Table des matières

Présentation

Il s'agit de réaliser le jeu du juste prix : un nombre est choisi aléatoirement entre deux bornes et le joueur doit le deviner en un minimum d'essais. A chaque essai, si le nombre n'est pas le bon, on indique au joueur si celui recherché est supérieur ou inférieur à celui énoncé.

Il sera développé en équipe de 3 ou 4 étudiants **afin de mettre en application l'utilisation de Subversion (un logiciel de gestion de versions)**.

Expression du besoin

A terme, ce jeu devra être utilisé par plusieurs personnes jouant sur des ordinateurs différents. Ainsi, pour pouvoir les mettre en compétition, un fichier contenant les meilleurs scores devra être partagé et maintenu automatiquement.

Concrètement, voici les fonctionnalités qui devront être disponibles pour le joueur :

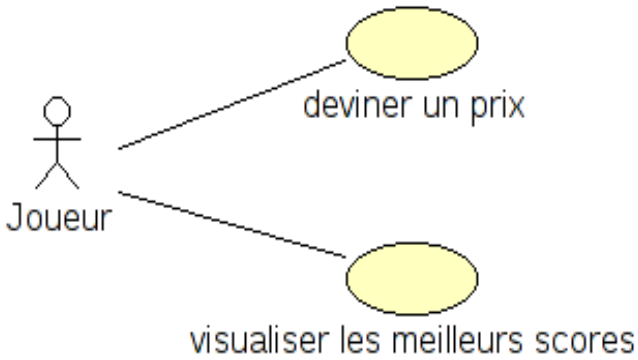
- saisir le nom du joueur
- jouer une partie
- afficher le nombre de coups utilisés pour arriver à trouver le juste prix
- afficher l'historique des parties
- afficher un classement provenant d'un fichier
- afficher la position du résultat du joueur dans le classement
- sauvegarder le résultat du joueur dans le fichier

Remarque : l'idée est issue d'un document de Michael Jégat (Corexpert) .

Moyens préliminaires disponibles et contraintes de réalisation

Spécifications

Le **diagramme des cas d'utilisation** (*Use Case*) du système est le suivant :



Ce qu'il faut retenir : Un diagramme des cas d'utilisation décrit les fonctionnalités attendues du système du point de vue d'un utilisateur. Les Cas d'Utilisation (CU) recentrent l'expression des besoins sur les utilisateurs. Les cas d'utilisation sont donc très utiles pour représenter ce que doit faire un système par rapport à son environnement.

Vous devez être capable, vis à vis d'un diagramme de cas d'utilisation, de le lire, le commenter et l'expliquer au regard des fonctionnalités décrites dans le cahier des charges. Vous devez pouvoir aussi le modifier et le compléter localement.

Pour vous aider, on vous fournit une ébauche du **diagramme de séquence** « jouer une partie » (c'est la méthode **jouer()** de la classe **IHMPartie**) :

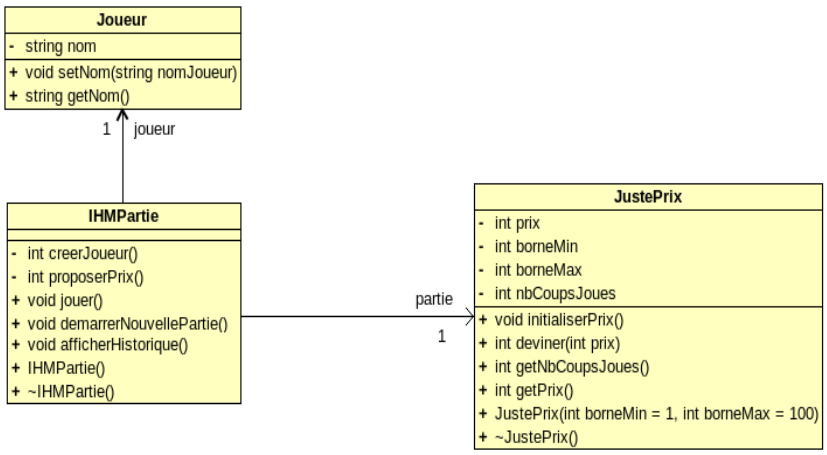
Ce qu'il faut retenir : Un diagramme de séquence est un diagramme d'interaction. Le but est de décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument. Il décrit un scénario d'un cas d'utilisation.

