenteObjectif`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.idZonePrecedenteRobot`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.S-ELECTION_ZONE_OBJECTIF`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.S-ELECTION_ZONE_ROBOT`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.strZoneObjectif`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.strZoneRobot`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneObjectif`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneRobot`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick()`.

```
{
 Log.d("IHMEcranPrincipal", "selectionnerZone()");

 final AlertDialog.Builder selectionZone = new AlertDialog.Builder(this)
;
 LayoutInflater factory = LayoutInflater.from(this);
 final View selectionZoneView = factory.inflate(R.layout.zones, null);
 selectionZone.setView(selectionZoneView);

 List<Button> boutonsZone = new ArrayList<Button>();

 Button btnZone1 = (Button)selectionZoneView.findViewById(R.id.case1);
 boutonsZone.add(btnZone1);
 Button btnZone2 = (Button)selectionZoneView.findViewById(R.id.case2);
 boutonsZone.add(btnZone2);
 Button btnZone3 = (Button)selectionZoneView.findViewById(R.id.case3);
 boutonsZone.add(btnZone3);
 Button btnZone4 = (Button)selectionZoneView.findViewById(R.id.case4);
 boutonsZone.add(btnZone4);
 Button btnZone5 = (Button)selectionZoneView.findViewById(R.id.case5);
 boutonsZone.add(btnZone5);
 Button btnZone6 = (Button)selectionZoneView.findViewById(R.id.case6);
 boutonsZone.add(btnZone6);
 Button btnZone7 = (Button)selectionZoneView.findViewById(R.id.case7);
 boutonsZone.add(btnZone7);
 Button btnZone8 = (Button)selectionZoneView.findViewById(R.id.case8);
 boutonsZone.add(btnZone8);
 Button btnZone9 = (Button)selectionZoneView.findViewById(R.id.case9);
 boutonsZone.add(btnZone9);

 if(choixObjectif > 0)
 {
 Button btnZone = boutonsZone.get(choixObjectif - 1);
 btnZone.setBackgroundResource(R.drawable.case_cible);
 }
 if(choixRobot > 0)
 {
 Button btnZone = boutonsZone.get(choixRobot - 1);
 btnZone.setBackgroundResource(R.drawable.case_robot);
 }

 for(int i = 0; i < 9; i++)
 {
 Button btnZone = boutonsZone.get(i);

 // Changer l'icône de fond du bouton si la zone correspondante est
 // actuellement occupée par le robot ou l'objectif
 if((i + 1) == zoneRobot) // i + 1 étant le numéro de zone actuel
 btnZone.setBackgroundResource(R.drawable.case_robot);

 /*if((i + 1) == zoneObjectif) // i + 1 étant le numéro de zone
 actuel
```

```

 btnZone.setBackgroundResource(R.drawable.case_cible);*/

 btnZone.setOnClickListener(new View.OnClickListener() {
 public void onClick(View v) {
 Button myButton = (Button)selectionZoneView.findViewById(v.
getId());
 int zoneActuelle = Integer.parseInt(myButton.getText().
toString().substring(myButton.getText().toString().length() - 1));

 switch (typeSelection)
 {
 case SELECTION_ZONE_OBJECTIF:
 idZonePrecedente = idZonePrecedenteObjectif;
 break;
 case SELECTION_ZONE_ROBOT:
 idZonePrecedente = idZonePrecedenteRobot;
 break;
 }

 if(idZonePrecedente != -1)
 {
 Button myButtonPrecedent = (Button)selectionZoneView.
findViewById(idZonePrecedente);
 int numeroZonePrecedente = Integer.parseInt(
myButtonPrecedent.getText().toString().substring(myButtonPrecedent.getText().toString().
length() - 1));

 switch (typeSelection)
 {
 case SELECTION_ZONE_OBJECTIF:
 if((v.getId() != idZonePrecedenteRobot) && (
myButtonPrecedent.getId() != idZonePrecedenteRobot))
 myButtonPrecedent.setBackgroundResource(R.
drawable.case_libre);
 break;
 /*case SELECTION_ZONE_ROBOT:
 if(v.getId() != idZonePrecedenteObjectif)
 myButtonPrecedent.setBackgroundResource(R.
drawable.case_cible);
 break;*/
 default:
 myButtonPrecedent.setBackgroundResource(R.
drawable.case_libre);
 break;
 }
 }

 switch (typeSelection)
 {
 case SELECTION_ZONE_OBJECTIF:
 if (v.getId() != idZonePrecedenteRobot)
 idZonePrecedenteObjectif = v.getId();
 break;
 case SELECTION_ZONE_ROBOT:
 idZonePrecedenteRobot = v.getId();
 break;
 }

 idZonePrecedente = v.getId();

 if(typeSelection == SELECTION_ZONE_OBJECTIF)
 {
 // Vérifier si l'objectif ne tombe pas sur la zone du
robot
 if(zoneActuelle != zoneRobot)
 {
 if (strZoneObjectif == myButton.getText().toString(
))
 {
 myButton.setBackgroundResource(R.drawable.
case_libre);
 strZoneObjectif = "ZONE 0";
 }
 else
 {
 myButton.setBackgroundResource(R.drawable.
case_cible);
 strZoneObjectif = myButton.getText().toString()
;
 }
 }

 Log.d("IHMEcranPrincipal", "Objectif : " +
strZoneObjectif);
 }
 }
 });

```

```

 }
 else if(typeSelection == SELECTION_ZONE_ROBOT)
 {
 if (strZoneRobot == myButton.getText().toString())
 {
 myButton.setBackgroundResource(R.drawable.
case_libre);
 strZoneRobot = "ZONE 0";
 }
 else
 {
 myButton.setBackgroundResource(R.drawable.
case_robot);
 strZoneRobot = myButton.getText().toString();
 }
 Log.d("IHMEcranPrincipal", "Robot : " + strZoneRobot);
 }
}
});
}

if(typeSelection == SELECTION_ZONE_OBJECTIF)
 selectionZone.setTitle("Placer l'objectif");
if(typeSelection == SELECTION_ZONE_ROBOT)
 selectionZone.setTitle("Placer le robot");

selectionZone.setPositiveButton("Valider", new DialogInterface.
OnClickListener()
{
 public void onClick(DialogInterface dialog, int which)
 {
 if(typeSelection == SELECTION_ZONE_OBJECTIF)
 {
 Log.d("IHMEcranPrincipal", "Zone objectif validée : " +
strZoneObjectif);
 zoneObjectif = Integer.parseInt(strZoneObjectif.substring(
strZoneObjectif.length() - 1));
 }
 else if(typeSelection == SELECTION_ZONE_ROBOT)
 {
 Log.d("IHMEcranPrincipal", "Zone robot validée : " +
strZoneRobot);
 zoneRobot = Integer.parseInt(strZoneRobot.substring(
strZoneRobot.length() - 1));
 // Puis sélectionner l'objectif
 selectionnerZone(SELECTION_ZONE_OBJECTIF, 0, 0);
 }

 if (idZonePrecedente == -1)
 idZonePrecedente = 0;
 }
});

selectionZone.setNegativeButton("Annuler", new DialogInterface.
OnClickListener()
{
 public void onClick(DialogInterface dialog, int which)
 {
 if(typeSelection == SELECTION_ZONE_OBJECTIF)
 {
 strZoneObjectif = "";
 zoneObjectif = 0;
 }
 else if(typeSelection == SELECTION_ZONE_ROBOT)
 {
 strZoneRobot = "";
 zoneRobot = 0;
 // Puis sélectionner l'objectif
 selectionnerZone(SELECTION_ZONE_OBJECTIF, 0, 0);
 }
 }
});

selectionZone.show();
}

```

**9.5.2.44** `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecues ( String nomAppareilSource, String donnees ) [private]`

Méthode `traiterDonneesRecues()` permettant le traitement des données bluetooth reçues en fonction de l'appareil source de ces données.

**Paramètres**

|                           |                                               |
|---------------------------|-----------------------------------------------|
| <i>nomAppareil-Source</i> | étant l'appareil duquel on reçoit les données |
| <i>donnees</i>            | étant les données reçues                      |

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM_PERIPHERIQUE_BLUETOOTH_LANCEUR`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM_PERIPHERIQUE_BLUETOOTH_TABLE`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesTable()`.

```
{
 Log.d("IHMEcranPrincipal", "traiterDonneesRecues()");

 switch (nomAppareilSource)
 {
 case NOM_PERIPHERIQUE_BLUETOOTH_LANCEUR:
 traiterDonneesRecuesLanceur(donnees);
 break;
 case NOM_PERIPHERIQUE_BLUETOOTH_TABLE:
 traiterDonneesRecuesTable(donnees);
 break;
 default:
 break;
 }
}
```

**9.5.2.45** `void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesLanceur ( String donnees ) [private]`

Méthode `traiterDonneesRecuesLanceur()` permettant le traitement des données reçues par le lanceur.

**Paramètres**

|                |                          |
|----------------|--------------------------|
| <i>donnees</i> | étant les données reçues |
|----------------|--------------------------|

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterErreurRecueLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ENTETE`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecues()`.

```
{
 Log.d("IHMEcranPrincipal", "traiterDonneesRecuesLanceur()");

 List<String> donneeRecue = new ArrayList<String>(Arrays.asList(donnees.
split(":")));

 switch (donneeRecue.get(0))
 {
 case TRAME_ENTETE:
 if(donneeRecue.get(1).contains("ERREUR"))
 traiterErreurRecueLanceur(donneeRecue.get(1));
 break;
 default:
 Log.e("IHMEcranPrincipal", "traiterDonneesRecuesLanceur() :
trame non reconnue !");
 }
}
```

**9.5.2.46** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesTable ( String *donnees* )  
[private]

Méthode [traiterDonneesRecuesTable\(\)](#) permettant le traitement des données reçues par la table.

#### Paramètres

|                |                          |
|----------------|--------------------------|
| <i>donnees</i> | étant les données reçues |
|----------------|--------------------------|

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementerBallesJouees\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME\\_ENTETE](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecues\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "traiterDonneesRecuesTable()");

 List<String> donneeRecue = new ArrayList<String>(Arrays.asList(donnees.
split(":")));

 switch (donneeRecue.get(0))
 {
 case TRAME_ENTETE:
 // Transférer la trame reçue à l'écran
 envoyerTramePeripheriqueBluetoothEcran(donnees);

 switch(donneeRecue.get(1))
 {
 case "IMPACT":
 if(Integer.parseInt(donneeRecue.get(2)) != 0) // Si
l'impacte a eu lieu sur une des 9 zones et non sur le côté du lanceur
 calculerReussiteSeance(Integer.parseInt(donneeRecue
.get(2)));
 else
 incrementerBallesJouees();
 break;
 case "FAUTE":
 calculerReussiteSeance(-1);
 }
 break;
 default:
 Log.e("IHMEcranPrincipal", "traiterDonneesRecuesTable() : trame
non reconnue !");
 }
}
```

**9.5.2.47** void com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterErreurRecueLanceur ( String *erreur* )  
[private]

Méthode [traiterErreurRecueLanceur\(\)](#) permettant la gestion des erreurs en provenance du lanceur.

#### Paramètres

|               |                             |
|---------------|-----------------------------|
| <i>erreur</i> | étant le code d'erreur reçu |
|---------------|-----------------------------|

Références [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance\(\)](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesLanceur\(\)](#).

```
{
 Log.d("IHMEcranPrincipal", "traiterErreurRecueLanceur() code d'erreur :
" + erreur);

 pauserSeance();
}
```

#### 9.5.2.48 `boolean com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis()` [private]

Méthode `verifierConnexionAppareilsBluetoothRequis()` vérifiant si les appareils Bluetooth nécessaires au bon déroulement d'une séance sont détectés et connectés, puis modifie l'IHM en conséquent.

##### Renvoie

booléen tous les appareils sont connectés ou non

Références `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actualiserIHMAppareilsBluetooth()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothOperationnel()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothSuffisant()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothEcran`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothTable`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick()`.

```
{
 Log.d("IHMEcranPrincipal", "verifierConnexionAppareilsBluetoothRequis()");

 boolean ecranEstDetecte = (peripheriqueBluetoothEcran!=null);
 boolean lanceurEstDetecte = (peripheriqueBluetoothLanceur != null);
 boolean tableEstDetectee = (peripheriqueBluetoothTable != null);
 boolean ecranEstConnecte = false;
 boolean lanceurEstConnecte = false;
 boolean tableEstConnectee = false;

 if(ecranEstDetecte)
 ecranEstConnecte = peripheriqueBluetoothEcran.estConnecte();

 if(lanceurEstDetecte)
 lanceurEstConnecte = peripheriqueBluetoothLanceur.estConnecte();

 if(tableEstDetectee)
 tableEstConnectee = peripheriqueBluetoothTable.estConnecte();

 actualiserIHMAppareilsBluetooth(tableEstConnectee, lanceurEstConnecte,
 ecranEstConnecte);

 boolean appareilsTousConnectes = ecranEstConnecte && lanceurEstConnecte
 && tableEstConnectee;
 boolean aucunAppareilConnecte = !ecranEstConnecte && !
 lanceurEstConnecte && !tableEstConnectee;

 if(appareilsTousConnectes) // Si tous les appareils sont connectés
 {
 modifierIHMBluetoothOperationnel();
 return true;
 }
 else if(aucunAppareilConnecte) // Si aucun appareil n'est connecté
 {
 modifierIHMBluetoothInsuffisant();
 return false;
 }
 else
 {
 modifierIHMBluetoothSuffisant(); // Si au moins un appareil est
 connecté
 }

 return false;
}
```

### 9.5.3 Documentation des données membres

#### 9.5.3.1 BluetoothAdapter `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.adaptateurBluetooth` = null [private]

Référéncé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth()`.

**9.5.3.2** `int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ballesJouees` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementerBallesJouees()`.

**9.5.3.3** `int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ballesReussies` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`.

**9.5.3.4** `SeekBar com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionEffetBalles` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBarresProgressionIHMUtilisables()`.

**9.5.3.5** `SeekBar com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionFrequenceBalles` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBarresProgressionIHMUtilisables()`.

**9.5.3.6** `SeekBar com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionNombreBalles` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBarresProgressionIHMUtilisables()`.

**9.5.3.7** `SeekBar com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionPuissanceBalles` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBarresProgressionIHMUtilisables()`.

**9.5.3.8** `SeekBar com.ttpa.iris.ttpamobile.IHMEcranPrincipal.barreProgressionRotationLanceur` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBarresProgressionIHMUtilisables()`.

## 9.5.3.9 ImageView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonActionSeance [private]

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementerBallesJouees\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothOperationnel\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothSuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance\(\)](#).

## 9.5.3.10 ImageButton com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonAppliquerNomJoueur [private]

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables\(\)](#).

## 9.5.3.11 ImageView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonArreterSeance [private]

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables\(\)](#).

## 9.5.3.12 ImageButton com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonBluetooth [private]

Ressources IHM

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothOperationnel\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothSuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables\(\)](#).

## 9.5.3.13 ImageButton com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonHistorique [private]

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables\(\)](#).

## 9.5.3.14 ImageButton com.ttpa.iris.ttpamobile.IHMEcranPrincipal.boutonParametres [private]

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothOperationnel\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothSuffisant\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.rendreBoutonsIHMClickables\(\)](#).

## 9.5.3.15 final int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR\_BOUTON\_GRIIS = Color.parseColor("#c0c5c6") [static, private]

## 9.5.3.16 final int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR\_BOUTON\_ORANGE = Color.parseColor("#f7bb31") [static, private]

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothSuffisant\(\)](#).

## 9.5.3.17 final int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR\_BOUTON\_ROUGE = Color.parseColor("#ee5e5e") [static, private]

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothInsuffisant\(\)](#).



**9.5.3.18** `final int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.COULEUR_BOUTON_VERT = Color.parseColor("#5eed7b")` [static, private]

Définitions

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothOperationnel()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierIHMBluetoothSuffisant()`.

**9.5.3.19** `Set<BluetoothDevice> com.ttpa.iris.ttpamobile.IHMEcranPrincipal.devices` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth()`.

**9.5.3.20** `final int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_ARRETEE = 0` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`.

**9.5.3.21** `final int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_DEMARREE = 1` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementerBallesJouees()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance()`.

**9.5.3.22** `final int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ETAT_SEANCE_PAUSE = 2` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance()`.

**9.5.3.23** `int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.etatSeance = ETAT_SEANCE_ARRETEE` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementerBallesJouees()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance()`.

**9.5.3.24** `final Handler com.ttpa.iris.ttpamobile.IHMEcranPrincipal.handler` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothTable()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth()`.

**9.5.3.25** `int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.idZonePrecedente = -1` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

**9.5.3.26** `int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.idZonePrecedenteObjectif = -1` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

9.5.3.27 `int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.idZonePrecedenteRobot = -1` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

9.5.3.28 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM_PERIPHERIQUE_BLUETOOTH_ECRAN = "TTPA-Ecran"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth()`.

9.5.3.29 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM_PERIPHERIQUE_BLUETOOTH_LANCEUR = "TTPA-Lanceur"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecues()`.

9.5.3.30 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.NOM_PERIPHERIQUE_BLUETOOTH_TABLE = "TTPA-Table"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecues()`.

9.5.3.31 `String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.nomJoueur` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ajouterJoueur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick()`.

9.5.3.32 `ParametreSeance com.ttpa.iris.ttpamobile.IHMEcranPrincipal.parametresActuels = new ParametreSeance()`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameConnexionPeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick()`.

9.5.3.33 `PeripheriqueBluetooth com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothEcran = null` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.deconnexionPeripheriquesBluetooth()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis()`.

9.5.3.34 `PeripheriqueBluetooth com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetoothLanceur = null` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.deconnexionPeripheriquesBluetooth()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis()`.

**9.5.3.35 PeripheriqueBluetooth** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriqueBluetooth-Table = null` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothTable()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.deconnexionPeripheriquesBluetooth()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothTable()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetoothRequis()`.

**9.5.3.36 List<PeripheriqueBluetooth>** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.peripheriques` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriquesBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth()`.

**9.5.3.37 final int** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.REQUEST_CODE_ENABLE_BLUETOOTH = 0` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrageBluetooth()`.

**9.5.3.38 Seance** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.seanceEnCours` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementerBallesJouees()`.

**9.5.3.39 final int** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.SELECTION_ZONE_OBJECTIF = 0` [static, private]

Gestion des zones

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

**9.5.3.40 final int** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.SELECTION_ZONE_ROBOT = 1` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actionnerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

**9.5.3.41 ServeurBDD** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.serveurBDD` [private]

Attributs de la classe

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ajouterJoueur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionBaseDeDonnees()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`.

**9.5.3.42 Spinner** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.spinnerListeJoueurs` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`.

**9.5.3.43 String** `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.strZoneObjectif = new String("ZONE 0")` [package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

9.5.3.44 `String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.strZoneRobot = new String("ZONE 0")`  
[package]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

9.5.3.45 `TextView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurEffetBalles` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`.

9.5.3.46 `TextView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurFrequenceBalles`  
[private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`.

9.5.3.47 `TextView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurNombreBalles` [private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`.

9.5.3.48 `TextView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurPuissanceBalles`  
[private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`.

9.5.3.49 `TextView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.texteValeurRotationLanceur`  
[private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.modifierValeursParametresIHM()`.

9.5.3.50 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ECRAN_DEBUT_SEANCE = "START"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothEcran()`.

9.5.3.51 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ECRAN_FIN_SEANCE = "FINSEANCE"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothEcran()`.

9.5.3.52 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ECRAN_PAUSE_SEANCE = "PAUSE"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance()`.

9.5.3.53 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ECRAN_REPRISE_SEANCE = "RESUME"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance()`.

9.5.3.54 `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_ENTETE = "$TTPA"`  
[static, private]

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameConnexionPeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothEcran()`.

`Lanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.traiterDonneesRecuesTable()`.

**9.5.3.55** `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_FIN = "\r\n" [static, private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`.

**9.5.3.56** `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_LANCEUR_ARRET_SEANCE = ":STOP:" + TRAME_FIN [static, private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothLanceur()`.

**9.5.3.57** `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_LANCEUR_PAUSE_SEANCE = ":PAUSE:" + TRAME_FIN [static, private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.pauserSeance()`.

**9.5.3.58** `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_LANCEUR_PING = ":PING:" + TRAME_FIN [static, private]`

**9.5.3.59** `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_LANCEUR_REPRISE_SEANCE = ":RESUME:" + TRAME_FIN [static, private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.reprendreSeance()`.

**9.5.3.60** `final String com.ttpa.iris.ttpamobile.IHMEcranPrincipal.TRAME_TABLE_ARRET_SEANCE = ":RESET" [static, private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothTable()`.

**9.5.3.61** `ImageView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantEcran [private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actualiserIHMAppeareilsBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`.

**9.5.3.62** `ImageView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantLanceur [private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actualiserIHMAppeareilsBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`.

**9.5.3.63** `ImageView com.ttpa.iris.ttpamobile.IHMEcranPrincipal.voyantTable [private]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.actualiserIHMAppeareilsBluetooth()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.affecterMembresIHM()`.

**9.5.3.64** `int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneObjectif = 0 [package]`

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone()`.

9.5.3.65 int com.ttpa.iris.ttpamobile.IHMEcranPrincipal.zoneRobot = 0 [package]

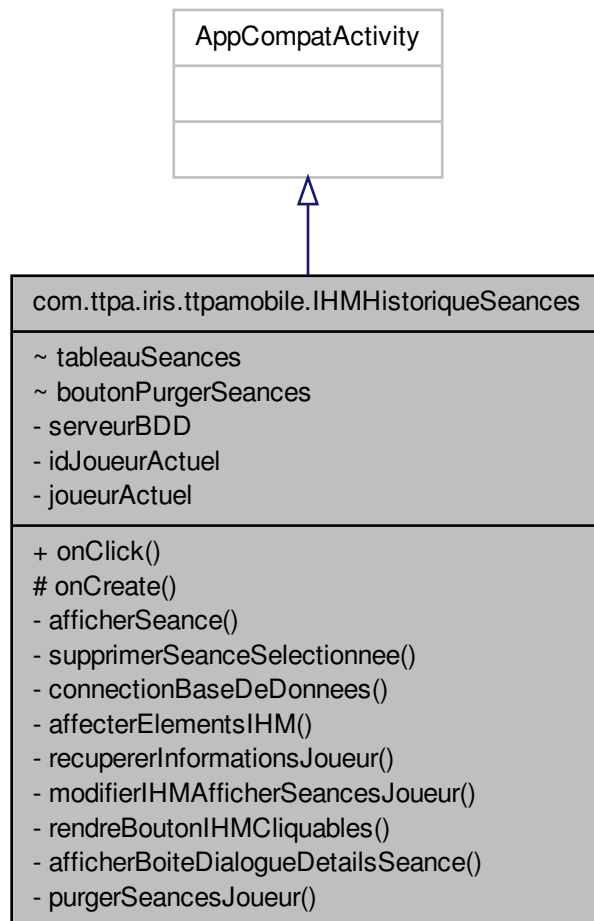
Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.selectionnerZone\(\)](#).

La documentation de cette classe a été générée à partir du fichier suivant :

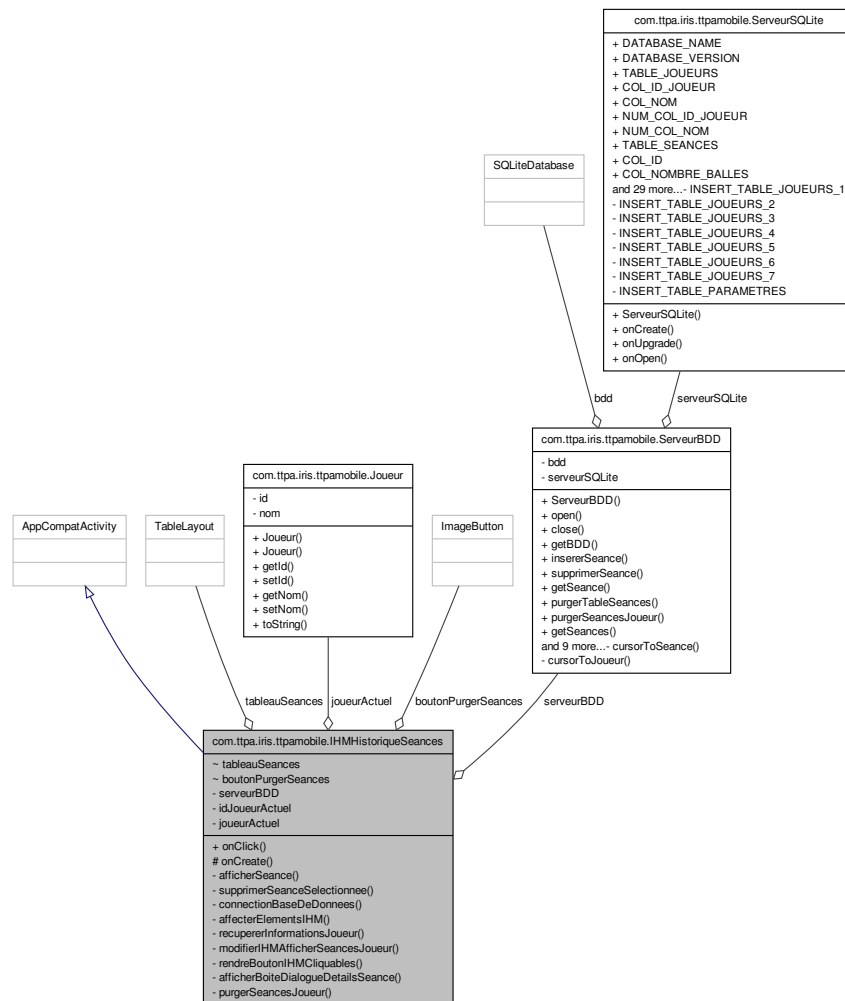
– [IHMEcranPrincipal.java](#)

## 9.6 Référence de la classe com.ttpa.iris.ttpamobile.IHMHistoriqueSeances

Grphe d'héritage de com.ttpa.iris.ttpamobile.IHMHistoriqueSeances :



Graphe de collaboration de com.ttpa.iris.ttpamobile.IHMHistoriqueSeances :



#### Fonctions membres publiques

- void [onClick](#) (View element)

#### Fonctions membres protégées

- void [onCreate](#) (Bundle savedInstanceState)

#### Attributs de paquetage

- `TableLayout` [tableauSeances](#)
- `ImageButton` [boutonPurgerSeances](#)

#### Fonctions membres privées

- void [afficherSeance](#) (`Seance` seance)
- void [supprimerSeanceSelectionnee](#) (int idSeanceElementSelectionne, int indexLigneTableau)

- void `connectionBaseDeDonnees()`
- void `affecterElementsIHM()`
- void `recupererInformationsJoueur()`
- void `modifierIHMAfficherSeancesJoueur()`
- void `rendreBoutonIHMClickables()`
- void `afficherBoiteDialogueDetailsSeance(int idSeanceSelectionnee)`
- void `purgerSeancesJoueur()`

#### Attributs privés

- `ServeurBDD serveurBDD`
- `int idJoueurActuel`
- `Joueur joueurActuel`

#### 9.6.1 Description détaillée

Created by smaniotto on 12/04/18. Classe `IHMHistoriqueSeances` définissant le comportement du layout 'ecran\_historique\_seances'.

#### 9.6.2 Documentation des fonctions membres

##### 9.6.2.1 void `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.affecterElementsIHM()` [private]

Méthode `affecterElementsIHM()` permettant d'affecter les éléments graphiques actuels de l'IHM.

Références `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.boutonPurgerSeances`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.tableauSeances`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onCreate()`.

```
{
 Log.d("IHMHistoriqueSeances", "affecterElementsIHM()");

 // Tableau des séances
 tableauSeances = (TableLayout) findViewById(R.id.tableauSeances);
 // Bouton de purge des séances du joueur
 boutonPurgerSeances = (ImageButton) findViewById(R.id.
boutonPurgerSeances);
}
```

##### 9.6.2.2 void `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherBoiteDialogueDetailsSeance(int idSeanceSelectionnee)` [private]

Méthode `afficherBoiteDialogueDetailsSeance()` affichant une boite de dialogue avec les informations détaillées de la séance sélectionnée.

#### Paramètres

|                              |                                    |
|------------------------------|------------------------------------|
| <i>idSeance-Selectionnee</i> | étant l'id de la séance à afficher |
|------------------------------|------------------------------------|

Références `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.Seance.getZoneObjectif()`, `com.ttpa.iris.ttpamobile.Seance.getZoneRobot()`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onClick()`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.serveurBDD`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onClick()`.

```
{
 Log.d("IHMHistoriqueSeances", "afficherBoiteDialogueDetailsSeance()");

 // Retrouver l'objet Seance correspondant grâce à la base de données
 Seance seanceSelectionnee = serveurBDD.getSeance(idSeanceSelectionnee);
}
```



```

// Afficher la boîte de dialogue
AlertDialog.Builder detailsSeance = new AlertDialog.Builder(this);
LayoutInflater factory = LayoutInflater.from(this);
final View detailsSeanceView = factory.inflate(R.layout.details_seance,
null);
detailsSeance.setView(detailsSeanceView);

detailsSeance.setTitle("Informations complémentaires de la séance");

// Affecter les informations de la séance aux champs correspondants
TextView texteValeurZoneRobot = (TextView) detailsSeanceView.
findViewById(R.id.texteValeurZoneRobot);
TextView texteValeurZoneObjectif = (TextView) detailsSeanceView.
findViewById(R.id.texteValeurZoneObjectif);

switch (seanceSelectionnee.getZoneRobot())
{
 case 0:
 case -1:
 texteValeurZoneRobot.setText("Aucune");
 break;
 default:
 texteValeurZoneRobot.setText("ZONE " + seanceSelectionnee.
getZoneRobot());
 break;
}

switch (seanceSelectionnee.getZoneObjectif())
{
 case 0:
 case -1:
 texteValeurZoneObjectif.setText("Aucune");
 break;
 default:
 texteValeurZoneObjectif.setText("ZONE " + seanceSelectionnee.
getZoneObjectif());
 break;
}

detailsSeance.setNegativeButton("Retour", new DialogInterface.
OnClickListener()
{
 public void onClick(DialogInterface dialog, int which)
 {
 }
});

detailsSeance.show();
}

```

### 9.6.2.3 `void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance ( Seance seance )` [private]

Méthode `afficherSeance()` permettant d'afficher dans un tableau une séance.

#### Paramètres

|               |                            |
|---------------|----------------------------|
| <i>seance</i> | étant la séance à afficher |
|---------------|----------------------------|

Références `com.ttpa.iris.ttpamobile.Seance.getDateDebut()`, `com.ttpa.iris.ttpamobile.Seance.getEffet()`, `com.ttpa.iris.ttpamobile.Seance.getFrequence()`, `com.ttpa.iris.ttpamobile.Seance.getId()`, `com.ttpa.iris.ttpamobile.Seance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.Seance.getTauxReussite()`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.tableauSeances`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur()`.

```

{
 Log.d("IHMHistoriqueSeances", "afficherSeance()");

 TableLayout tableauSeances = (TableLayout) findViewById(R.id.
tableauSeances);

 // Déclaration des paramètres visuels des champs
 TableRow.LayoutParams lpLigne = new TableRow.LayoutParams(TableRow.
LayoutParams.WRAP_CONTENT);

```

```

 TableRow.LayoutParams lpChamp = new TableRow.LayoutParams(TableRow.
 LayoutParams.MATCH_PARENT, TableRow.LayoutParams.MATCH_PARENT, 1f);

 // Déclaration et paramétrage de la ligne accueillant la séance
 TableRow ligne= new TableRow(this);
 ligne.setLayoutParams(lpChamp);
 float taille_police_ecriture = (float)Integer.parseInt(getString(R.
 string.taille_police_historique));

 // Déclaration et paramétrage du champ ID
 TextView texteSeanceId = new TextView(this);
 texteSeanceId.setLayoutParams(lpChamp);
 texteSeanceId.setGravity(Gravity.CENTER);
 texteSeanceId.setVisibility(View.GONE);
 texteSeanceId.setText(Integer.toString(seance.getId()));

 // Déclaration et paramétrage du champ Date
 TextView texteSeanceDate = new TextView(this);
 texteSeanceDate.setLayoutParams(lpChamp);
 texteSeanceDate.setGravity(Gravity.CENTER);
 texteSeanceDate.setTextSize(taille_police_ecriture);
 texteSeanceDate.setText(seance.getDateDebut());

 // Déclaration et paramétrage du champ Fréquence
 TextView texteSeanceFrequence = new TextView(this);
 texteSeanceFrequence.setLayoutParams(lpChamp);
 texteSeanceFrequence.setGravity(Gravity.CENTER);
 texteSeanceFrequence.setTextSize(taille_police_ecriture);
 texteSeanceFrequence.setText(Integer.toString(seance.getFrequence()) +
 " balles/min");

 // Déclaration et paramétrage du champ Nb Balles
 TextView texteSeanceNombreBalles = new TextView(this);
 texteSeanceNombreBalles.setLayoutParams(lpChamp);
 texteSeanceNombreBalles.setGravity(Gravity.CENTER);
 texteSeanceNombreBalles.setTextSize(taille_police_ecriture);
 texteSeanceNombreBalles.setText(Integer.toString(seance.getNombreBalles
 ()));

 // Déclaration et paramétrage du champ Effet
 TextView texteSeanceEffet = new TextView(this);
 texteSeanceEffet.setLayoutParams(lpChamp);
 texteSeanceEffet.setGravity(Gravity.CENTER);
 texteSeanceEffet.setTextSize(taille_police_ecriture);
 texteSeanceEffet.setText(seance.getEffet());

 // Déclaration et paramétrage du champ Taux Réussite
 TextView texteSeanceTauxReussite = new TextView(this);
 texteSeanceTauxReussite.setLayoutParams(lpChamp);
 texteSeanceTauxReussite.setGravity(Gravity.CENTER);
 texteSeanceTauxReussite.setTextSize(taille_police_ecriture);
 texteSeanceTauxReussite.setText(Float.toString(seance.getTauxReussite()
) + "%");

 // Déclaration et paramétrage du champ Action
 ImageButton boutonDetailsSeance = new ImageButton(this);
 boutonDetailsSeance.setLayoutParams(lpChamp);

 boutonDetailsSeance.setBackgroundColor(Color.argb(0, 0, 0, 0)); //
 Rendre le fond transparent
 boutonDetailsSeance.setImageResource(R.drawable.ic_loupe_afficher);
 boutonDetailsSeance.setOnClickListener(this); // Rendre le bouton
 cliquable

 // Ajouter les champs à la ligne
 ligne.addView(texteSeanceId);
 ligne.addView(texteSeanceDate);
 ligne.addView(texteSeanceFrequence);
 ligne.addView(texteSeanceNombreBalles);
 ligne.addView(texteSeanceEffet);
 ligne.addView(texteSeanceTauxReussite);
 ligne.addView(boutonDetailsSeance);

 // Ajouter la ligne au tableau des séances à l'index 1 : après l'entête
 du tableau
 tableauSeances.addView(ligne, 1);
 }

```

#### 9.6.2.4 void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.connectionBaseDeDonnees ( ) [private]

Méthode `connectionBaseDeDonnees()` permettant de se connecter à la base de données.

Références `com.ttpa.iris.ttpamobile.ServeurBDD.open()`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.serveurBDD`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onCreate()`.

```
{
 Log.d("IHMHistoriqueSeances", "connectionBaseDeDonnees()");

 serveurBDD = new ServeurBDD(this);
 serveurBDD.open();
}
```

#### 9.6.2.5 void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur ( ) [private]

Méthode `modifierIHMAfficherSeancesJoueur()` affichant les séances du joueur actuel dans un tableau de l'IHM.

Références `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance()`, `com.ttpa.iris.ttpamobile.Joueur.getId()`, `com.ttpa.iris.ttpamobile.Joueur.getNom()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.joueurActuel`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.serveurBDD`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onCreate()`.

```
{
 Log.d("IHMHistoriqueSeances", "modifierIHMAfficherSeancesJoueur()");

 TextView texteHistoriqueSeances = (TextView) findViewById(R.id.
 texteHistoriqueSeances);
 texteHistoriqueSeances.setText("Historique des séances de " +
 joueurActuel.getNom() + " :");

 // Récupérer la liste des séances existantes du joueur
 List<Seance> seances = serveurBDD.getSeances(joueurActuel.getId());
 Log.d("IHMHistoriqueSeances", "modifierIHMAfficherSeancesJoueur()
 Nombre de séances du joueur: " + seances.size());
 // Pour chaque séance existante
 for(int i = 0; i < seances.size(); i++)
 {
 Log.d("IHMHistoriqueSeances", "modifierIHMAfficherSeancesJoueur()
 Séance n° " + seances.get(i).getId() + ": \n" + seances.get(i).toString());

 // Ajouter la séance dans le tableau de séances
 afficherSeance(seances.get(i));
 }
}
```

#### 9.6.2.6 void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onClick ( View element )

Méthode `onClick()` permettant la gestion du click sur un bouton.

Paramètres

|                |                                                   |
|----------------|---------------------------------------------------|
| <i>element</i> | étant l'élément sur lequel l'utilisateur a cliqué |
|----------------|---------------------------------------------------|

Références `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherBoiteDialogueDetailsSeance()`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.boutonPurgerSeances`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur()`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.tableauSeances`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherBoiteDialogueDetailsSeance()`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur()`.

```

{
 Log.d("IHMHistoriqueSeances", "onClick()");

 Seance seance = null;

 if (element == boutonPurgerSeances)
 {
 // Purger les séances déjà existantes du joueur
 purgerSeancesJoueur();
 }
 else // l'élément est un bouton du champ action
 {
 // Retrouver l'ID de la séance sélectionnée
 for (int i = 1; i < tableauSeances.getChildCount(); ++i) {

 TableRow ligne = (TableRow) tableauSeances.getChildAt(i);

 if (ligne == element.getParent()) {
 TextView texteIdSeance = (TextView) ligne.getChildAt(0); //
 0 : Position du champ ID de la ligne
 int idSeanceElementSelectionne = Integer.parseInt(
 texteIdSeance.getText().toString()); // ID de la séance à supprimer

 // Afficher la boîte de dialogue de la séance sélectionnée
 afficherBoiteDialogueDetailsSeance(
 idSeanceElementSelectionne);
 }
 }
 }
}

```

#### 9.6.2.7 void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onCreate ( Bundle savedInstanceState ) [protected]

Méthode onCreate appelée au démarrage de l'activité permettant l'initialisation des composants.

##### Paramètres

|                           |  |
|---------------------------|--|
| <i>savedInstanceState</i> |  |
|---------------------------|--|

Références [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.affectedElementsIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.connectionBaseDeDonnees\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.recupererInformationsJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.rendreBoutonIHMClickables\(\)](#).

```

{
 super.onCreate(savedInstanceState);
 setContentView(R.layout.ecran_historique_seances);

 // Se connecter à la base de données
 connectionBaseDeDonnees();

 // Affecter les éléments graphiques de l'IHM
 affecterElementsIHM();

 // Récupérer les informations du joueur actuel
 recupererInformationsJoueur();

 // Modifier l'IHM pour afficher le joueur
 modifierIHMAfficherSeancesJoueur();

 // Permettre à tous les boutons de suppression de séance d'être
 cliquables
 rendreBoutonIHMClickables();
}

```

#### 9.6.2.8 void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur ( ) [private]

Méthode [purgerSeancesJoueur\(\)](#) affichant une boîte de dialogue qui va supprimer les séances du joueur actuel selon la décision prise.

Références `com.ttpa.iris.ttpamobile.Joueur.getNom()`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.idJoueurActuel`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.joueurActuel`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onClick()`, `com.ttpa.iris.ttpamobile.ServeurBDD.purgerSeancesJoueur()`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.serveurBDD`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.tableauSeances`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onClick()`.

```
{
 Log.d("IHMHistoriqueSeances", "purgerSeancesJoueur()");

 // Afficher la boîte de dialogue
 AlertDialog.Builder detailsSeance = new AlertDialog.Builder(this);

 detailsSeance.setMessage("Vous êtes sur le point de supprimer définitivement la séance du joueur " + joueurActuel.getNom() + ".");

 detailsSeance.setPositiveButton("Continuer", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int which)
 {
 // Purger les séances déjà existantes du joueur
 Toast.makeText(getApplicationContext(), "Purge des séances...", Toast.LENGTH_SHORT).show();

 // Purger la table des séances de la base de données
 serveurBDD.purgerSeancesJoueur(idJoueurActuel);

 // Supprimer le contenu du tableau des séances
 int compteur = tableauSeances.getChildCount();
 for (int i = 1; i < compteur; i++) { // On ne supprime pas la 1ere ligne qui est l'entête
 View child = tableauSeances.getChildAt(i);
 if (child instanceof TableRow) ((ViewGroup) child).removeAllViews();
 }
 }
 });

 detailsSeance.setNegativeButton("Annuler", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int which)
 {
 }
 });

 detailsSeance.show();
}
```

#### 9.6.2.9 `void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.recupererInformationsJoueur ( )` [private]

Méthode `recupererInformationsJoueur()` affectant aux attributs les caractéristiques du joueur actuel, importées depuis la base de données.

Références `com.ttpa.iris.ttpamobile.ServeurBDD.getIdJoueurParametres()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueur()`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.idJoueurActuel`, `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.joueurActuel`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.serveurBDD`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onCreate()`.

```
{
 Log.d("IHMHistoriqueSeances", "recupererInformationsJoueur()");

 idJoueurActuel = serveurBDD.getIdJoueurParametres();
 joueurActuel = serveurBDD.getJoueur(idJoueurActuel);
}
```

#### 9.6.2.10 `void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.rendreBoutonIHMClickables ( )` [private]

Méthode `rendreBoutonIHMClickables()` permettant aux boutons de l'IHM d'être cliquables.

Références `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.boutonPurgerSeances`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.tableauSeances`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onCreate()`.

```
{
 Log.d("IHMHistoriqueSeances", "rendreBoutonIHMClickables()");

 // Boutons présents dans le tableau d'historique de séances du joueur
 int compteurLignes = tableauSeances.getChildCount();
 for (int i = 0; i < compteurLignes; ++i) {
 TableRow ligne = (TableRow) tableauSeances.getChildAt(i);

 if (ligne.getChildAt(ligne.getChildCount() - 1) instanceof
 ImageButton) {
 View element = ligne.getChildAt(ligne.getChildCount() - 1); //
 Le bouton de suppression de séance étant le dernier élément de la ligne
 element.setOnClickListener(this);
 }

 // Bouton de purge de séance du joueur
 boutonPurgerSeances.setOnClickListener(this);
 }
}
```

#### 9.6.2.11 `void com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.supprimerSeanceSelectionnee ( int idSeanceElementSelectionnee, int indexLigneTableau )` [private]

Méthode `supprimerSeanceSelectionnee()` permettant la suppression d'une séance précise.

Paramètres

|                                      |                                     |
|--------------------------------------|-------------------------------------|
| <i>idSeance-Element-Selectionnee</i> | étant l'ID de la séance à supprimer |
| <i>indexLigne-Tableau</i>            | étant son index dans le tableau     |

Références `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.serveurBDD`, `com.ttpa.iris.ttpamobile.ServeurBDD.supprimerSeance()`, et `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.tableauSeances`.

```
{
 Log.d("IHMHistoriqueSeances", "supprimerSeanceSelectionnee()");

 // Supprimer la séance sélectionnée de la base de données
 serveurBDD.supprimerSeance(idSeanceElementSelectionnee);

 // Supprimer le contenu de la ligne correspondante du tableau des
 séances
 TableLayout tableauSeances = (TableLayout) findViewById(R.id.
 tableauSeances);
 TableRow ligne = (TableRow) tableauSeances.getChildAt(indexLigneTableau
);
 tableauSeances.removeView(ligne);

 Toast.makeText(getApplicationContext(), "Suppression de la séance " +
 idSeanceElementSelectionnee + "...", Toast.LENGTH_SHORT).show();
}
```

### 9.6.3 Documentation des données membres

#### 9.6.3.1 ImageButton com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.boutonPurgerSeances [package]

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.affecterElementsIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onClick\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.rendreBoutonIHMClickables\(\)](#).

#### 9.6.3.2 int com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.idJoueurActuel [private]

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.recupererInformationsJoueur\(\)](#).

#### 9.6.3.3 Joueur com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.joueurActuel [private]

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.recupererInformationsJoueur\(\)](#).

#### 9.6.3.4 ServeurBDD com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.serveurBDD [private]

Attributs de la classe

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherBoiteDialogueDetailsSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.connectionBaseDeDonnees\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.recupererInformationsJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.supprimerSeanceSelectionnee\(\)](#).

#### 9.6.3.5 TableLayout com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.tableauSeances [package]

Éléments graphiques de l'IHM

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.affecterElementsIHM\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.onClick\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.rendreBoutonIHMClickables\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.supprimerSeanceSelectionnee\(\)](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- [IHMHistoriqueSeances.java](#)

## 9.7 Référence de la classe com.ttpa.iris.ttpamobile.Joueur

Fonctions membres publiques

- [Joueur \(\)](#)
- [Joueur \(String nom\)](#)
- [int getId \(\)](#)
- [void setId \(int id\)](#)
- [String getNom \(\)](#)
- [void setNom \(String nom\)](#)
- [String toString \(\)](#)

Attributs privés

- [int id](#)
- [String nom](#)

## 9.7.1 Description détaillée

Created by smaniotto on 10/04/18.

## 9.7.2 Documentation des constructeurs et destructeur

9.7.2.1 `com.ttpa.iris.ttpamobile.Joueur.Joueur ( )`

```
{
}
```

9.7.2.2 `com.ttpa.iris.ttpamobile.Joueur.Joueur ( String nom )`

Références [com.ttpa.iris.ttpamobile.Joueur.nom](#).

```
{
 this.nom = nom;
}
```

## 9.7.3 Documentation des fonctions membres

9.7.3.1 `int com.ttpa.iris.ttpamobile.Joueur.getId ( )`

Références [com.ttpa.iris.ttpamobile.Joueur.id](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur\(\)](#).

```
{
 return id;
}
```

9.7.3.2 `String com.ttpa.iris.ttpamobile.Joueur.getNom ( )`

Références [com.ttpa.iris.ttpamobile.Joueur.nom](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.insererJoueur\(\)](#), [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur\(\)](#).

```
{
 return nom;
}
```

9.7.3.3 `void com.ttpa.iris.ttpamobile.Joueur.setId ( int id )`

Références [com.ttpa.iris.ttpamobile.Joueur.id](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToJoueur\(\)](#).

```
{
 this.id = id;
}
```

9.7.3.4 `void com.ttpa.iris.ttpamobile.Joueur.setNom ( String nom )`

Références [com.ttpa.iris.ttpamobile.Joueur.nom](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToJoueur\(\)](#).



```

{
 this.nom = nom;
}

```

### 9.7.3.5 String com.ttpa.iris.ttpamobile.Joueur.toString ( )

Méthode toString permettant la visualisation des caractéristiques d'un joueur.

#### Renvoie

String les caractéristiques d'un joueur.

Références [com.ttpa.iris.ttpamobile.Joueur.nom](#).

```

{
 return "Nom : " + nom;
}

```

### 9.7.4 Documentation des données membres

#### 9.7.4.1 int com.ttpa.iris.ttpamobile.Joueur.id [private]

Référencé par [com.ttpa.iris.ttpamobile.Joueur.getId\(\)](#), et [com.ttpa.iris.ttpamobile.Joueur.setId\(\)](#).

#### 9.7.4.2 String com.ttpa.iris.ttpamobile.Joueur.nom [private]

Référencé par [com.ttpa.iris.ttpamobile.Joueur.getNom\(\)](#), [com.ttpa.iris.ttpamobile.Joueur.Joueur\(\)](#), [com.ttpa.iris.ttpamobile.Joueur.setNom\(\)](#), et [com.ttpa.iris.ttpamobile.Joueur.toString\(\)](#).

La documentation de cette classe a été générée à partir du fichier suivant :

– [Joueur.java](#)

## 9.8 Référence de la classe com.ttpa.iris.ttpamobile.ParametreSeance

### Fonctions membres publiques

- [ParametreSeance \(\)](#)
- [ParametreSeance \(String nomJoueur, String effetBalles, int intensiteEffet, int puissanceBalles, int frequenceBalles, int rotation, int nombreBalles\)](#)
- [String getNomJoueur \(\)](#)
- [void setNomJoueur \(String nomJoueur\)](#)
- [int getFrequenceBalles \(\)](#)
- [void setFrequenceBalles \(int frequenceBalles\)](#)
- [int getNombreBalles \(\)](#)
- [void setNombreBalles \(int nombreBalles\)](#)
- [String getEffetBalles \(\)](#)
- [void setEffetBalles \(String effetBalles\)](#)
- [void setIntensiteEffet \(int intensiteEffet\)](#)
- [int getPuissanceBalles \(\)](#)
- [void setPuissanceBalles \(int puissanceBalles\)](#)
- [int getRotation \(\)](#)
- [void setRotation \(int rotation\)](#)
- [String toString \(\)](#)
- [boolean estValide \(\)](#)
- [final boolean frequenceEstValide \(int frequence\)](#)
- [final boolean nombreBallesEstValide \(int nombreBalles\)](#)
- [final boolean puissanceBallesEstValide \(int puissanceBalles\)](#)
- [final boolean rotationEstValide \(int rotation\)](#)
- [final boolean effetEstValide \(String effet\)](#)

### Attributs publics statiques

- [static final int MIN\\_FREQUENCE\\_PARAMETRE = 1](#)

- static final int [MAX\\_FREQUENCE\\_PARAMETER](#) = 60
- static final int [MIN\\_NOMBRE\\_BALLES\\_PARAMETER](#) = 1
- static final int [MAX\\_NOMBRE\\_BALLES\\_PARAMETER](#) = 50
- static final int [MIN\\_PUISSANCE\\_BALLES\\_PARAMETER](#) = 1
- static final int [MAX\\_PUISSANCE\\_BALLES\\_PARAMETER](#) = 10
- static final int [MIN\\_ROTATION\\_PARAMETER](#) = 0
- static final int [MAX\\_ROTATION\\_PARAMETER](#) = 180

#### Fonctions de paquetage

- final char [getEffet](#) ()
- final String [getEffetComplet](#) ()
- final int [getIntensiteEffet](#) ()

#### Attributs privés

- String [nomJoueur](#)
- String [effetBalles](#)
- int [intensiteEffet](#)
- int [puissanceBalles](#)
- int [frequenceBalles](#)
- int [rotation](#)
- int [nombreBalles](#)

#### 9.8.1 Description détaillée

Created by smaniotto on 19/03/18. Classe [ParametreSeance](#) définissant les caractéristiques et le comportement d'un paramètre de séance.

#### 9.8.2 Documentation des constructeurs et destructeur

##### 9.8.2.1 [com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance](#) ( )

Méthode [ParametreSeance](#) constructeur par défaut de la classe [ParametreSeance](#).

```
{ }
```

##### 9.8.2.2 [com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance](#) ( String *nomJoueur*, String *effetBalles*, int *intensiteEffet*, int *puissanceBalles*, int *frequenceBalles*, int *rotation*, int *nombreBalles* )

Méthode [ParametreSeance](#) constructeur de la classe [ParametreSeance](#).

#### Paramètres

|                         |                                                                             |
|-------------------------|-----------------------------------------------------------------------------|
| <i>nomJoueur</i>        | étant le nom du joueur pratiquant la séance.                                |
| <i>effetBalles</i>      | étant l'effet appliqué aux balles tout au long de la séance.                |
| <i>intensiteEffet</i>   | étant l'intensité de l'effet appliqué aux balles tout au long de la séance. |
| <i>frequence-Balles</i> | étant la fréquence d'envoi des balles (en balles/minute) de la séance.      |
| <i>puissance-Balles</i> | étant la puissance ou coefficient de vitesse balles à envoyer.              |
| <i>rotation</i>         | étant la rotation du lanceur en degrés.                                     |
| <i>nombreBalles</i>     | étant le nombre de balles à envoyer.                                        |

Références [com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.frequenceBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.intensiteEffet](#), [com.ttpa.iris.ttpamobile.ParametreSeance.nombreBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.nomJoueur](#), [com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBalles](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.rotation](#).

```

{
 this.nomJoueur = nomJoueur;
 this.effetBalles = effetBalles;
 this.intensiteEffet = intensiteEffet;
 this.puissanceBalles = puissanceBalles;
 this.frequenceBalles = frequenceBalles;
 this.rotation = rotation;
 this.nombreBalles = nombreBalles;
}

```

### 9.8.3 Documentation des fonctions membres

#### 9.8.3.1 final boolean com.ttpa.iris.ttpamobile.ParametreSeance.effetEstValide ( String effet )

Méthode effetEstValide vérifiant la validité de l'effet du paramétrage.

##### Paramètres

|              |                           |
|--------------|---------------------------|
| <i>effet</i> | étant l'effet à vérifier. |
|--------------|---------------------------|

##### Renvoie

Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#).

```

{
 switch (effet)
 {
 case "Lifté":
 return true;
 case "Aucun":
 return true;
 case "Coupé":
 return true;
 default:
 return false;
 }
}

```

#### 9.8.3.2 boolean com.ttpa.iris.ttpamobile.ParametreSeance.estValide ( )

Méthode estValide vérifiant la validité des caractéristiques du paramétrage.

##### Renvoie

Références [com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.effetEstValide\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.frequenceBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.frequenceEstValide\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.nombreBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.nombreBallesEstValide\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBallesEstValide\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.rotation](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.rotationEstValide\(\)](#).

```

{
 return (frequenceEstValide(this.frequenceBalles) &&
 nombreBallesEstValide(this.nombreBalles) && effetEstValide(this.effetBalles) &&
 puissanceBallesEstValide(this.puissanceBalles) && rotationEstValide(this.rotation));
}

```

#### 9.8.3.3 final boolean com.ttpa.iris.ttpamobile.ParametreSeance.frequenceEstValide ( int frequence )

Méthode frequenceEstValide vérifiant la validité de la fréquence du paramétrage.

**Renvoie**

Références [com.ttpa.iris.ttpamobile.ParametreSeance.MAX\\_FREQUENCE\\_PARAMETRE](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.MIN\\_FREQUENCE\\_PARAMETRE](#).

Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#).

```
{
 return (frequence >= MIN_FREQUENCE_PARAMETRE && frequence <=
MAX_FREQUENCE_PARAMETRE);
}
```

**9.8.3.4 final char com.ttpa.iris.ttpamobile.ParametreSeance.getEffet ( ) [package]**

Méthode getEffet vérifiant la validité de l'effet du paramétrage.

**Renvoie**

char l'effet pour la trame

Références [com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur\(\)](#).

```
{
 switch (this.effetBalles)
 {
 case "Lifté":
 return 'L';
 case "Aucun":
 return 'S';
 case "Coupé":
 return 'C';
 default:
 return 'S';
 }
}
```

**9.8.3.5 String com.ttpa.iris.ttpamobile.ParametreSeance.getEffetBalles ( )**

Méthode [getEffetBalles\(\)](#) accesseur de l'attribut effetBalles.

**Renvoie**

effetBalles

Références [com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles](#).

```
{ return effetBalles; }
```

**9.8.3.6 final String com.ttpa.iris.ttpamobile.ParametreSeance.getEffetCompleet ( ) [package]**

Méthode getEffetCompleet accesseur de l'attribut effetBalles.

**Renvoie**

String l'effet pour la trame

Références [com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#).

```
{ return this.effetBalles; }
```

**9.8.3.7** `int com.ttpa.iris.ttpamobile.ParametreSeance.getFrequenceBalles ( )`

Méthode `getFrequenceBalles()` accesseur de l'attribut `frequenceBalles`.

Renvoie

`frequenceBalles`

Références `com.ttpa.iris.ttpamobile.ParametreSeance.frequenceBalles`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`.

```
{
 return frequenceBalles;
}
```

**9.8.3.8** `final int com.ttpa.iris.ttpamobile.ParametreSeance.getIntensiteEffet ( )` [package]

Méthode `getIntensiteEffet` accesseur de l'attribut `intensiteEffet`.

Renvoie

`intensiteEffet` l'effet pour la trame

Références `com.ttpa.iris.ttpamobile.ParametreSeance.intensiteEffet`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`.

```
{ return this.intensiteEffet; }
```

**9.8.3.9** `int com.ttpa.iris.ttpamobile.ParametreSeance.getNombreBalles ( )`

Méthode `getNombreBalles()` accesseur de l'attribut `nombreBalles`.

Renvoie

`nombreBalles`

Références `com.ttpa.iris.ttpamobile.ParametreSeance.nombreBalles`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur()`, `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameParametrageSeancePeripheriqueBluetoothEcran()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur()`.

```
{
 return nombreBalles;
}
```

**9.8.3.10** `String com.ttpa.iris.ttpamobile.ParametreSeance.getNomJoueur ( )`

Méthode `getNomJoueur()` accesseur de l'attribut `nomJoueur`.

**Renvoie**`nomJoueur`Références [com.ttpa.iris.ttpamobile.ParametreSeance.nomJoueur](#).Référéncé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameConnexionPeripheriqueBluetooth-Ecran\(\)](#).

```
{ return nomJoueur; }
```

**9.8.3.11 `int com.ttpa.iris.ttpamobile.ParametreSeance.getPuissanceBalles ( )`**Références [com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBalles](#).Référéncé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur\(\)](#).

```
{
 return puissanceBalles;
}
```

**9.8.3.12 `int com.ttpa.iris.ttpamobile.ParametreSeance.getRotation ( )`**Références [com.ttpa.iris.ttpamobile.ParametreSeance.rotation](#).Référéncé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur\(\)](#).

```
{
 return rotation;
}
```

**9.8.3.13 `final boolean com.ttpa.iris.ttpamobile.ParametreSeance.nombreBallesEstValide ( int nombreBalles )`**Méthode `nombreBallesEstValide` vérifiant la validité du nombre de balles du paramétrage.**Paramètres**

|                     |                                       |
|---------------------|---------------------------------------|
| <i>nombreBalles</i> | étant le nombre de balles à vérifier. |
|---------------------|---------------------------------------|

**Renvoie**Références [com.ttpa.iris.ttpamobile.ParametreSeance.MAX\\_NOMBRE\\_BALLES\\_PARAMETRE](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.MIN\\_NOMBRE\\_BALLES\\_PARAMETRE](#).Référéncé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#).

```
{
 return (nombreBalles >= MIN_NOMBRE_BALLES_PARAMETRE && nombreBalles <=
MAX_NOMBRE_BALLES_PARAMETRE);
}
```

**9.8.3.14 `final boolean com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBallesEstValide ( int puissanceBalles )`**Méthode `puissanceBallesEstValide` vérifiant la validité de la puissance des balles.

## Paramètres

|                         |                    |
|-------------------------|--------------------|
| <i>puissance-Balles</i> | étant la puissance |
|-------------------------|--------------------|

## Renvoi

Références [com.ttpa.iris.ttpamobile.ParametreSeance.MAX\\_PUISSANCE\\_BALLES\\_PARAMETER](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.MIN\\_PUISSANCE\\_BALLES\\_PARAMETER](#).

Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#).

```
{
 return (puissanceBalles >= MIN_PUISSANCE_BALLES_PARAMETER &&
 puissanceBalles <= MAX_PUISSANCE_BALLES_PARAMETER);
}
```

9.8.3.15 final boolean `com.ttpa.iris.ttpamobile.ParametreSeance.rotationEstValide ( int rotation )`

Méthode `rotationEstValide` vérifiant la validité de la rotation.

## Paramètres

|                 |                   |
|-----------------|-------------------|
| <i>rotation</i> | étant la rotation |
|-----------------|-------------------|

## Renvoi

Références [com.ttpa.iris.ttpamobile.ParametreSeance.MAX\\_ROTATION\\_PARAMETER](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.MIN\\_ROTATION\\_PARAMETER](#).

Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#).

```
{
 return (rotation >= MIN_ROTATION_PARAMETER && rotation <=
 MAX_ROTATION_PARAMETER);
}
```

9.8.3.16 void `com.ttpa.iris.ttpamobile.ParametreSeance.setEffetBalles ( String effetBalles )`

Méthode `setEffet` mutateur de l'attribut `effetBalles`.

## Paramètres

|                    |                           |
|--------------------|---------------------------|
| <i>effetBalles</i> | étant l'effet à affecter. |
|--------------------|---------------------------|

Références [com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance\(\)](#).

```
{
 this.effetBalles = effetBalles;
}
```

9.8.3.17 void `com.ttpa.iris.ttpamobile.ParametreSeance.setFrequenceBalles ( int frequenceBalles )`

Méthode `setFrequenceBalles` mutateur de l'attribut `frequenceBalles`.

## Paramètres

|                         |                                |
|-------------------------|--------------------------------|
| <i>frequence-Balles</i> | étant la fréquence à affecter. |
|-------------------------|--------------------------------|

Références [com.ttpa.iris.ttpamobile.ParametreSeance.frequenceBalles](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance\(\)](#).

```
{
 this.frequenceBalles = frequenceBalles;
}
```

**9.8.3.18 void `com.ttpa.iris.ttpamobile.ParametreSeance.setIntensiteEffet` ( int *intensiteEffet* )**

Méthode `setIntensiteEffet` mutateur de l'attribut `intensiteEffet`.

## Paramètres

|                       |                                          |
|-----------------------|------------------------------------------|
| <i>intensiteEffet</i> | étant l'intensité de l'effet à affecter. |
|-----------------------|------------------------------------------|

Références [com.ttpa.iris.ttpamobile.ParametreSeance.intensiteEffet](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance\(\)](#).

```
{
 this.intensiteEffet = intensiteEffet;
}
```

**9.8.3.19 void `com.ttpa.iris.ttpamobile.ParametreSeance.setNombreBalles` ( int *nombreBalles* )**

Méthode `setNombreBalles()` mutateur de l'attribut `nombreBalles`.

## Paramètres

|                     |                                       |
|---------------------|---------------------------------------|
| <i>nombreBalles</i> | étant le nombre de balles à affecter. |
|---------------------|---------------------------------------|

Références [com.ttpa.iris.ttpamobile.ParametreSeance.nombreBalles](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance\(\)](#).

```
{
 this.nombreBalles = nombreBalles;
}
```

**9.8.3.20 void `com.ttpa.iris.ttpamobile.ParametreSeance.setNomJoueur` ( String *nomJoueur* )**

Méthode `setNomJoueur` mutateur de l'attribut `nomJoueur`.

## Paramètres

|                  |                         |
|------------------|-------------------------|
| <i>nomJoueur</i> | étant le nom à affecter |
|------------------|-------------------------|

Références [com.ttpa.iris.ttpamobile.ParametreSeance.nomJoueur](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.onClick\(\)](#).

```
{ this.nomJoueur = nomJoueur; }
```



**9.8.3.21** `void com.ttpa.iris.ttpamobile.ParametreSeance.setPuissanceBalles ( int puissanceBalles )`

Références [com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBalles](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance\(\)](#).

```
{
 this.puissanceBalles = puissanceBalles;
}
```

**9.8.3.22** `void com.ttpa.iris.ttpamobile.ParametreSeance.setRotation ( int rotation )`

Références [com.ttpa.iris.ttpamobile.ParametreSeance.rotation](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.appliquerParametresSeance\(\)](#).

```
{
 this.rotation = rotation;
}
```

**9.8.3.23** `String com.ttpa.iris.ttpamobile.ParametreSeance.toString ( )`

Méthode `toString` permettant la visualisation des caractéristiques du paramétrage.

**Renvoie**

les caractéristiques du paramétrage.

Références [com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.frequenceBalles](#), [com.ttpa.iris.ttpamobile.ParametreSeance.intensiteEffet](#), [com.ttpa.iris.ttpamobile.ParametreSeance.nombreBalles](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBalles](#).

```
{
 return "Frequence : " + frequenceBalles + "\nNombre balles : " +
 nombreBalles + "\nEffet : " + effetBalles + "\nIntensité Effet : " +
 intensiteEffet + "\nPuissance balles : " + puissanceBalles;
}
```

**9.8.4** Documentation des données membres**9.8.4.1** `String com.ttpa.iris.ttpamobile.ParametreSeance.effetBalles` `[private]`

Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.getEffet\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.getEffetBalles\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.getEffetComplet\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.setEffetBalles\(\)](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.toString\(\)](#).

**9.8.4.2** `int com.ttpa.iris.ttpamobile.ParametreSeance.frequenceBalles` `[private]`

Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.getFrequenceBalles\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.setFrequenceBalles\(\)](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.toString\(\)](#).

**9.8.4.3** `int com.ttpa.iris.ttpamobile.ParametreSeance.intensiteEffet` `[private]`

Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.getIntensiteEffet\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.setIntensiteEffet\(\)](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.toString\(\)](#).

9.8.4.4 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MAX_FREQUENCY_PARAMETER = 60`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.frequenceEstValide()`.

9.8.4.5 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MAX_NOMBRE_BALLES_PARAMETER = 50`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.nombreBallesEstValide()`.

9.8.4.6 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MAX_PUISSANCE_BALLES_PARAMETER = 10`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBallesEstValide()`.

9.8.4.7 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MAX_ROTATION_PARAMETER = 180`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.rotationEstValide()`.

9.8.4.8 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MIN_FREQUENCY_PARAMETER = 1`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.frequenceEstValide()`.

9.8.4.9 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MIN_NOMBRE_BALLES_PARAMETER = 1`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.nombreBallesEstValide()`.

9.8.4.10 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MIN_PUISSANCE_BALLES_PARAMETER = 1`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBallesEstValide()`.

9.8.4.11 `final int com.ttpa.iris.ttpamobile.ParametreSeance.MIN_ROTATION_PARAMETER = 0`  
`[static]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.rotationEstValide()`.

9.8.4.12 `int com.ttpa.iris.ttpamobile.ParametreSeance.nombreBalles` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.estValide()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setNombreBalles()`, et `com.ttpa.iris.ttpamobile.ParametreSeance.toString()`.

9.8.4.13 `String com.ttpa.iris.ttpamobile.ParametreSeance.nomJoueur` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.getNomJoueur()`, `com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance()`, et `com.ttpa.iris.ttpamobile.ParametreSeance.setNomJoueur()`.

9.8.4.14 `int com.ttpa.iris.ttpamobile.ParametreSeance.puissanceBalles` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.ParametreSeance.estValide()`, `com.ttpa.iris.ttpamobile.ParametreSeance.getPuissanceBalles()`, `com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance()`, `com.ttpa.iris.ttpamobile.ParametreSeance.setPuissanceBalles()`, et `com.ttpa.iris.ttpamobile.ParametreSeance.toString()`.

9.8.4.15 int com.ttpa.iris.ttpamobile.ParametreSeance.rotation [private]

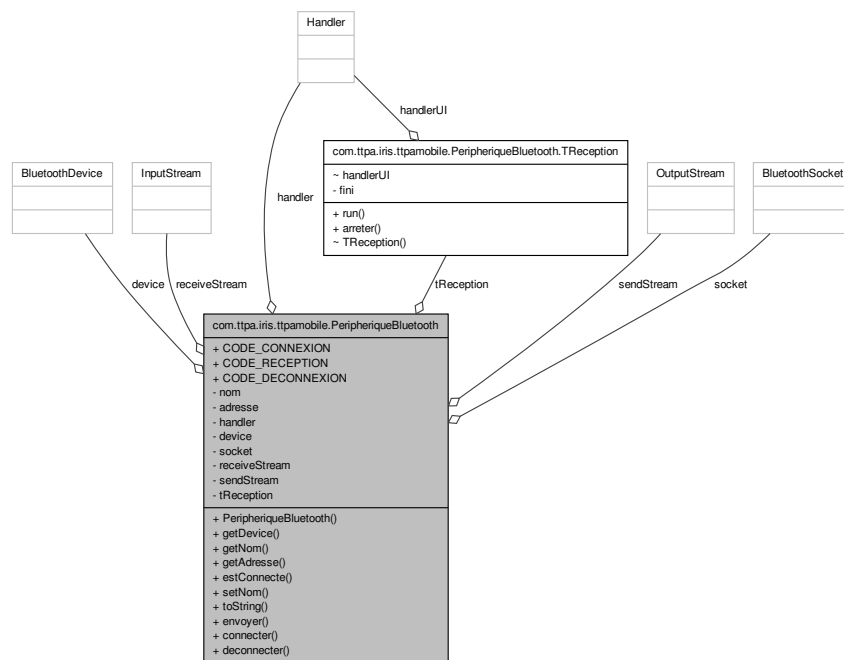
Référencé par [com.ttpa.iris.ttpamobile.ParametreSeance.estValide\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.getRotation\(\)](#), [com.ttpa.iris.ttpamobile.ParametreSeance.ParametreSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ParametreSeance.setRotation\(\)](#).

La documentation de cette classe a été générée à partir du fichier suivant :

– [ParametreSeance.java](#)

## 9.9 Référence de la classe com.ttpa.iris.ttpamobile.PeripheriqueBluetooth

Graphe de collaboration de com.ttpa.iris.ttpamobile.PeripheriqueBluetooth :



### Classes

– class [TReception](#)

### Fonctions membres publiques

- [PeripheriqueBluetooth](#) (BluetoothDevice [device](#), Handler [handler](#))
- BluetoothDevice [getDevice](#) ()
- String [getNom](#) ()
- String [getAdresse](#) ()
- boolean [estConnecte](#) ()
- void [setNom](#) (String [nom](#))
- String [toString](#) ()
- void [envoyer](#) (String data)
- void [connecter](#) ()
- boolean [deconnecter](#) (boolean fermeture)

## Attributs publics statiques

- static final int `CODE_CONNEXION` = 0
- static final int `CODE_RECEPTION` = 1
- static final int `CODE_DECONNEXION` = 2

## Attributs privés

- String `nom`
- String `adresse`
- Handler `handler` = null
- BluetoothDevice `device` = null
- BluetoothSocket `socket` = null
- InputStream `receiveStream` = null
- OutputStream `sendStream` = null
- `TReception tReception`

## 9.9.1 Description détaillée

Created by smaniotto on 04/04/18.

## 9.9.2 Documentation des constructeurs et destructeur

9.9.2.1 `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth ( BluetoothDevice device, Handler handler )`

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.adresse`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.device`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.handler`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.nom`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.receiveStream`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.sendStream`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.socket`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.tReception`.

```
{
 if(device != null)
 {
 this.device = device;
 this.nom = device.getName();
 this.adresse = device.getAddress();
 this.handler = handler;
 }
 else
 {
 this.device = device;
 this.nom = "Aucun";
 this.adresse = "";
 this.handler = handler;
 }

 try
 {
 System.out.println("<Bluetooth> nom " + device.getName());
 ParcelUuid[] uuids = device.getUuids();
 if(uuids != null)
 for(int i = 0; i < uuids.length; i++)
 System.out.println("<Bluetooth> uuid " + uuids[i].getUuid()
.toString());
 else
 System.out.println("<Bluetooth> uuid null !");

 socket = device.createRfcommSocketToServiceRecord(UUID.fromString("
00001101-0000-1000-8000-00805F9B34FB"));
 //socket =
device.createRfcommSocketToServiceRecord(uuids[0].getUuid());
 System.out.println("<Bluetooth> new socket");

 // API 23
 //System.out.println("<Bluetooth> max receive packet size : " +
socket.getMaxReceivePacketSize());
 //System.out.println("<Bluetooth> max transmit packet size : " +
```

```

socket.getMaxTransmitPacketSize());
//System.out.println("<Bluetooth> connection type : " +
socket.getConnectionType());

receiveStream = socket.getInputStream();
sendStream = socket.getOutputStream();
}
catch (IOException e)
{
 e.printStackTrace();
 System.out.println("<Bluetooth> Erreur socket !");
 socket = null;
}

if(socket != null)
 tReception = new TReception(handler);
}

```

### 9.9.3 Documentation des fonctions membres

#### 9.9.3.1 void com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter ( )

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.CODE\\_CONNEXION](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getAdresse\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getNom\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.handler](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.socket](#), et [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.tReception](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothLanceur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothTable\(\)](#).

```

{
 new Thread()
 {
 @Override public void run()
 {
 try
 {
 System.out.println("<Bluetooth> socket connect ...");
 socket.connect();
 System.out.println("<Bluetooth> socket connect ok");
 Message msg = Message.obtain();
 //msg.arg1 = CODE_CONNEXION;
 Bundle b = new Bundle();
 b.putString("nom", getNom());
 b.putString("adresse", getAdresse());
 b.putInt("etat", CODE_CONNEXION);
 b.putString("donnees", "");
 msg.setData(b);
 handler.sendMessage(msg);

 tReception.start();
 }
 catch (IOException e)
 {
 System.out.println("<Bluetooth> Erreur connect !");
 e.printStackTrace();
 }
 }
 }.start();
}

```

#### 9.9.3.2 boolean com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.deconnecter ( boolean fermeture )

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.arreter\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.socket](#), et [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.tReception](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.deconnexionPeripheriquesBluetooth\(\)](#).

```

{
 try
 {

```

```

 //if(estConnecte())
 {
 tReception.arreter();

 if(fermeture)
 {
 socket.close();
 System.out.println("<Bluetooth> socket close");
 }
 return true;
 }
 }
 catch (IOException e)
 {
 System.out.println("<Bluetooth> Erreur close !");
 e.printStackTrace();
 return false;
 }
}

```

### 9.9.3.3 void com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer ( String data )

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.sendStream](#), et [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.socket](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothTable\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothLanceur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripheriqueBluetoothLanceur\(\)](#).

```

{
 if(socket == null)
 return;

 try
 {
 //if(estConnecte())
 if(socket.isConnected())
 {
 System.out.println("<Bluetooth> Envoyer " + data);
 sendStream.write(data.getBytes());
 sendStream.flush();
 }
 else
 {
 System.out.println("<Bluetooth> Envoyer (non connecté) " + data
);
 }
 }
 catch (IOException e)
 {
 System.out.println("<Bluetooth> Erreur socket write !");
 e.printStackTrace();
 }
}

```

### 9.9.3.4 boolean com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte ( )

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.socket](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionPeripheriqueBluetoothTable\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameArretPeripheriqueBluetoothTable\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameDebutSeancePeripheriqueBluetoothLanceur\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothEcran\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTramePeripheriqueBluetoothLanceur\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.envoyerTrameRepriseSeancePeripherique-](#)

`BluetoothLanceur()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.verifierConnexionAppareilsBluetooth-Requis()`.

```
{
 if (socket != null)
 return socket.isConnected();
 return false;
}
```

#### 9.9.3.5 String `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getAdresse ( )`

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.adresse`.

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`.

```
{
 return adresse;
}
```

#### 9.9.3.6 BluetoothDevice `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getDevice ( )`

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.device`.

```
{
 return device;
}
```

#### 9.9.3.7 String `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getNom ( )`

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.nom`.

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`.

```
{
 return nom;
}
```

#### 9.9.3.8 void `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.setNom ( String nom )`

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.nom`.

```
{
 this.nom = nom;
}
```

#### 9.9.3.9 String `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.toString ( )`

Références `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.adresse`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.nom`.

```
{
 return "\nNom : " + nom + "\nAdresse : " + adresse;
}
```

### 9.9.4 Documentation des données membres

#### 9.9.4.1 String `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.adresse` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getAdresse()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.toString()`.

9.9.4.2 `final int com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.CODE_CONNEXION = 0` [static]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter()`.

9.9.4.3 `final int com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.CODE_DECONNEXION = 2`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`.

9.9.4.4 `final int com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.CODE_RECEPTION = 1` [static]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`.

9.9.4.5 `BluetoothDevice com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.device = null` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getDevice()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`.

9.9.4.6 `Handler com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.handler = null` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`.

9.9.4.7 `String com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.nom` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getNom()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.setNom()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.toString()`.

9.9.4.8 `InputStream com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.receiveStream = null` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`.

9.9.4.9 `OutputStream com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.sendStream = null` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`.

9.9.4.10 `BluetoothSocket com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.socket = null` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.deconnecter()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.envoyer()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.estConnecte()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`.

9.9.4.11 `TReception com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.tReception` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.connecter()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.deconnecter()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.PeripheriqueBluetooth()`.

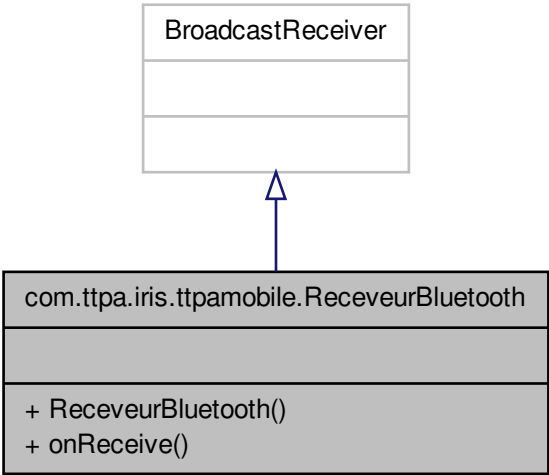
La documentation de cette classe a été générée à partir du fichier suivant :

– `PeripheriqueBluetooth.java`

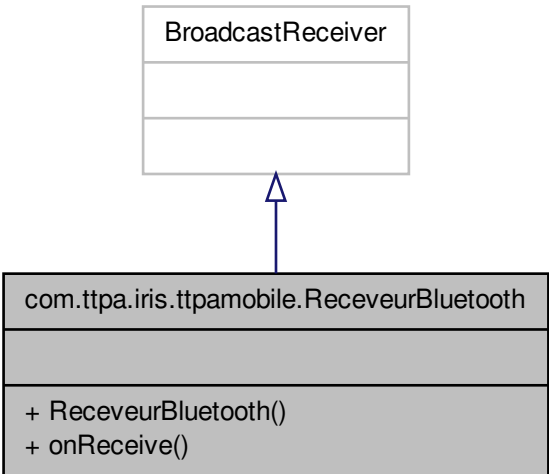


9.10 Référence de la classe com.ttpa.iris.ttpamobile.ReceveurBluetooth

Graphe d'héritage de com.ttpa.iris.ttpamobile.ReceveurBluetooth :



Graphe de collaboration de com.ttpa.iris.ttpamobile.ReceveurBluetooth :



## Fonctions membres publiques

- [ReceveurBluetooth](#) ()
- void [onReceive](#) (Context context, Intent intent)

## 9.10.1 Documentation des constructeurs et destructeur

## 9.10.1.1 com.ttpa.iris.ttpamobile.ReceveurBluetooth.ReceveurBluetooth ( )

```
{
}
```

## 9.10.2 Documentation des fonctions membres

## 9.10.2.1 void com.ttpa.iris.ttpamobile.ReceveurBluetooth.onReceive ( Context context, Intent intent )

```
{
 String action = intent.getAction();
 if (BluetoothDevice.ACTION_FOUND.equals(action))
 {
 BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.
 EXTRA_DEVICE);
 Toast.makeText(context, "Nouveau périphérique : " + device.getName(
), Toast.LENGTH_SHORT).show();
 }
}
```

La documentation de cette classe a été générée à partir du fichier suivant :

- [ReceveurBluetooth.java](#)

## 9.11 Référence de la classe com.ttpa.iris.ttpamobile.Seance

## Fonctions membres publiques

- [Seance](#) ()
- [Seance](#) (int [frequence](#), int [nombreBalles](#), String [effet](#), int [intensiteEffet](#), int [puissance](#), int [rotation](#))
- String [horodaterBD](#) ()
- int [getId](#) ()
- void [setId](#) (int [id](#))
- int [getFrequence](#) ()
- void [setFrequence](#) (int [frequence](#))
- int [getNombreBalles](#) ()
- void [setNombreBalles](#) (int [nombreBalles](#))
- void [setEffet](#) (String [effet](#))
- String [getEffet](#) ()
- void [setIntensiteEffet](#) (int [intensiteEffet](#))
- float [getTauxReussite](#) ()
- void [setTauxReussite](#) (float [tauxReussite](#))
- int [getPuissance](#) ()
- void [setPuissance](#) (int [puissance](#))
- int [getRotation](#) ()
- void [setRotation](#) (int [rotation](#))
- int [getZoneObjectif](#) ()
- void [setZoneObjectif](#) (int [zoneObjectif](#))
- int [getZoneRobot](#) ()
- void [setZoneRobot](#) (int [zoneRobot](#))
- String [getDateDebut](#) ()
- void [setDateDebut](#) (String [dateDebut](#))
- String [getDateFin](#) ()
- void [setDateFin](#) (String [dateFin](#))
- int [getIdJoueur](#) ()
- void [setIdJoueur](#) (int [idJoueur](#))
- String [toString](#) ()

## Fonctions de paquetage

- final int [getIntensiteEffet](#) ()

## Attributs privés

- int [id](#)
- int [frequence](#)
- int [nombreBalles](#)
- String [effet](#)
- int [intensiteEffet](#)
- int [puissance](#)
- int [rotation](#)
- int [zoneObjectif](#)
- int [zoneRobot](#)
- float [tauxReussite](#)
- String [dateDebut](#)
- String [dateFin](#)
- int [idJoueur](#)

## 9.11.1 Description détaillée

Created by smaniotto on 16/03/18. Classe [Seance](#) définissant les caractéristiques et le comportement d'une séance.

## 9.11.2 Documentation des constructeurs et destructeur

## 9.11.2.1 com.ttpa.iris.ttpamobile.Seance.Seance ( )

Méthode [Seance](#) constructeur par défaut de la classe [Seance](#).

```
{ }
```

9.11.2.2 com.ttpa.iris.ttpamobile.Seance.Seance ( int *frequence*, int *nombreBalles*, String *effet*, int *intensiteEffet*, int *puissance*, int *rotation* )

Méthode [Seance](#) constructeur de la classe [Seance](#).

## Paramètres

|                     |                                                                        |
|---------------------|------------------------------------------------------------------------|
| <i>frequence</i>    | étant la fréquence d'envoi des balles (en balles/minute) de la séance. |
| <i>nombreBalles</i> | étant le nombre de balles à envoyer.                                   |
| <i>effet</i>        | étant l'effet appliqué aux balles tout au long de la séance.           |
| <i>puissance</i>    | étant la puissance de la balle envoyée.                                |
| <i>rotation</i>     | étant la rotation du lanceur.                                          |

Références [com.ttpa.iris.ttpamobile.Seance.dateDebut](#), [com.ttpa.iris.ttpamobile.Seance.dateFin](#), [com.ttpa.iris.ttpamobile.Seance.effet](#), [com.ttpa.iris.ttpamobile.Seance.frequence](#), [com.ttpa.iris.ttpamobile.Seance.horodaterBD\(\)](#), [com.ttpa.iris.ttpamobile.Seance.idJoueur](#), [com.ttpa.iris.ttpamobile.Seance.intensiteEffet](#), [com.ttpa.iris.ttpamobile.Seance.nombreBalles](#), [com.ttpa.iris.ttpamobile.Seance.puissance](#), [com.ttpa.iris.ttpamobile.Seance.rotation](#), [com.ttpa.iris.ttpamobile.Seance.tauxReussite](#), [com.ttpa.iris.ttpamobile.Seance.zoneObjectif](#), et [com.ttpa.iris.ttpamobile.Seance.zoneRobot](#).

```
{
 this.frequence = frequence;
 this.nombreBalles = nombreBalles;
 this.effet = effet;
 this.intensiteEffet = intensiteEffet;
 this.puissance = puissance;
 this.rotation = rotation;
}
```

```
this.zoneObjectif = -1;
this.zoneRobot = -1;
this.tauxReussite = 0;
this.dateDebut = horodaterBD();
this.dateFin = horodaterBD();
this.idJoueur = -1;
}
```

### 9.11.3 Documentation des fonctions membres

#### 9.11.3.1 String `com.ttpa.iris.ttpamobile.Seance.getDateDebut ( )`

Références [com.ttpa.iris.ttpamobile.Seance.dateDebut](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return dateDebut;
}
```

#### 9.11.3.2 String `com.ttpa.iris.ttpamobile.Seance.getDateFin ( )`

Références [com.ttpa.iris.ttpamobile.Seance.dateFin](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return dateFin;
}
```

#### 9.11.3.3 String `com.ttpa.iris.ttpamobile.Seance.getEffet ( )`

Méthode `getEffet` accesseur de l'attribut effet.

Renvoie

effet

Références [com.ttpa.iris.ttpamobile.Seance.effet](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{ return effet; }
```

#### 9.11.3.4 int `com.ttpa.iris.ttpamobile.Seance.getFrequence ( )`

Méthode `getFrequence` accesseur de l'attribut frequence.

Renvoie

frequence

Références [com.ttpa.iris.ttpamobile.Seance.frequence](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return frequence;
}
```

**9.11.3.5** `int com.ttpa.iris.ttpamobile.Seance.getId ( )`

Méthode `getId` accesseur de l'attribut `id`.

Renvoie

`id`

Références [com.ttpa.iris.ttpamobile.Seance.id](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance\(\)](#).

```
{ return id; }
```

**9.11.3.6** `int com.ttpa.iris.ttpamobile.Seance.getIdJoueur ( )`

Références [com.ttpa.iris.ttpamobile.Seance.idJoueur](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return idJoueur;
}
```

**9.11.3.7** `final int com.ttpa.iris.ttpamobile.Seance.getIntensiteEffet ( )` [package]

Méthode `getIntensiteEffet` accesseur de l'attribut `intensiteEffet`.

Renvoie

`intensiteEffet`

Références [com.ttpa.iris.ttpamobile.Seance.intensiteEffet](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{ return intensiteEffet; }
```

**9.11.3.8** `int com.ttpa.iris.ttpamobile.Seance.getNombreBalles ( )`

Méthode `getNombreBalles` accesseur de l'attribut `nombreBalles`.

Renvoie

`nombreBalles`

Références [com.ttpa.iris.ttpamobile.Seance.nombreBalles](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.incrementerBallesJouees\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return nombreBalles;
}
```

**9.11.3.9** `int com.ttpa.iris.ttpamobile.Seance.getPuissance ( )`

Références [com.ttpa.iris.ttpamobile.Seance.puissance](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return puissance;
}
```

**9.11.3.10** `int com.ttpa.iris.ttpamobile.Seance.getRotation ( )`

Références [com.ttpa.iris.ttpamobile.Seance.rotation](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return rotation;
}
```

**9.11.3.11** `float com.ttpa.iris.ttpamobile.Seance.getTauxReussite ( )`

Méthode `getTauxReussite` accesseur de l'attribut `tauxReussite`.

Renvoie

Références [com.ttpa.iris.ttpamobile.Seance.tauxReussite](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return tauxReussite;
}
```

**9.11.3.12** `int com.ttpa.iris.ttpamobile.Seance.getZoneObjectif ( )`

Références [com.ttpa.iris.ttpamobile.Seance.zoneObjectif](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherBoiteDialogueDetailsSeance\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return zoneObjectif;
}
```

**9.11.3.13** `int com.ttpa.iris.ttpamobile.Seance.getZoneRobot ( )`

Références [com.ttpa.iris.ttpamobile.Seance.zoneRobot](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherBoiteDialogueDetailsSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

```
{
 return zoneRobot;
}
```

**9.11.3.14** `String com.ttpa.iris.ttpamobile.Seance.horodaterBD ( )`

Référencé par [com.ttpa.iris.ttpamobile.Seance.Seance\(\)](#).

```
{
 Calendar calendar = Calendar.getInstance();
 // Format SQLite : "2018-04-28 14:11:52"
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
 final String strDate = simpleDateFormat.format(calendar.getTime());
 return strDate;
}
```

**9.11.3.15** `void com.ttpa.iris.ttpamobile.Seance.setDateDebut ( String dateDebut )`

Références [com.ttpa.iris.ttpamobile.Seance.dateDebut](#).

Référéncé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

```
{
 this.dateDebut = dateDebut;
}
```

**9.11.3.16** `void com.ttpa.iris.ttpamobile.Seance.setDateFin ( String dateFin )`

Références [com.ttpa.iris.ttpamobile.Seance.dateFin](#).

Référéncé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

```
{
 this.dateFin = dateFin;
}
```

**9.11.3.17** `void com.ttpa.iris.ttpamobile.Seance.setEffet ( String effet )`

Méthode `setEffet` mutateur de l'attribut `effet`.

Paramètres

|              |                           |
|--------------|---------------------------|
| <i>effet</i> | étant l'effet à affecter. |
|--------------|---------------------------|

Références [com.ttpa.iris.ttpamobile.Seance.effet](#).

Référéncé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

```
{
 this.effet = effet;
}
```

**9.11.3.18** `void com.ttpa.iris.ttpamobile.Seance.setFrequence ( int frequence )`

Méthode `setFrequence` mutateur de l'attribut `frequence`.

Paramètres

|                  |                                |
|------------------|--------------------------------|
| <i>frequence</i> | étant la fréquence à affecter. |
|------------------|--------------------------------|

Références [com.ttpa.iris.ttpamobile.Seance.frequence](#).

Référéncé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

```
{
 this.frequence = frequence;
}
```

**9.11.3.19** `void com.ttpa.iris.ttpamobile.Seance.setId ( int id )`

Méthode `setId` mutateur de l'attribut `id`.

Paramètres

|           |                        |
|-----------|------------------------|
| <i>id</i> | étant l'id à affecter. |
|-----------|------------------------|

Références [com.ttpa.iris.ttpamobile.Seance.id](#).

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

```
{ this.id = id; }
```

#### 9.11.3.20 `void com.ttpa.iris.ttpamobile.Seance.setIdJoueur ( int idJoueur )`

Références `com.ttpa.iris.ttpamobile.Seance.idJoueur`.

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`, et `com.ttpa.iris.ttpamobile.IHM-EcranPrincipal.demarrerSeance()`.

```
{
 this.idJoueur = idJoueur;
}
```

#### 9.11.3.21 `void com.ttpa.iris.ttpamobile.Seance.setIntensiteEffet ( int intensiteEffet )`

Méthode `setIntensiteEffet` mutateur de l'attribut `intensiteEffet`.

##### Paramètres

|                       |                                          |
|-----------------------|------------------------------------------|
| <i>intensiteEffet</i> | étant l'intensité de l'effet à affecter. |
|-----------------------|------------------------------------------|

Références `com.ttpa.iris.ttpamobile.Seance.intensiteEffet`.

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

```
{
 this.intensiteEffet = intensiteEffet;
}
```

#### 9.11.3.22 `void com.ttpa.iris.ttpamobile.Seance.setNombreBalles ( int nombreBalles )`

Méthode `setNombreBalles` mutateur de l'attribut `nombreBalles`.

##### Paramètres

|                     |                                       |
|---------------------|---------------------------------------|
| <i>nombreBalles</i> | étant le nombre de balles à affecter. |
|---------------------|---------------------------------------|

Références `com.ttpa.iris.ttpamobile.Seance.nombreBalles`.

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

```
{
 this.nombreBalles = nombreBalles;
}
```

#### 9.11.3.23 `void com.ttpa.iris.ttpamobile.Seance.setPuissance ( int puissance )`

Références `com.ttpa.iris.ttpamobile.Seance.puissance`.

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

```
{
 this.puissance = puissance;
}
```

#### 9.11.3.24 `void com.ttpa.iris.ttpamobile.Seance.setRotation ( int rotation )`

Références `com.ttpa.iris.ttpamobile.Seance.rotation`.

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.



```
{
 this.rotation = rotation;
}
```

#### 9.11.3.25 void com.ttpa.iris.ttpamobile.Seance.setTauxReussite ( float *tauxReussite* )

Méthode setTauxReussite mutateur de l'attribut tauxReussite.

##### Paramètres

|                     |                                       |
|---------------------|---------------------------------------|
| <i>tauxReussite</i> | étant le taux de réussite à affecter. |
|---------------------|---------------------------------------|

Références [com.ttpa.iris.ttpamobile.Seance.tauxReussite](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.calculerReussiteSeance\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

```
{ this.tauxReussite = tauxReussite; }
```

#### 9.11.3.26 void com.ttpa.iris.ttpamobile.Seance.setZoneObjectif ( int *zoneObjectif* )

Références [com.ttpa.iris.ttpamobile.Seance.zoneObjectif](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#).

```
{
 this.zoneObjectif = zoneObjectif;
}
```

#### 9.11.3.27 void com.ttpa.iris.ttpamobile.Seance.setZoneRobot ( int *zoneRobot* )

Références [com.ttpa.iris.ttpamobile.Seance.zoneRobot](#).

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#), et [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#).

```
{
 this.zoneRobot = zoneRobot;
}
```

#### 9.11.3.28 String com.ttpa.iris.ttpamobile.Seance.toString ( )

Méthode toString permettant la visualisation des caractéristiques de la séance.

##### Renvoie

les caractéristiques de la séance.

Références [com.ttpa.iris.ttpamobile.Seance.dateDebut](#), [com.ttpa.iris.ttpamobile.Seance.dateFin](#), [com.ttpa.iris.ttpamobile.Seance.effet](#), [com.ttpa.iris.ttpamobile.Seance.frequence](#), [com.ttpa.iris.ttpamobile.Seance.idJoueur](#), [com.ttpa.iris.ttpamobile.Seance.intensiteEffet](#), [com.ttpa.iris.ttpamobile.Seance.nombreBalles](#), [com.ttpa.iris.ttpamobile.Seance.puissance](#), [com.ttpa.iris.ttpamobile.Seance.rotation](#), [com.ttpa.iris.ttpamobile.Seance.tauxReussite](#), [com.ttpa.iris.ttpamobile.Seance.zoneObjectif](#), et [com.ttpa.iris.ttpamobile.Seance.zoneRobot](#).

```
{
 return "Frequence : " + frequence + "\nNombre balles : " + nombreBalles
 + "\nEffet : " + effet + "\nIntensité effet : " + intensiteEffet + "\nTaux
 réussite : " + tauxReussite + "\nPuissance : " + puissance + "\nRotation robot : " +
 rotation + "\nZone objectif : " + zoneObjectif + "\nZone robot : " + zoneRobot
 + "\nDate début : " + dateDebut + "\nDate fin : " + dateFin + "\nID joueur : " +
 idJoueur;
}
```

## 9.11.4 Documentation des données membres

9.11.4.1 `String com.ttpa.iris.ttpamobile.Seance.dateDebut` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getDateDebut()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setDateDebut()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.2 `String com.ttpa.iris.ttpamobile.Seance.dateFin` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getDateFin()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setDateFin()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.3 `String com.ttpa.iris.ttpamobile.Seance.effet` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getEffet()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setEffet()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.4 `int com.ttpa.iris.ttpamobile.Seance.frequence` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getFrequence()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setFrequence()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.5 `int com.ttpa.iris.ttpamobile.Seance.id` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getId()`, et `com.ttpa.iris.ttpamobile.Seance.setId()`.

9.11.4.6 `int com.ttpa.iris.ttpamobile.Seance.idJoueur` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getIdJoueur()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setIdJoueur()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.7 `int com.ttpa.iris.ttpamobile.Seance.intensiteEffet` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getIntensiteEffet()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setIntensiteEffet()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.8 `int com.ttpa.iris.ttpamobile.Seance.nombreBalles` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setNombreBalles()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.9 `int com.ttpa.iris.ttpamobile.Seance.puissance` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getPuissance()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setPuissance()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.10 `int com.ttpa.iris.ttpamobile.Seance.rotation` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getRotation()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setRotation()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

9.11.4.11 `float com.ttpa.iris.ttpamobile.Seance.tauxReussite` [private]

Référencé par `com.ttpa.iris.ttpamobile.Seance.getTauxReussite()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setTauxReussite()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

**9.11.4.12** `int com.ttpa.iris.ttpamobile.Seance.zoneObjectif` `[private]`

Référencé par `com.ttpa.iris.ttpamobile.Seance.getZoneObjectif()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setZoneObjectif()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

**9.11.4.13** `int com.ttpa.iris.ttpamobile.Seance.zoneRobot` `[private]`

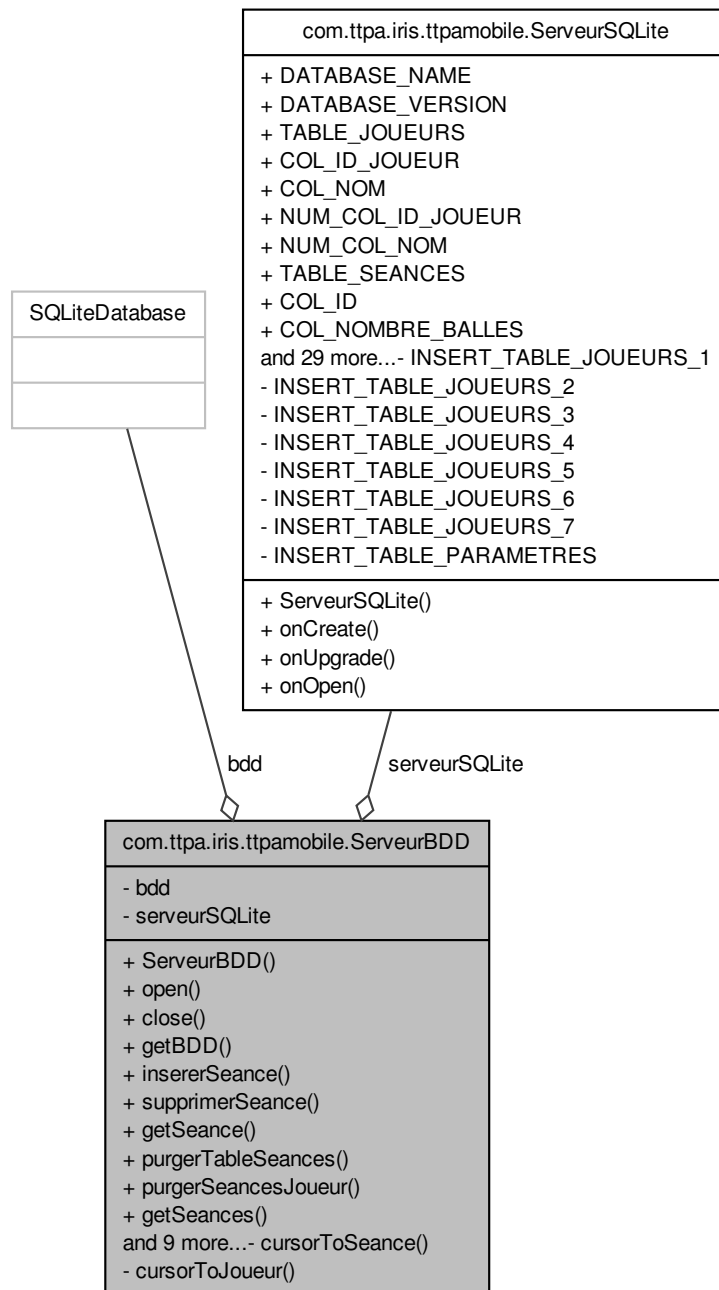
Référencé par `com.ttpa.iris.ttpamobile.Seance.getZoneRobot()`, `com.ttpa.iris.ttpamobile.Seance.Seance()`, `com.ttpa.iris.ttpamobile.Seance.setZoneRobot()`, et `com.ttpa.iris.ttpamobile.Seance.toString()`.

La documentation de cette classe a été générée à partir du fichier suivant :

- [Seance.java](#)

## 9.12 Référence de la classe com.ttpa.iris.ttpamobile.ServeurBDD

Graphes de collaboration de com.ttpa.iris.ttpamobile.ServeurBDD :



## Fonctions membres publiques

- [ServeurBDD](#) (Context context)
- void [open](#) ()

- void `close` ()
  - SQLiteDatabase `getBDD` ()
  - long `insérerSeance` (`Seance` seance)
  - int `supprimerSeance` (int id)
  - `Seance` `getSeance` (int id)
  - void `purgerTableSeances` ()
  - int `purgerSeancesJoueur` (int idJoueur)
  - List< `Seance` > `getSeances` ()
  - List< `Seance` > `getSeances` (int idJoueur)
  - long `insérerJoueur` (`Joueur` joueur)
  - int `supprimerJoueur` (int id)
  - int `supprimerJoueur` (String nom)
  - `Joueur` `getJoueur` (int id)
  - `Joueur` `getJoueur` (String nom)
  - void `purgerTableJoueurs` ()
  - List< `Joueur` > `getJoueurs` ()
  - int `getIdJoueurParametres` ()
  - int `setIdJoueurParametres` (int idJoueur)
- Permet de mettre à jour un enregistrement de la table.*

#### Fonctions membres privées

- `Seance` `cursorToSeance` (Cursor c, boolean one)
- `Joueur` `cursorToJoueur` (Cursor c, boolean one)

#### Attributs privés

- SQLiteDatabase `bdd` = null
- `ServeurSQLite` `serveurSQLite` = null

#### 9.12.1 Description détaillée

Classe `ServeurBDD` définissant les caractéristiques et le comportement d'un serveur de base de données.

#### 9.12.2 Documentation des constructeurs et destructeur

##### 9.12.2.1 com.ttpa.iris.ttpamobile.ServeurBDD.ServeurBDD ( Context context )

Méthode `ServeurBDD` constructeur de la classe `ServeurBDD`.

#### Paramètres

|                |  |
|----------------|--|
| <i>context</i> |  |
|----------------|--|

Références `com.ttpa.iris.ttpamobile.ServeurBDD.serveurSQLite`.

```
{
 // cn crée la BDD et ses tables
 serveurSQLite = new ServeurSQLite(context);
}
```

#### 9.12.3 Documentation des fonctions membres

##### 9.12.3.1 void com.ttpa.iris.ttpamobile.ServeurBDD.close ( )

Méthode `close` fermant la base de données.

Références `com.ttpa.iris.ttpamobile.ServeurBDD.bdd`.

```
{
 if (bdd != null)
```

```

 if (bdd.isOpen())
 bdd.close();
 }

```

### 9.12.3.2 Joueur com.ttpa.iris.ttpamobile.ServeurBDD.cursorToJoueur ( Cursor *c*, boolean *one* ) [private]

Méthode cursorToJoueur permettant de convertir un curseur en un objet de type [Joueur](#).

#### Paramètres

|            |  |
|------------|--|
| <i>c</i>   |  |
| <i>one</i> |  |

#### Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_ID\\_JOUEUR](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_NOM](#), [com.ttpa.iris.ttpamobile.Joueur.setId\(\)](#), et [com.ttpa.iris.ttpamobile.Joueur.setNom\(\)](#).

```

{
 if (c.getCount() == 0)
 return null;

 if (one == true)
 c.moveToFirst();

 Joueur joueur = new Joueur();

 joueur.setId(c.getInt(ServeurSQLite.NUM_COL_ID_JOUEUR));
 joueur.setNom(c.getString(ServeurSQLite.NUM_COL_NOM));

 if (one == true)
 c.close();

 return joueur;
}

```

### 9.12.3.3 Seance com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance ( Cursor *c*, boolean *one* ) [private]

Méthode cursorToSeance permettant de convertir un curseur en un objet de type [Seance](#).

#### Paramètres

|            |  |
|------------|--|
| <i>c</i>   |  |
| <i>one</i> |  |

#### Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_DATE\\_DEBUT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_DATE\\_FIN](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_EFFECT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_FREQUENCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_ID](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_ID\\_JOUEUR\\_SEANCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_INTENSITE\\_EFFECT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_NOMBRE\\_BALLES](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_PUISSANCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_ROTATION](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_TAUX\\_REUSSITE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_ZONE\\_OBJECTIF](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.NUM\\_COL\\_ZONE\\_ROBOT](#), [com.ttpa.iris.ttpamobile.Seance.setDateDebut\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setDateFin\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setEffect\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setFrequence\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setIdJoueur\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setIntensiteEffect\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setNombreBalles\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setPuisance\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setRotation\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setTauxReussite\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setZoneObjectif\(\)](#), [com.ttpa.iris.ttpamobile.Seance.setZoneRobot\(\)](#).

`iris.ttpamobile.Seance.setFrequence()`, `com.ttpa.iris.ttpamobile.Seance.setId()`, `com.ttpa.iris.ttpamobile.Seance.setIdJoueur()`, `com.ttpa.iris.ttpamobile.Seance.setIntensiteEffet()`, `com.ttpa.iris.ttpamobile.Seance.setNombreBalles()`, `com.ttpa.iris.ttpamobile.Seance.setPuissance()`, `com.ttpa.iris.ttpamobile.Seance.setRotation()`, `com.ttpa.iris.ttpamobile.Seance.setTauxReussite()`, `com.ttpa.iris.ttpamobile.Seance.setZoneObjectif()`, et `com.ttpa.iris.ttpamobile.Seance.setZoneRobot()`.

```
{
 if (c.getCount() == 0)
 return null;

 if (one == true)
 c.moveToFirst();

 Seance seance = new Seance();

 seance.setId(c.getInt(ServeurSQLite.NUM_COL_ID));
 seance.setNombreBalles(c.getInt(ServeurSQLite.NUM_COL_NOMBRE_BALLES));
 seance.setFrequence(c.getInt(ServeurSQLite.NUM_COL_FREQUENCE));
 seance.setEffet(c.getString(ServeurSQLite.NUM_COL_EFFET));
 seance.setIntensiteEffet(c.getInt(ServeurSQLite.NUM_COL_INTENSITE_EFFET));
 seance.setPuissance(c.getInt(ServeurSQLite.NUM_COL_PUISSANCE));
 seance.setRotation(c.getInt(ServeurSQLite.NUM_COL_ROTATION));
 seance.setZoneObjectif(c.getInt(ServeurSQLite.NUM_COL_ZONE_OBJECTIF));
 seance.setZoneRobot(c.getInt(ServeurSQLite.NUM_COL_ZONE_ROBOT));
 seance.setTauxReussite(c.getInt(ServeurSQLite.NUM_COL_TAUX_REUSSITE));
 seance.setDateDebut(c.getString(ServeurSQLite.NUM_COL_DATE_DEBUT));
 seance.setDateFin(c.getString(ServeurSQLite.NUM_COL_DATE_FIN));
 seance.setIdJoueur(c.getInt(ServeurSQLite.NUM_COL_ID_JOUEUR_SEANCE));

 if (one == true)
 c.close();

 return seance;
}
```

#### 9.12.3.4 SQLiteDatabase `com.ttpa.iris.ttpamobile.ServeurBDD.getBDD()`

Méthode `getBDD` accesseur de l'attribut `bdd`.

Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurBDD.bdd](#).

```
{
 return bdd;
}
```

#### 9.12.3.5 `int com.ttpa.iris.ttpamobile.ServeurBDD.getIdJoueurParametres()`

Renvoie

`int` l'id du joueur

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID\\_JOUEUR](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_PARAMETRES](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#), [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.demarrerSeance\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.recupererInformationsJoueur\(\)](#).

```
{
 Cursor c = bdd.query(ServeurSQLite.TABLE_PARAMETRES, new String[] {
 ServeurSQLite.COL_ID_JOUEUR, ServeurSQLite.COL_ID_PARAMETRE + " = '1'", null, null,
 null, null);

 if (c.getCount() == 0)
```

```

 return 0;

 c.moveToFirst();

 int idJoueur = c.getInt(0);

 c.close();

 return idJoueur;
 }

```

#### 9.12.3.6 Joueur `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueur ( int id )`

Méthode `getJoueur` permettant l'accès à un joueur grâce à son attribut `id`.

##### Paramètres

|           |                                   |
|-----------|-----------------------------------|
| <i>id</i> | étant l'id du joueur à retourner. |
|-----------|-----------------------------------|

##### Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID\\_JOUEUR](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_NOM](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_JOUEURS](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#), et [com.ttpa.iris.ttpamobile.IHMHistoireSeances.recupererInformationsJoueur\(\)](#).

```

{
 Cursor c = bdd.query(ServeurSQLite.TABLE_JOUEURS, new String[] {
 ServeurSQLite.COL_ID_JOUEUR, ServeurSQLite.COL_NOM}, ServeurSQLite.COL_ID_JOUEUR + " = "
 + id, null, null, null, null);

 return cursorToJoueur(c, true);
}

```

#### 9.12.3.7 Joueur `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueur ( String nom )`

Méthode `getJoueur` permettant l'accès à un joueur grâce à son nom.

##### Paramètres

|            |                                      |
|------------|--------------------------------------|
| <i>nom</i> | String le nom du joueur à retourner. |
|------------|--------------------------------------|

##### Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID\\_JOUEUR](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_NOM](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_JOUEURS](#).

```

{
 Cursor c = bdd.query(ServeurSQLite.TABLE_JOUEURS, new String[] {
 ServeurSQLite.COL_ID_JOUEUR, ServeurSQLite.COL_NOM}, ServeurSQLite.COL_NOM + " = '" +
 nom + "'", null, null, null, null);

 return cursorToJoueur(c, true);
}

```

#### 9.12.3.8 `List<Joueur> com.ttpa.iris.ttpamobile.ServeurBDD.getJoueurs ( )`

Méthode `getJoueurs` retournant tous les joueurs présents dans la table des joueurs.



**Renvoie**

liste d'objets `Joueur`.

Références `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ID_JOUEUR`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_NOM`, et `com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE_JOUEURS`.

Référencé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs()`.

```
{
 List<Joueur> joueurs = new ArrayList<Joueur>();

 Cursor cursor = bdd.query(ServeurSQLite.TABLE_JOUEURS, new String[] {
 ServeurSQLite.COL_ID_JOUEUR, ServeurSQLite.COL_NOM}, null, null, null, null, null)
 ;

 cursor.moveToFirst();
 while (!cursor.isAfterLast())
 {
 Joueur joueur = cursorToJoueur(cursor, false);
 joueurs.add(joueur);
 cursor.moveToNext();
 }

 cursor.close();

 return joueurs;
}
```

**9.12.3.9 Seance `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance ( int id )`**

Méthode `getSeance` permettant l'accès à une séance grâce à son attribut `id`.

**Paramètres**

|                 |                                      |
|-----------------|--------------------------------------|
| <code>id</code> | étant l'id de la séance à retourner. |
|-----------------|--------------------------------------|

**Renvoie**

Références `com.ttpa.iris.ttpamobile.ServeurBDD.bdd`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_DATE_DEBUT`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_DATE_FIN`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_EFFECT`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_FREQUENCE`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ID`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ID_JOUEUR`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_INTENSITE_EFFECT`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_NOMBRE_BALLES`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_PUISSANCE`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ROTATION`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_TAUX_REUSSITE`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ZONE_OBJECTIF`, `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ZONE_ROBOT`, et `com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE_SEANCES`.

Référencé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.afficherBoiteDialogueDetailsSeance()`.

```
{
 Cursor c = bdd.query(ServeurSQLite.TABLE_SEANCES, new String[] {
 ServeurSQLite.COL_ID, ServeurSQLite.COL_NOMBRE_BALLES, ServeurSQLite.COL_FREQUENCE,
 ServeurSQLite.COL_EFFECT, ServeurSQLite.COL_INTENSITE_EFFECT, ServeurSQLite.
 COL_PUISSANCE, ServeurSQLite.COL_ROTATION, ServeurSQLite.COL_ZONE_OBJECTIF, ServeurSQLite.
 COL_ZONE_ROBOT, ServeurSQLite.COL_TAUX_REUSSITE, ServeurSQLite.COL_DATE_DEBUT,
 ServeurSQLite.COL_DATE_FIN, ServeurSQLite.COL_ID_JOUEUR}, ServeurSQLite.COL_ID +
 " = " + id, null, null, null, null);

 return cursorToSeance(c, true);
}
```

**9.12.3.10 `List<Seance> com.ttpa.iris.ttpamobile.ServeurBDD.getSeances ( )`**

Méthode `getSeances` retournant toutes les séances présentes dans la table des séances.

**Renvoi**

liste d'objets [Seance](#).

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_DATE\\_DEBUT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_DATE\\_FIN](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_EFFECT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_FREQUENCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID\\_JOUEUR](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_INTENSITE\\_EFFECT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_NOMBRE\\_BALLES](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_PUISSANCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ROTATION](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_TAUX\\_REUSSITE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ZONE\\_OBJECTIF](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ZONE\\_ROBOT](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_SEANCES](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.modifierIHMAfficherSeancesJoueur\(\)](#).

```
{
 List<Seance> seances = new ArrayList<Seance>();

 Cursor cursor = bdd.query(ServeurSQLite.TABLE_SEANCES, new String[] {
 ServeurSQLite.COL_ID, ServeurSQLite.COL_FREQUENCE, ServeurSQLite.COL_NOMBRE_BALLES
 , ServeurSQLite.COL_EFFECT, ServeurSQLite.COL_INTENSITE_EFFECT, ServeurSQLite.
 COL_PUISSANCE, ServeurSQLite.COL_ROTATION, ServeurSQLite.COL_ZONE_OBJECTIF,
 ServeurSQLite.COL_ZONE_ROBOT, ServeurSQLite.COL_TAUX_REUSSITE, ServeurSQLite.
 COL_DATE_DEBUT, ServeurSQLite.COL_DATE_FIN, ServeurSQLite.COL_ID_JOUEUR}, null, null, null,
 null, null);

 cursor.moveToFirst();
 while (!cursor.isAfterLast())
 {
 Seance seance = cursorToSeance(cursor, false);
 seances.add(seance);
 cursor.moveToNext();
 }

 cursor.close();

 return seances;
}
```

**9.12.3.11 List<Seance> com.ttpa.iris.ttpamobile.ServeurBDD.getSeances ( int idJoueur )**

Méthode getSeances retournant toutes les séances présentes dans la table des séances ayant pour id idJoueur.

**Paramètres**

|                 |  |
|-----------------|--|
| <i>idJoueur</i> |  |
|-----------------|--|

**Renvoi**

liste d'objets [Seance](#).

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_DATE\\_DEBUT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_DATE\\_FIN](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_EFFECT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_FREQUENCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID\\_JOUEUR](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_INTENSITE\\_EFFECT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_NOMBRE\\_BALLES](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_PUISSANCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ROTATION](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_TAUX\\_REUSSITE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ZONE\\_OBJECTIF](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ZONE\\_ROBOT](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_SEANCES](#).

```
{
 List<Seance> seances = new ArrayList<Seance>();

 Cursor cursor = bdd.query(ServeurSQLite.TABLE_SEANCES, new String[] {
```

```

ServeurSQLite.COL_ID, ServeurSQLite.COL_FREQUENCE, ServeurSQLite.COL_NOMBRE_BALLES
, ServeurSQLite.COL_EFFET, ServeurSQLite.COL_INTENSITE_EFFET, ServeurSQLite.
COL_PUISSANCE, ServeurSQLite.COL_ROTATION, ServeurSQLite.COL_ZONE_OBJECTIF,
ServeurSQLite.COL_ZONE_ROBOT, ServeurSQLite.COL_TAUX_REUSSITE, ServeurSQLite.
COL_DATE_DEBUT, ServeurSQLite.COL_DATE_FIN, ServeurSQLite.COL_ID_JOUEUR}, ServeurSQLite.
COL_ID_JOUEUR + " = " + idJoueur, null, null, null, null);

 cursor.moveToFirst();
 while (!cursor.isAfterLast())
 {
 Seance seance = cursorToSeance(cursor, false);
 seances.add(seance);
 cursor.moveToNext();
 }

 cursor.close();

 return seances;
}

```

#### 9.12.3.12 long com.ttpa.iris.ttpamobile.ServeurBDD.insererJoueur ( Joueur *joueur* )

Méthode insererJoueur permettant l'insertion d'un joueur.

##### Paramètres

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>joueur</i> | étant le joueur à insérer dans la base de données. |
|---------------|----------------------------------------------------|

##### Renvoie

l'id dans la base de données.

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_NOM](#), [com.ttpa.iris.ttpamobile.Joueur.getNom\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_JOUEURS](#).

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.ajouterJoueur\(\)](#).

```

{
 ContentValues values = new ContentValues();

 //values.put(ServeurSQLite.COL_ID_JOUEUR, joueur.getId());
 values.put(ServeurSQLite.COL_NOM, joueur.getNom());

 return bdd.insert(ServeurSQLite.TABLE_JOUEURS, null, values);
}

```

#### 9.12.3.13 long com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance ( Seance *seance* )

Méthode insererSeance permettant l'insertion d'une séance dans la base de données.

##### Paramètres

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>seance</i> | étant la séance à insérer dans la base de données. |
|---------------|----------------------------------------------------|

##### Renvoie

l'id de la séance dans la base de données.

Références [com.ttpa.iris.ttpamobile.ServeurBDD.bdd](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_DATE\\_DEBUT](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_DATE\\_FIN](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_EFFET](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_FREQUENCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID\\_JOUEUR](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_INTENSITE\\_EFFET](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_NOMBRE\\_BALLES](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_PUISSANCE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ROTATION](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_TAUX\\_REUSSITE](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ZONE\\_OBJECTIF](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ZONE\\_ROBOT](#), [com.ttpa.iris.ttpamobile.Seance.getDateDebut\(\)](#),

`com.ttpa.iris.ttpamobile.Seance.getDateFin()`, `com.ttpa.iris.ttpamobile.Seance.getEffet()`, `com.ttpa.iris.ttpamobile.Seance.getFrequence()`, `com.ttpa.iris.ttpamobile.Seance.getIdJoueur()`, `com.ttpa.iris.ttpamobile.Seance.getIntensiteEffet()`, `com.ttpa.iris.ttpamobile.Seance.getNombreBalles()`, `com.ttpa.iris.ttpamobile.Seance.getPuissance()`, `com.ttpa.iris.ttpamobile.Seance.getRotation()`, `com.ttpa.iris.ttpamobile.Seance.getTauxReussite()`, `com.ttpa.iris.ttpamobile.Seance.getZoneObjectif()`, `com.ttpa.iris.ttpamobile.Seance.getZoneRobot()`, et `com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE_SEANCES`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.arreterSeance()`.

```
{
 ContentValues values = new ContentValues();

 values.put(ServeurSQLite.COL_FREQUENCE, seance.getFrequence());
 values.put(ServeurSQLite.COL_NOMBRE_BALLES, seance.getNombreBalles());
 values.put(ServeurSQLite.COL_EFFET, seance.getEffet());
 values.put(ServeurSQLite.COL_INTENSITE_EFFET, seance.getIntensiteEffet());
 values.put(ServeurSQLite.COL_PUISSANCE, seance.getPuissance());
 values.put(ServeurSQLite.COL_ROTATION, seance.getRotation());
 values.put(ServeurSQLite.COL_ZONE_OBJECTIF, seance.getZoneObjectif());
 values.put(ServeurSQLite.COL_ZONE_ROBOT, seance.getZoneRobot());
 values.put(ServeurSQLite.COL_TAUX_REUSSITE, seance.getTauxReussite());
 values.put(ServeurSQLite.COL_DATE_DEBUT, seance.getDateDebut());
 values.put(ServeurSQLite.COL_DATE_FIN, seance.getDateFin());
 values.put(ServeurSQLite.COL_ID_JOUEUR, seance.getIdJoueur());

 return bdd.insert(ServeurSQLite.TABLE_SEANCES, null, values);
}
```

#### 9.12.3.14 `void com.ttpa.iris.ttpamobile.ServeurBDD.open ( )`

Méthode `open` ouvrant la base de données en écriture.

Références `com.ttpa.iris.ttpamobile.ServeurBDD.bdd`, et `com.ttpa.iris.ttpamobile.ServeurBDD.serveurSQLite`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.connectionBaseDeDonnees()`, et `com.ttpa.iris.ttpamobile.IHMEcranPrincipal.connexionBaseDeDonnees()`.

```
{
 // on ouvre la BDD en écriture
 if (bdd == null)
 bdd = serveurSQLite.getWritableDatabase();
}
```

#### 9.12.3.15 `int com.ttpa.iris.ttpamobile.ServeurBDD.purgerSeancesJoueur ( int idJoueur )`

Méthode `purgerSeancesJoueur` permettant la suppression des séances d'un joueur.

Paramètres

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>idJoueur</i> | étant l'id du joueur auquel on doit purger les séances. |
|-----------------|---------------------------------------------------------|

Renvoie

Références `com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ID_JOUEUR`, et `com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE_SEANCES`.

Référéncé par `com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.purgerSeancesJoueur()`.

```
{
 return bdd.delete(ServeurSQLite.TABLE_SEANCES, ServeurSQLite.
 COL_ID_JOUEUR + " = " + idJoueur, null);
}
```

**9.12.3.16 void com.ttpa.iris.ttpamobile.ServeurBDD.purgerTableJoueurs ( )**

Méthode purgerTableJoueurs permettant la purge (suppression totale) de la table des joueurs.

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE\\_BDD\\_JOUEURS](#).

```
{
 bdd.execSQL("DROP TABLE IF EXISTS " + serveurSQLite.TABLE_JOUEURS); //
 Supprimer la table
 bdd.execSQL(ServeurSQLite.CREATE_BDD_JOUEURS); // Recréer la table
}
```

**9.12.3.17 void com.ttpa.iris.ttpamobile.ServeurBDD.purgerTableSeances ( )**

Méthode purgerTableSeances permettant la purge (suppression totale) de la table des séances.

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE\\_BDD\\_SEANCES](#).

```
{
 bdd.execSQL("DROP TABLE IF EXISTS " + serveurSQLite.TABLE_SEANCES); //
 Supprimer la table
 bdd.execSQL(ServeurSQLite.CREATE_BDD_SEANCES); // Recréer la table
}
```

**9.12.3.18 int com.ttpa.iris.ttpamobile.ServeurBDD.setIdJoueurParametres ( int idJoueur )**

Paramètres

|                 |                                        |
|-----------------|----------------------------------------|
| <i>idJoueur</i> | int représente l'identifiant du joueur |
|-----------------|----------------------------------------|

Renvoie

un int qui permet de savoir si la mise à jour de l'enregistrement a réussi

Référencé par [com.ttpa.iris.ttpamobile.IHMEcranPrincipal.creerListeJoueurs\(\)](#).

```
{
 ContentValues values = new ContentValues();
 values.put("ID_JOUEUR", idJoueur);
 return bdd.update("table_parametres", values, "ID_PARAMETRE = " + 1,
 null);
}
```

**9.12.3.19 int com.ttpa.iris.ttpamobile.ServeurBDD.supprimerJoueur ( int id )**

Méthode supprimerJoueur permettant la suppression d'un joueur.

Paramètres

|           |                                   |
|-----------|-----------------------------------|
| <i>id</i> | étant l'id du joueur à supprimer. |
|-----------|-----------------------------------|

Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID\\_JOUEUR](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_JOUEURS](#).

```
{
 return bdd.delete(ServeurSQLite.TABLE_JOUEURS, ServeurSQLite.
 COL_ID_JOUEUR + " = " + id, null);
}
```

#### 9.12.3.20 `int com.ttpa.iris.ttpamobile.ServeurBDD.supprimerJoueur ( String nom )`

Méthode `supprimerJoueur` permettant la suppression d'un joueur.

##### Paramètres

|            |                                      |
|------------|--------------------------------------|
| <i>nom</i> | String le nom du joueur à supprimer. |
|------------|--------------------------------------|

##### Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_NOM](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_JOUEURS](#).

```
{
 return bdd.delete(ServeurSQLite.TABLE_JOUEURS, ServeurSQLite.COL_NOM +
 " = '" + nom + "'", null);
}
```

#### 9.12.3.21 `int com.ttpa.iris.ttpamobile.ServeurBDD.supprimerSeance ( int id )`

Méthode `supprimerSeance` permettant la suppression d'une séance dans la base de données.

##### Paramètres

|           |                                      |
|-----------|--------------------------------------|
| <i>id</i> | étant l'id de la séance à supprimer. |
|-----------|--------------------------------------|

##### Renvoie

Références [com.ttpa.iris.ttpamobile.ServeurBDD.bdd](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.COL\\_ID](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_SEANCES](#).

Référéncé par [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances.supprimerSeanceSelectionnee\(\)](#).

```
{
 return bdd.delete(ServeurSQLite.TABLE_SEANCES, ServeurSQLite.COL_ID + "
 = " + id, null);
}
```

### 9.12.4 Documentation des données membres

#### 9.12.4.1 `SQLiteDatabase com.ttpa.iris.ttpamobile.ServeurBDD.bdd = null` [private]

Référéncé par [com.ttpa.iris.ttpamobile.ServeurBDD.close\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.getBDD\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.getSeance\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.open\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.supprimerSeance\(\)](#).

#### 9.12.4.2 `ServeurSQLite com.ttpa.iris.ttpamobile.ServeurBDD.serveurSQLite = null` [private]

Référéncé par [com.ttpa.iris.ttpamobile.ServeurBDD.open\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.ServeurBDD\(\)](#).

La documentation de cette classe a été générée à partir du fichier suivant :

– [ServeurBDD.java](#)

## 9.13 Référence de la classe `com.ttpa.iris.ttpamobile.ServeurSQLite`

## Fonctions membres publiques

- `ServeurSQLite` (Context context)
  - void `onCreate` (SQLiteDatabase db)
  - void `onUpgrade` (SQLiteDatabase db, int oldVersion, int newVersion)
  - void `onOpen` (SQLiteDatabase db)
- Ajoute les droits en lecture et en écriture à la base de données lors de son ouverture.*

## Attributs publics statiques

- static final String `DATABASE_NAME` = "ttpa\_mobile.db"
- static final int `DATABASE_VERSION` = 1
- static final String `TABLE_JOUEURS` = "table\_joueurs"
- static final String `COL_ID_JOUEUR` = "ID\_JOUEUR"
- static final String `COL_NOM` = "NOM"
- static final int `NUM_COL_ID_JOUEUR` = 0
- static final int `NUM_COL_NOM` = 1
- static final String `TABLE_SEANCES` = "table\_seances"
- static final String `COL_ID` = "ID\_SEANCE"
- static final String `COL_NOMBRE_BALLES` = "NOMBRE\_BALLES"
- static final String `COL_FREQUENCE` = "FREQUENCE"
- static final String `COL_EFFET` = "EFFET"
- static final String `COL_INTENSITE_EFFET` = "INTENSITE\_EFFET"
- static final String `COL_PUISSANCE` = "PUISSANCE"
- static final String `COL_ROTATION` = "ROTATION"
- static final String `COL_ZONE_OBJECTIF` = "ZONE\_OBJECTIF"
- static final String `COL_ZONE_ROBOT` = "ZONE\_ROBOT"
- static final String `COL_TAUX_REUSSITE` = "TAUX\_REUSSITE"
- static final String `COL_DATE_DEBUT` = "DATE\_DEBUT"
- static final String `COL_DATE_FIN` = "DATE\_FIN"
- static final int `NUM_COL_ID` = 0
- static final int `NUM_COL_FREQUENCE` = 1
- static final int `NUM_COL_NOMBRE_BALLES` = 2
- static final int `NUM_COL_EFFET` = 3
- static final int `NUM_COL_INTENSITE_EFFET` = 4
- static final int `NUM_COL_PUISSANCE` = 5
- static final int `NUM_COL_ROTATION` = 6
- static final int `NUM_COL_ZONE_OBJECTIF` = 7
- static final int `NUM_COL_ZONE_ROBOT` = 8
- static final int `NUM_COL_TAUX_REUSSITE` = 9
- static final int `NUM_COL_DATE_DEBUT` = 10
- static final int `NUM_COL_DATE_FIN` = 11
- static final int `NUM_COL_ID_JOUEUR_SEANCE` = 12
- static final String `TABLE_PARAMETRES` = "table\_parametres"
- static final String `COL_ID_PARAMETRE` = "ID\_PARAMETRE"
- static final int `NUM_COL_ID_PARAMETRE` = 0
- static final int `NUM_COL_ID_JOUEUR_PARAMETRE` = 1
- static final String `CREATE_BDD_JOUEURS` = " VARCHAR(255) NOT NULL);"
- static final String `CREATE_BDD_SEANCES` = "CONSTRAINT fk\_seances\_1 FOREIGN KEY (ID\_JOUEUR) REFERENCES table\_joueurs (ID\_JOUEUR) ON DELETE CASCADE);"
- static final String `CREATE_BDD_PARAMETRES` = "CONSTRAINT fk\_parametres\_1 FOREIGN KEY (ID\_JOUEUR) REFERENCES table\_joueurs (ID\_JOUEUR));"

## Attributs privés statiques

- static final String `INSERT_TABLE_JOUEURS_1` = "INSERT INTO table\_joueurs(NOM) VALUES('LEGO-UT Christophe');"
- static final String `INSERT_TABLE_JOUEURS_2` = "INSERT INTO table\_joueurs(NOM) VALUES('MARTINEZ Michel');"
- static final String `INSERT_TABLE_JOUEURS_3` = "INSERT INTO table\_joueurs(NOM) VALUES('LEBESSON Emmanuel');"
- static final String `INSERT_TABLE_JOUEURS_4` = "INSERT INTO table\_joueurs(NOM) VALUES('ELOI Damien');"
- static final String `INSERT_TABLE_JOUEURS_5` = "INSERT INTO table\_joueurs(NOM) VALUES('MATTENET Adrien');"
- static final String `INSERT_TABLE_JOUEURS_6` = "INSERT INTO table\_joueurs(NOM) VALUES('CHILA Patrick');"

- static final String `INSERT_TABLE_JOUEURS_7` = "INSERT INTO table\_joueurs(NOM) VALUES('BEAU-MONT Jérôme');"
- static final String `INSERT_TABLE_PARAMETRES` = "INSERT INTO table\_parametres(ID\_PARAMETRE, ID\_JOUEUR) VALUES(1, 1);"

#### 9.13.1 Description détaillée

Classe `ServeurSQLite` définissant les caractéristiques de la base de données.

#### 9.13.2 Documentation des constructeurs et destructeur

##### 9.13.2.1 `com.ttpa.iris.ttpamobile.ServeurSQLite.ServeurSQLite ( Context context )`

Méthode `ServeurSQLite` constructeur de la classe `ServeurSQLite`.

Paramètres

|                |  |
|----------------|--|
| <i>context</i> |  |
|----------------|--|

Références `com.ttpa.iris.ttpamobile.ServeurSQLite.DATABASE_NAME`, et `com.ttpa.iris.ttpamobile.ServeurSQLite.DATABASE_VERSION`.

```
{
 super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
```

#### 9.13.3 Documentation des fonctions membres

##### 9.13.3.1 `void com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate ( SQLiteDatabase db )`

Méthode `onCreate` appelée à la création de l'objet et permettant l'exécution des requêtes créant les tables de la base de données.

Paramètres

|           |  |
|-----------|--|
| <i>db</i> |  |
|-----------|--|

Références `com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE_BDD_JOUEURS`, `com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE_BDD_PARAMETRES`, `com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE_BDD_SEANCES`, `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_1`, `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_2`, `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_3`, `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_4`, `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_5`, `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_6`, `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_7`, et `com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_PARAMETRES`.

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onUpgrade()`.

```
{
 // On crée la table des séances
 db.execSQL("pragma foreign_keys = on;");
 db.execSQL(CREATE_BDD_JOUEURS);
 db.execSQL(CREATE_BDD_SEANCES);
 db.execSQL(CREATE_BDD_PARAMETRES);
 db.execSQL(INSERT_TABLE_JOUEURS_1);
 db.execSQL(INSERT_TABLE_JOUEURS_2);
 db.execSQL(INSERT_TABLE_JOUEURS_3);
 db.execSQL(INSERT_TABLE_JOUEURS_4);
 db.execSQL(INSERT_TABLE_JOUEURS_5);
 db.execSQL(INSERT_TABLE_JOUEURS_6);
 db.execSQL(INSERT_TABLE_JOUEURS_7);
 db.execSQL(INSERT_TABLE_PARAMETRES);
}
```



```

String path = db.getPath();
File f = new File(path);
boolean r = f.setReadable(true, false);
if(r)
{
 Log.d("TTPA", "onCreate : Ajout droit lecture " + path); // d =
debug
}
else
{
 Log.e("TTPA", "onCreate : Erreur ajout droit lecture " + path); //
e = erreur
}
r = f.setWritable(true, false);
if(r)
{
 Log.d("TTPA", "onCreate : Ajout droit écriture " + path); // d =
debug
}
else
{
 Log.e("TTPA", "onCreate : Erreur ajout droit écriture " + path); //
e = erreur
}
File parentDir = f.getAbsolutePath().getParentFile();
r = parentDir.setReadable(true, false);
if(r)
{
 Log.d("TTPA", "onCreate : Ajout droit lecture " + parentDir.getPath
()); // d = debug
}
else
{
 Log.e("TTPA", "onCreate : Erreur ajout droit lecture " + parentDir.
getPath()); // e = erreur
}
r = parentDir.setWritable(true, false);
if(r)
{
 Log.d("TTPA", "onCreate : Ajout droit écriture " + parentDir.
getPath()); // d = debug
}
else
{
 Log.e("TTPA", "onCreate : Erreur ajout droit écriture " + parentDir.
getPath()); // e = erreur
}
}

```

### 9.13.3.2 void com.ttpa.iris.ttpamobile.ServeurSQLite.onOpen ( SQLiteDatabase db )

#### Paramètres

|           |                                                          |
|-----------|----------------------------------------------------------|
| <b>db</b> | un type SQLiteDatabase qui représente la base de données |
|-----------|----------------------------------------------------------|

```

{
 //onUpgrade(db, 1, 2);

 String path = db.getPath();
 File f = new File(path);
 boolean r = f.setReadable(true, false);
 if(r)
 {
 Log.d("TTPA", "onOpen : Ajout droit lecture " + path); // d = debug
 }
 else
 {
 Log.e("TTPA", "onOpen : Erreur ajout droit lecture " + path); // e
= erreur
 }
 r = f.setWritable(true, false);
 if(r)
 {
 Log.d("TTPA", "onOpen : Ajout droit écriture " + path); // d =
debug
 }
 else
 {
 Log.e("TTPA", "onOpen : Erreur ajout droit écriture " + path); // e

```

```

 = erreur
 }
 File parentDir = f.getAbsoluteFile().getParentFile();
 r = parentDir.setReadable(true, false);
 if(r)
 {
 Log.d("TTPA", "onOpen : Ajout droit lecture " + parentDir.getPath()
); // d = debug
 }
 else
 {
 Log.e("TTPA", "onOpen : Erreur ajout droit lecture " + parentDir.
getPath()); // e = erreur
 }
 r = parentDir.setWritable(true, false);
 if(r)
 {
 Log.d("TTPA", "onOpen : Ajout droit écriture " + parentDir.getPath(
); // d = debug
 }
 else
 {
 Log.e("TTPA", "onOpen : Erreur ajout droit écriture " + parentDir.
getPath()); // e = erreur
 }
}

```

### 9.13.3.3 void com.ttpa.iris.ttpamobile.ServeurSQLite.onUpgrade ( SQLiteDatabase db, int oldVersion, int newVersion )

Méthode onUpgrade supprimant et recréant toutes les tables de la base de données.

#### Paramètres

|                   |  |
|-------------------|--|
| <i>db</i>         |  |
| <i>oldVersion</i> |  |
| <i>newVersion</i> |  |

Références [com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate\(\)](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_JOUEURS](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_PARAMETRES](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE\\_SEANCES](#).

```

{
 // On supprime la table puis on la recrée
 db.execSQL("DROP TABLE IF EXISTS " + TABLE_JOUEURS + ";");
 db.execSQL("DROP TABLE IF EXISTS " + TABLE_SEANCES + ";");
 db.execSQL("DROP TABLE IF EXISTS " + TABLE_PARAMETRES + ";");
 onCreate(db);
}

```

### 9.13.4 Documentation des données membres

#### 9.13.4.1 final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL\_DATE\_DEBUT = "DATE\_DEBUT" [static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.getSeance\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.getSeances\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

#### 9.13.4.2 final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL\_DATE\_FIN = "DATE\_FIN" [static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.getSeance\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.getSeances\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

#### 9.13.4.3 final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL\_EFFET = "EFFET" [static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.getSeance\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.getSeances\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#).

**9.13.4.4** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_FREQUENCE = "FREQUENCE"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

**9.13.4.5** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ID = "ID_SEANCE"` [static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.supprimerSeance()`.

**9.13.4.6** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ID_JOUEUR = "ID_JOUEUR"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getIdJoueurParametres()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueur()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueurs()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.purgerSeancesJoueur()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.supprimerJoueur()`.

**9.13.4.7** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ID_PARAMETRE = "ID.PARAMETRE"`  
[static]

**9.13.4.8** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_INTENSITE_EFFET = "INTENSITE_EFFET"` [static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

**9.13.4.9** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_NOM = "NOM"` [static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueur()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getJoueurs()`, `com.ttpa.iris.ttpamobile.ServeurBDD.insererJoueur()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.supprimerJoueur()`.

**9.13.4.10** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_NOMBRE_BALLES = "NOMBRE_BALLES"` [static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

**9.13.4.11** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_PUISSANCE = "PUISSANCE"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

**9.13.4.12** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ROTATION = "ROTATION"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

**9.13.4.13** `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_TAUX_REUSSITE = "TAUX_REUSSITE"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

9.13.4.14 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ZONE_OBJECTIF = "ZONE_OBJECTIF"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

9.13.4.15 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.COL_ZONE_ROBOT = "ZONE_ROBOT"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.getSeance()`, `com.ttpa.iris.ttpamobile.ServeurBDD.getSeances()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance()`.

9.13.4.16 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE_BDD_JOUEURS = "`  
`VARCHAR(255) NOT NULL);"` [static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.purgerTableJoueurs()`.

9.13.4.17 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE_BDD_PARAMETRES = "`  
`"CONSTRAINT fk_parametres_1 FOREIGN KEY (ID_JOUEUR) REFERENCES table_joueurs (ID_JOUEUR));"`  
[static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.18 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.CREATE_BDD_SEANCES = "CONSTRAINT`  
`fk_seances_1 FOREIGN KEY (ID_JOUEUR) REFERENCES table_joueurs (ID_JOUEUR) ON DELETE`  
`CASCADE);"` [static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`, et `com.ttpa.iris.ttpamobile.ServeurBDD.purgerTableSeances()`.

9.13.4.19 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.DATABASE_NAME = "ttpa_mobile.db"`  
[static]

Attributs de la classe `ServeurSQLite`.

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.ServeurSQLite()`.

9.13.4.20 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.DATABASE_VERSION = 1` [static]

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.ServeurSQLite()`.

9.13.4.21 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_1 = "INSERT`  
`INTO table_joueurs(NOM) VALUES('LEGOUT Christophe');"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.22 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_2 = "INSERT`  
`INTO table_joueurs(NOM) VALUES('MARTINEZ Michel');"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.23 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_3 = "INSERT`  
`INTO table_joueurs(NOM) VALUES('LEBESSION Emmanuel');"` [static, private]

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.24 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_4 = "INSERT INTO table.joueurs(NOM) VALUES('ELOI Damien');"` `[static, private]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.25 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_5 = "INSERT INTO table.joueurs(NOM) VALUES('MATTENET Adrien');"` `[static, private]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.26 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_6 = "INSERT INTO table.joueurs(NOM) VALUES('CHILA Patrick');"` `[static, private]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.27 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_JOUEURS_7 = "INSERT INTO table.joueurs(NOM) VALUES('BEAUMONT Jérôme');"` `[static, private]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.28 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.INSERT_TABLE_PARAMETRES = "INSERT INTO table.parametres(ID_PARAMETRE, ID_JOUEUR) VALUES(1, 1);"` `[static, private]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurSQLite.onCreate()`.

9.13.4.29 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_DATE_DEBUT = 10` `[static]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

9.13.4.30 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_DATE_FIN = 11` `[static]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

9.13.4.31 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_EFFECT = 3` `[static]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

9.13.4.32 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_FREQUENCE = 1` `[static]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

9.13.4.33 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ID = 0` `[static]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

9.13.4.34 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ID_JOUEUR = 0` `[static]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToJoueur()`.

9.13.4.35 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ID_JOUEUR_PARAMETRE = 1` `[static]`

9.13.4.36 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ID_JOUEUR_SEANCE = 12` `[static]`

Référencé par `com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance()`.

9.13.4.37 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ID_PARAMETRE = 0` `[static]`

9.13.4.38 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_INTENSITE_EFFET = 4`  
[static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

9.13.4.39 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_NOM = 1` [static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToJoueur\(\)](#).

9.13.4.40 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_NOMBRE_BALLES = 2`  
[static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

9.13.4.41 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_PUISSANCE = 5` [static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

9.13.4.42 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ROTATION = 6` [static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

9.13.4.43 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_TAUX_REUSSITE = 9`  
[static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

9.13.4.44 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ZONE_OBJECTIF = 7`  
[static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

9.13.4.45 `final int com.ttpa.iris.ttpamobile.ServeurSQLite.NUM_COL_ZONE_ROBOT = 8` [static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.cursorToSeance\(\)](#).

9.13.4.46 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE_JOUEURS = "table_joueurs"`  
[static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.getJoueur\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.getJoueurs\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.insererJoueur\(\)](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.onUpgrade\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.supprimerJoueur\(\)](#).

9.13.4.47 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE_PARAMETRES = "table_parametres"`  
[static]

Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.getIdJoueurParametres\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurSQLite.onUpgrade\(\)](#).

9.13.4.48 `final String com.ttpa.iris.ttpamobile.ServeurSQLite.TABLE_SEANCES = "table_seances"`  
[static]

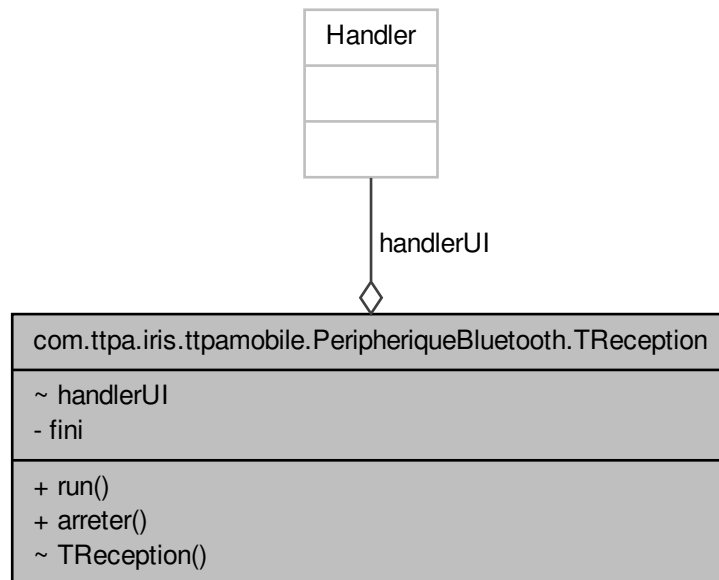
Référencé par [com.ttpa.iris.ttpamobile.ServeurBDD.getSeance\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.getSeances\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.insererSeance\(\)](#), [com.ttpa.iris.ttpamobile.ServeurSQLite.onUpgrade\(\)](#), [com.ttpa.iris.ttpamobile.ServeurBDD.purgerSeancesJoueur\(\)](#), et [com.ttpa.iris.ttpamobile.ServeurBDD.supprimerSeance\(\)](#).

La documentation de cette classe a été générée à partir du fichier suivant :

– [ServeurSQLite.java](#)

## 9.14 Référence de la classe com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception

Graphe de collaboration de com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception :



## Fonctions membres publiques

- void [run](#) ()
- void [arreter](#) ()

## Fonctions de paquetage

- [TReception](#) (Handler h)

## Attributs de paquetage

- Handler [handlerUI](#)

## Attributs privés

- boolean [fini](#)

## 9.14.1 Documentation des constructeurs et destructeur

9.14.1.1 com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.TReception ( Handler h )  
[package]

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.fini](#), et [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.handlerUI](#).

```

{
 handlerUI = h;
 fini = false;
}

```

### 9.14.2 Documentation des fonctions membres

#### 9.14.2.1 `void com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.arreter ( )`

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.fini](#).

Référencé par [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.deconnecter\(\)](#).

```

{
 if(fini == false)
 {
 fini = true;
 }
 try
 {
 Thread.sleep(500);
 }
 catch (InterruptedException e)
 {
 e.printStackTrace();
 }
}

```

#### 9.14.2.2 `void com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run ( )`

Références [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.CODE\\_DECONNEXION](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.CODE\\_RECEPTION](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.fini](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getAdresse\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.getNom\(\)](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.handler](#), [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.handlerUI](#), et [com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.receiveStream](#).

```

{
 System.out.println("<Bluetooth> Attente reception");
 while(!fini)
 {
 try
 {
 if(receiveStream.available() > 0)
 {
 byte buffer[] = new byte[100];
 int k = receiveStream.read(buffer, 0, 100);

 if(k > 0)
 {
 byte rawdata[] = new byte[k];
 for(int i=0;i<k;i++)
 rawdata[i] = buffer[i];

 String data = new String(rawdata);
 System.out.println("<Bluetooth> Reception " + data)

 Message msg = Message.obtain();
 /*msg.what = Peripherique.CODE_RECEPTION;
 msg.obj = data;
 handlerUI.sendMessage(msg);*/

 //Message msg = handlerUI.obtainMessage();
 Bundle b = new Bundle();
 b.putString("nom", getNom());
 b.putString("adresse", getAdresse());
 b.putInt("etat", CODE_RECEPTION);
 b.putString("donnees", data);
 msg.setData(b);
 handlerUI.sendMessage(msg);
 }
 }
 }
 }
}

```



```

 try
 {
 Thread.sleep(250);
 }
 catch (InterruptedException e)
 {
 e.printStackTrace();
 }
 }
 catch (IOException e)
 {
 //System.out.println("<Socket> error read");
 e.printStackTrace();
 }
}
Message msg = Message.obtain();
//msg.arg1 = CODE_DECONNEXION;
Bundle b = new Bundle();
b.putString("nom", getNom());
b.putString("adresse", getAdresse());
b.putInt("etat", CODE_DECONNEXION);
b.putString("donnees", "");
msg.setData(b);
handler.sendMessage(msg);
System.out.println("<Bluetooth> Fin reception");
}

```

### 9.14.3 Documentation des données membres

#### 9.14.3.1 boolean `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.fini` [private]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.arreter()`, `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.TReception()`.

#### 9.14.3.2 Handler `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.handlerUI` [package]

Référencé par `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.run()`, et `com.ttpa.iris.ttpamobile.PeripheriqueBluetooth.TReception.TReception()`.

La documentation de cette classe a été générée à partir du fichier suivant :

– [PeripheriqueBluetooth.java](#)

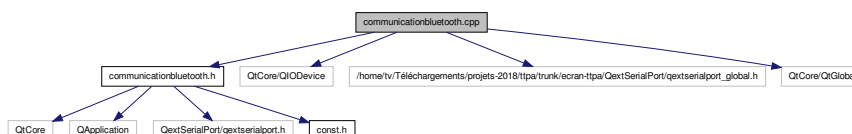
## 10 Documentation des fichiers

### 10.1 Référence du fichier `AndroidManifest.xml`

### 10.2 Référence du fichier `communicationbluetooth.cpp`

Définition de la classe [CommunicationBluetooth](#).

`#include "communicationbluetooth.h"` Graphe des dépendances par inclusion de `communicationbluetooth.cpp` :

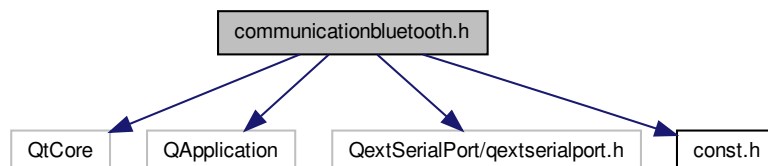


## 10.2.1 Description détaillée

## 10.3 Référence du fichier communicationbluetooth.h

Déclaration de la classe [CommunicationBluetooth](#).

```
#include <QtCore> #include <QApplication> #include "QextSerialPort/qextserialport.h"
#include "const.h" Graphe des dépendances par inclusion de communicationbluetooth.h :
```



## Classes

- class [CommunicationBluetooth](#)  
Assure la réception des trames via le Bluetooth.

## Macros

- #define [PERIODE\\_SURVEILLANCE](#) 200  
durée de la temporisation périodique en ms

## 10.3.1 Description détaillée

## 10.3.2 Documentation des macros

## 10.3.2.1 #define PERIODE\_SURVEILLANCE 200

## 10.4 Référence du fichier const.h

## Macros

- #define [WIDGET\\_SIZE\\_MAX](#) 16777215
- #define [TAILLE\\_FENETRE\\_DEFAULT\\_WIDTH](#) 960
- #define [TAILLE\\_FENETRE\\_DEFAULT\\_HEIGHT](#) 540
- #define [RATIO\\_ENTETE](#) 10
- #define [TAILLE\\_TEXTE](#) 20
- #define [TAILLE\\_TEXTE\\_SMALL](#) 10
- #define [TAILLE\\_TEXTE\\_NORMAL](#) 30
- #define [TAILLE\\_TEXTE\\_BIG](#) 42
- #define [TAILLE\\_TEXTE\\_NOM](#) 25
- #define [DELAI\\_FIXFENETRE](#) 200
- #define [DEV\\_BALLESMAX](#) 30
- #define [IHM\\_BALLESENVOYEES](#) ""
- #define [IHM\\_NOMDETEST](#) QString : :fromUtf8("Simon GAUZY")
- #define [IHM\\_PERIPHERIQUEDETEST](#) QString : :fromUtf8("Périphérique\_de\_demonstration")
- #define [NB\\_ZONES](#) 9
- #define [BALLES\\_MAX\\_DEFAULT](#) 20
- #define [DELAI\\_COUP](#) 400
- #define [HAUTEUR\\_FILET](#) 20

```

- #define TAILLE_TEXTE_NB 30
- #define TAILLE_TEXTE_NB_BIG 42
- #define TAILLE_OVERLAY 128
- #define TABLE_STAT1 QString : :fromUtf8("Hors Table :")
- #define LOGO_ATTENTECONNECTION "ATTENTE DE CONNEXION"
- #define LOGO_ATTENTECONFIGURATION QString : :fromUtf8("ATTENTE DE CONFIGURATION DE
 LA SÉANCE")
- #define LOGO_ATTENTEIDENTIFICATION "ATTENTE D'IDENTIFICATION DU JOUEUR"
- #define LOGO_JOUEUR_CONNECTE QString : :fromUtf8(" est connecté")
- #define RECAP_TITRE_TEXTE QString : :fromUtf8("FIN DE SÉANCE")
- #define RECAP_STAT1 QString : :fromUtf8("Balles Dans L'Objectif :")
- #define RECAP_STAT1_ALT QString : :fromUtf8("Balles Renvoyées :")
- #define RECAP_STAT2 QString : :fromUtf8("Balles Hors Table :")
- #define RECAP_STAT3 QString : :fromUtf8("Série Maximale :")
- #define RECAP_STAT4 QString : :fromUtf8("")
- #define CSS_TIMER_ON QString : :fromUtf8("QLabel{color : #B08000 ;}")
- #define CSS_TIMER_OFF QString : :fromUtf8("QLabel{color : #A00000 ;}")
- #define CSS_TIMER_RES QString : :fromUtf8("QLabel{color : #00A000 ;}")
- #define CSS_FOND_INACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgba(50, 150, 255,
 0) ;\\nborder : 3px solid rgba(255,255,255,30) ;\\ncolor : #FFFFFF ;\\n}")
- #define CSS_FOND_ACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgb(0, 150,
 50) ;\\nborder : 3px solid #00FF00 ;\\ncolor : #FFFFFF ;\\n}")
- #define CSS_FOND_RATE QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgb(175, 50,
 25) ;\\nborder : 3px solid #FF0000 ;\\ncolor : #FFFFFF ;\\n}")
- #define CSS_FOND_ROBOT QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgba(0, 0, 0,
 120) ;\\nborder : 3px solid #000000 ;\\ncolor : #00FF00 ;\\n}")
- #define CSS_FOND_OBJECTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgba(200, 160, 30,
 120) ;\\nborder : 3px solid #FFEE00 ;\\ncolor : #FFFF77 ;\\n}")

```

#### Énumérations

```

- enum zones_e { ZONE_HAUTGAUCHE = 0, ZONE_HAUTMILIEU, ZONE_HAUTDROITE, ZONE_MILIEU-
 UGAUCHE, ZONE_MILIEUMILIEU, ZONE_MILIEUDROITE, ZONE_BASGAUCHE, ZONE_BASMILIEU,
 ZONE_BASDROITE, ZONE_ENJEU = 15, ZONE_AUCUNE = 20 }
 Enumeration des zones de la table coté robot.
- enum layer_e { LAYER_LOGO = 0, LAYER_TABLE = 1, LAYER_RECAP = 2 }
 Enumeration des fenetres de l'IHM.
- enum etatRFCOMM_e { RFCOMM_ARRETE = 0, RFCOMM_CONNECTE, RFCOMM_FERME }
 Enumeration des etats possible du port RFCOMM.

```

#### 10.4.1 Documentation des macros

10.4.1.1 #define BALLES\_MAX\_DEFAULT 20

10.4.1.2 #define CSS\_FOND\_ACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgb(0, 150, 50) ;\\nborder : 3px solid #00FF00 ;\\ncolor : #FFFFFF ;\\n}")

Référencé par CTable : :impacterZone().

10.4.1.3 #define CSS\_FOND\_INACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgba(50, 150, 255, 0) ;\\nborder : 3px solid rgba(255,255,255,30) ;\\ncolor : #FFFFFF ;\\n}")

Référencé par CTable : :CTable(), CTable : :rafraichirCSS(), CTable : :rafraichirInactif(), CTable : :setZoneObjectif(), et CTable : :setZoneRobot().

10.4.1.4 #define CSS\_FOND\_OBJECTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgba(200, 160, 30, 120) ;\\nborder : 3px solid #FFEE00 ;\\ncolor : #FFFF77 ;\\n}")

Référencé par CTable : :rafraichirInactif(), et CTable : :setZoneObjectif().

10.4.1.5 `#define CSS_FOND_RATE QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgb(175, 50, 25);\\nborder : 3px solid #FF0000;\\ncolor : #FFFFFF;\\n}")`

Référencé par [CTable : :impacterZone\(\)](#).

10.4.1.6 `#define CSS_FOND_ROBOT QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgba(0, 0, 0, 120);\\nborder : 3px solid #000000;\\ncolor : #00FF00;\\n}")`

Référencé par [CTable : :setZoneRobot\(\)](#).

10.4.1.7 `#define CSS_TIMER_OFF QString : :fromUtf8("QLabel{color : #A00000;}")`

Référencé par [Clhm : :pauserSeance\(\)](#).

10.4.1.8 `#define CSS_TIMER_ON QString : :fromUtf8("QLabel{color : #B08000;}")`

Référencé par [Clhm : :commencerSeance\(\)](#), et [Clhm : :rafraichirTimerSeance\(\)](#).

10.4.1.9 `#define CSS_TIMER_RES QString : :fromUtf8("QLabel{color : #00A000;}")`

Référencé par [Clhm : :reprendreSeance\(\)](#).

10.4.1.10 `#define DELAI_COUP 400`

Référencé par [CTable : :impacterZone\(\)](#).

10.4.1.11 `#define DELAI_FIXFENETRE 200`

Référencé par [Clhm : :Clhm\(\)](#).

10.4.1.12 `#define DEV_BALLESMAX 30`

Référencé par [Clhm : :gererArguments\(\)](#), et [CTable : :resetSeance\(\)](#).

10.4.1.13 `#define HAUTEUR_FILET 20`

Référencé par [CTable : :setFiletTaille\(\)](#).

10.4.1.14 `#define IHM_BALLESENVOYEES ""`

10.4.1.15 `#define IHM_NOMDETEST QString : :fromUtf8("Simon GAUZY")`

Référencé par [Clhm : :setInfoConnectDemo\(\)](#).

10.4.1.16 `#define IHM_PERIPHERIQUEDETEST QString : :fromUtf8("Périférique de démonstration")`

Référencé par [Clhm : :setNomPeripheriqueDemo\(\)](#).

10.4.1.17 `#define LOGO_ATTENTECONFIGURATION QString : :fromUtf8("ATTENTE DE CONFIGURATION DE LA SÉANCE")`

Référencé par [Clhm : :connecterJoueur\(\)](#), et [Clhm : :setInfoConnect\(\)](#).

10.4.1.18 `#define LOGO_ATTENTECONNEXION "ATTENTE DE CONNEXION"`

Référencé par [Clhm : :deconnecterJoueur\(\)](#), et [Clhm : :initialisationFenetre\(\)](#).

10.4.1.19 `#define LOGO_ATTENTEIDENTIFICATION "ATTENTE D'IDENTIFICATION DU JOUEUR"`

10.4.1.20 `#define LOGO_JOUEUR_CONNECTE` QString : :fromUtf8(" est connecté")

Référencé par `Clhm : :setNomPeripherique()`.

10.4.1.21 `#define NB_ZONES` 9

Référencé par `CTable : :CTable()`, `CTable : :impacterZone()`, `CTable : :rafraichirCSS()`, `CTable : :rafraichirInactif()`, `CTable : :rafraichirNbBallesZone()`, `CTable : :resetNbBallesZone()`, `CTable : :setLayerEcran()`, `CTable : :setZoneObjectif()`, et `CTable : :setZoneRobot()`.

10.4.1.22 `#define RATIO_ENTETE` 10

Référencé par `Clhm : :initialisationFenetre()`.

10.4.1.23 `#define RECAP_STAT1` QString : :fromUtf8("Balles Dans L'Objectif :")

Référencé par `Clhm : :finirSeance()`.

10.4.1.24 `#define RECAP_STAT1_ALT` QString : :fromUtf8("Balles Renvoyées :")

Référencé par `Clhm : :finirSeance()`.

10.4.1.25 `#define RECAP_STAT2` QString : :fromUtf8("Balles Hors Table :")

Référencé par `Clhm : :finirSeance()`.

10.4.1.26 `#define RECAP_STAT3` QString : :fromUtf8("Série Maximale :")

Référencé par `Clhm : :finirSeance()`.

10.4.1.27 `#define RECAP_STAT4` QString : :fromUtf8("")

10.4.1.28 `#define RECAP_TITRE_TEXTE` QString : :fromUtf8("FIN DE SÉANCE")

Référencé par `Clhm : :initialisationFenetre()`.

10.4.1.29 `#define TABLE_STAT1` QString : :fromUtf8("Hors Table :")

Référencé par `Clhm : :initialisationStats()`.

10.4.1.30 `#define TAILLE_FENETRE_DEFAULT_HEIGHT` 540

Référencé par `Clhm : :getRatioFenetreY()`.

10.4.1.31 `#define TAILLE_FENETRE_DEFAULT_WIDTH` 960

Référencé par `Clhm : :getRatioFenetreX()`.

10.4.1.32 `#define TAILLE_OVERLAY` 128

Référencé par `CTable : :rafraichirCSS()`.

10.4.1.33 `#define TAILLE_TEXTE` 20

Référencé par `CTable : :rafraichirCSS()`, et `Clhm : :rafraichirCSS()`.

10.4.1.34 `#define TAILLE_TEXTE_BIG` 42

10.4.1.35 `#define TAILLE_TEXTE_NB 30`

Référencé par `CTable : :rafraichirCSS()`, et `CTable : :setLayerEcran()`.

10.4.1.36 `#define TAILLE_TEXTE_NB_BIG 42`

Référencé par `CTable : :rafraichirCSS()`.

10.4.1.37 `#define TAILLE_TEXTE_NOM 25`

Référencé par `Clhm : :rafraichirCSS()`, et `Clhm : :setInfoConnect()`.

10.4.1.38 `#define TAILLE_TEXTE_NORMAL 30`

Référencé par `Clhm : :rafraichirCSS()`.

10.4.1.39 `#define TAILLE_TEXTE_SMALL 10`

Référencé par `Clhm : :rafraichirCSS()`.

10.4.1.40 `#define WIDGET_SIZE_MAX 16777215`

Référencé par `CTable : :CTable()`, et `CTable : :setFileTaille()`.

## 10.4.2 Documentation du type de l'énumération

### 10.4.2.1 `enum etatRFCOMM_e`

Valeurs énumérées :

***RFCOMM\_ARRETE***  
***RFCOMM\_CONNECTE***  
***RFCOMM\_FERME***

```
{
 RFCOMM_ARRETE = 0,
 RFCOMM_CONNECTE,
 RFCOMM_FERME
};
```

### 10.4.2.2 `enum layer_e`

Valeurs énumérées :

***LAYER\_LOGO***  
***LAYER\_TABLE***  
***LAYER\_RECAP***

```
{
 LAYER_LOGO = 0,
 LAYER_TABLE = 1,
 LAYER_RECAP = 2
};
```

### 10.4.2.3 `enum zones_e`

Valeurs énumérées :

***ZONE\_HAUTGAUCHE***  
***ZONE\_HAUTMILIEU***

```

ZONE_HAUTDROITE
ZONE_MILIEUGAUCHE
ZONE_MILIEUMILIEU
ZONE_MILIEUDROITE
ZONE_BASGAUCHE
ZONE_BASMILIEU
ZONE_BASDROITE
ZONE_ENJEU
ZONE_AUCUNE

```

```

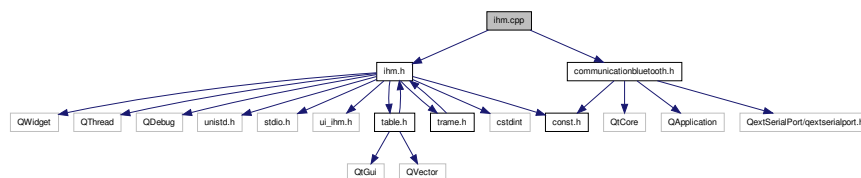
{
 ZONE_HAUTGAUCHE = 0,
 ZONE_HAUTMILIEU,
 ZONE_HAUTDROITE,
 ZONE_MILIEUGAUCHE,
 ZONE_MILIEUMILIEU,
 ZONE_MILIEUDROITE,
 ZONE_BASGAUCHE,
 ZONE_BASMILIEU,
 ZONE_BASDROITE,

 ZONE_ENJEU = 15,
 ZONE_AUCUNE = 20
};

```

## 10.5 Référence du fichier ihm.cpp

#include "ihm.h" #include "communicationbluetooth.h" Graphe des dépendances par inclusion de ihm.cpp :



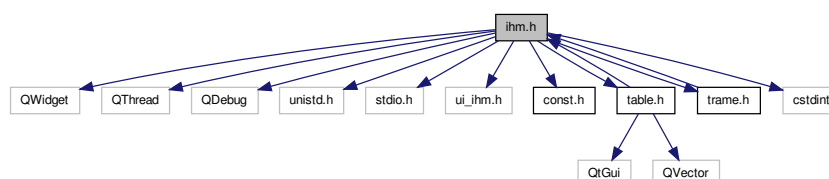
## 10.6 Référence du fichier ihm.h

La fenêtre principale de l'application.

```

#include <QWidget> #include <QThread> #include <QDebug> #include <unistd.h>
#include <stdio.h> #include "ui_ihm.h" #include "const.h" #include
"table.h" #include "trame.h" #include <cstdint> Graphe des dépendances par in-
clusion de ihm.h :

```



## Classes

- class [Clhm](#)  
*Classe principale de l'application (IHM)*

## Macros

- #define [PORT\\_BLUETOOTH](#) "rfcomm0"

## 10.6.1 Description détaillée

## Auteur

Racamond Adrien

## Version

1.0

## 10.6.2 Documentation des macros

10.6.2.1 #define [PORT\\_BLUETOOTH](#) "rfcomm0"

Référencé par [Clhm](#) : [:Clhm\(\)](#).

## 10.7 Référence du fichier IHMEcranPrincipal.java

## Classes

- class [com.ttpa.iris.ttpamobile.IHMEcranPrincipal](#)

## 10.8 Référence du fichier IHMHistoriqueSeances.java

## Classes

- class [com.ttpa.iris.ttpamobile.IHMHistoriqueSeances](#)

## 10.9 Référence du fichier Joueur.java

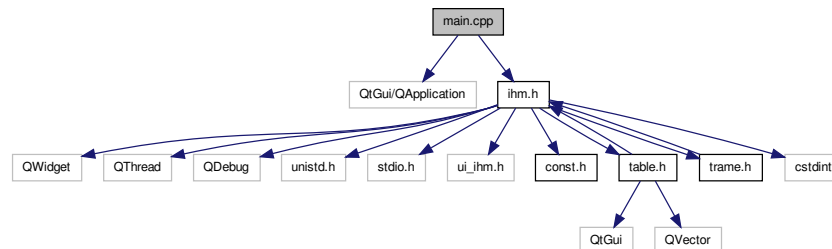
## Classes

- class [com.ttpa.iris.ttpamobile.Joueur](#)



## 10.10 Référence du fichier main.cpp

`#include <QtGui/QApplication> #include "ihm.h"` Graphe des dépendances par inclusion de main.cpp :



### Fonctions

– `int main (int argc, char *argv[])`

#### 10.10.1 Documentation des fonctions

##### 10.10.1.1 `int main ( int argc, char * argv[] )`

Référéncé par `CIhm : :connecterSignaux()`.

```

{
 QApplication a(argc, argv);

 CIhm ihm;
 ihm.show();

 return a.exec();
}

```

## 10.11 Référence du fichier ParametreSeance.java

### Classes

– class `com.tpa.iris.ttpamobile.ParametreSeance`

## 10.12 Référence du fichier PeripheriqueBluetooth.java

### Classes

– class `com.tpa.iris.ttpamobile.PeripheriqueBluetooth`  
 – class `com.tpa.iris.ttpamobile.PeripheriqueBluetooth.TReception`

## 10.13 Référence du fichier ReceveurBluetooth.java

### Classes

– class `com.tpa.iris.ttpamobile.ReceveurBluetooth`

## 10.14 Référence du fichier Seance.java

### Classes

- class [com.ttpa.iris.ttpamobile.Seance](#)

## 10.15 Référence du fichier ServeurBDD.java

### Classes

- class [com.ttpa.iris.ttpamobile.ServeurBDD](#)

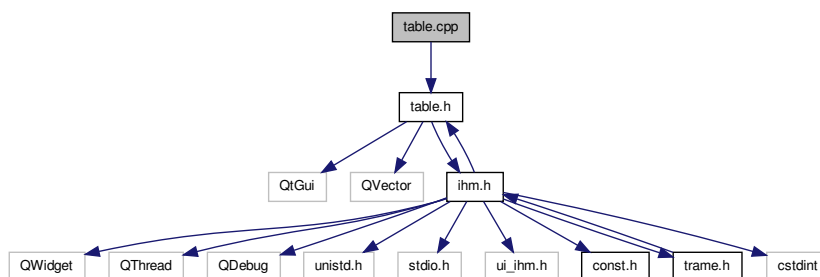
## 10.16 Référence du fichier ServeurSQLite.java

### Classes

- class [com.ttpa.iris.ttpamobile.ServeurSQLite](#)

## 10.17 Référence du fichier table.cpp

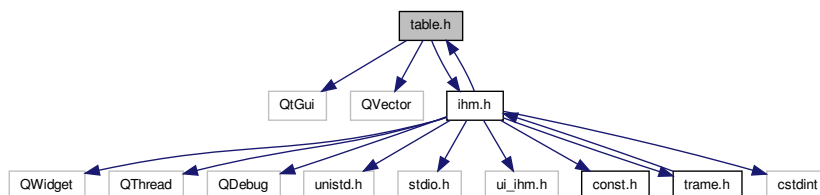
`#include "table.h"` Graphe des dépendances par inclusion de table.cpp :



## 10.18 Référence du fichier table.h

Classe gérant la table et les calculs lié a la seance, impacts, affichage des zones de la table.

`#include <QtGui> #include <QVector> #include "ihm.h"` Graphe des dépendances par inclusion de table.h :



## Classes

– class [CTable](#)

## 10.18.1 Description détaillée

## Auteur

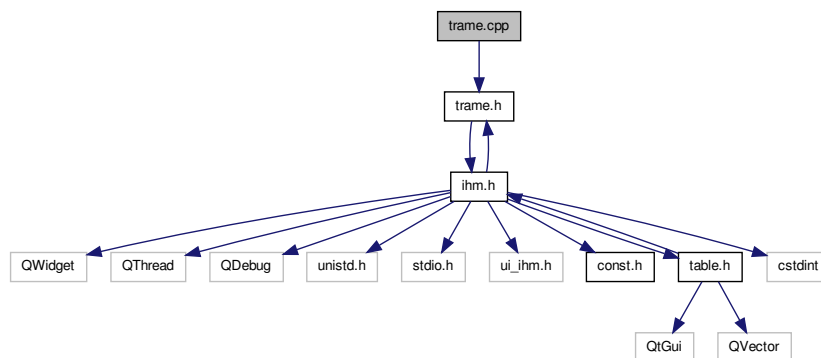
Racamond Adrien

## Version

1.0

## 10.19 Référence du fichier trame.cpp

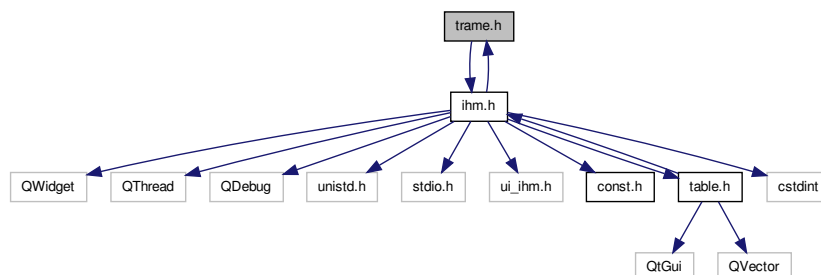
`#include "trame.h"` Graphe des dépendances par inclusion de trame.cpp :



## 10.20 Référence du fichier trame.h

Classe gérant la table et les calculs lié a la seance, impacts, affichage des zones de la table.

`#include "ihm.h"` Graphe des dépendances par inclusion de trame.h :



**Classes**

– class [CTrame](#)

**10.20.1 Description détaillée****Auteur**

Racamond Adrien

**Version**

1.0