*Un diagramme de séquence représente donc les interactions entre objets, en insistant sur la chronologie des **envois de message**. C'est un diagramme qui représente la structure dynamique d'un système car il utilise une représentation temporelle. Les objets, intervenant dans l'interaction, sont matérialisés par une « **ligne de vie** », et les messages échangés au cours du temps sont mentionnés sous une forme textuelle. Les messages sont des **méthodes d'une classe** et l'envoi d'un message correspond donc à l'**appel de cette méthode**.*

Vous devez être capable de lire, commenter et compléter un diagramme de séquence à partir d'expressions textuelles et / ou de la définition des objets.

Vous devez aussi être capable de produire du code à partir d'un diagramme de séquence.

Le **diagramme de classes** pour la version 1 pourrait être le suivant :



Ce qu'il faut retenir : Les diagrammes de classes (et d'objets) représentent la structure statique d'un système : les classes, (les objets,) leurs structures internes (attributs et méthodes) et leurs relations.

Vous devez être capable de lire, commenter et compléter un diagramme de classes en s'appuyant sur les dossiers de spécification, de conception préliminaire et les documentations techniques. La compétence terminale visée est alors : C3.9.

Vous devez aussi être capable de d'identifier et d'interpréter les éléments pertinents d'un diagramme de classes de manière à pouvoir traduire sous la forme de code orienté objet les résultats de la conception détaillée. Par exemple, vous codez tout ou partie d'une méthode.

*Remarque : **joueur** et **partie** sont des attributs de la classe **IHMPartie**. Ils représentent les 2 relations permanentes qui existent entre les classes.*

Contrainte de développement

On appliquera un **cycle de développement itératif et incrémental**.

Il sera développé en trois itérations contenant les fonctionnalités suivantes :

- Version 1.0 :
  - saisir le nom du joueur
  - jouer une partie
  - afficher le nombre de coups joués pour trouver le juste prix
  - *afficher l'historique des parties*

*Ce qu'il faut retenir : Un développement itératif s'organise en une série de développement très courts de durée fixe nommée itérations. Le résultat de chaque itération est un **système partiel exécutable, testé et intégré** (mais incomplet). Comme le système croît avec le temps de façon incrémentale, cette méthode de développement est nommée **développement itératif et incrémental**.*

- Version 2.0 :
  - afficher un classement provenant d'un fichier
  - afficher la position du résultat du joueur dans le classement

- Version 3.0 :
  - sauvegarder le résultat du joueur dans le fichier

Contrainte de l'environnement

Système d'exploitation : choix libre (Linux ou Windows)
Environnement de développement : choix libre
Atelier de génie logiciel : bouml
Logiciel de gestion de versions : **subversion**

Contrainte économique

Aucune

Documents et moyens technologiques mis à disposition

Documents :

- des tutoriels et documentations sur Subversion (dans le répertoire doc)

Ref.	**Description**
tutoriel-riouxSVN.odt tutoriel-riouxSVN.pdf	tutoriel au format ODT et PDF présentant la mise en œuvre de subversion chez l'hébergeur spécialisé **RiouxSVN** (http://riouxsvn.com/)
svn-refcard.pdf	le « *Subversion Quick Reference Card* » contenant une description de l'ensemble des sous commandes et options de la commande **svn**.
imerir/	Dossier contenant un ensemble de cocuments au format PDF sur Subversion réalisés par **Michael Jégat** (Corexpert) pour l'école d'ingénieur IMERIR
svn-exemple-pratique.odt svn-exemple-pratique.pdf	document au format ODT et PDF présentant un exemple simple et détaillé de l'utilisation de subversion par deux développeurs

- cours et exemples sur le gestion de fichiers en C++ (`cours-c-c++-fichiers.pdf` et `exemples-fichier.zip`)
- une annexe sur les fichiers de gestion de projet (`Changelog`, `TODO` et `README`)

Moyens technologiques :

- Accès Internet
- vidéo-projecteur avec écran interactif et rétro-projecteur

Exigences qualité à respecter

Exigences qualité sur le produit à réaliser

Le produit à réaliser doit répondre aux facteurs de qualité suivants :

- maniable : il sera facile d'emploi avec une interface homme-machine simple et conviviale
- robuste : il conservera un fonctionnement conforme aux spécifications après un arrêt
- normal ou d'urgence; garantira la validité des informations échangées.
- maintenable : il offrira une bonne facilité de localisation et correction des problèmes résiduels.
- adaptabilité : il facilitera la suppression, l'évolution de fonctionnalités existantes ou l'ajout de nouvelles fonctionnalités
- portabilité : il minimisera les répercussions d'un changement d'environnement logiciel et matériel

Exigences qualité sur le développement

En ce qui concerne les exigences qualité du développement :

- La modélisation UML doit être réalisée avec un atelier de génie logiciel (bouml)
- L'architecture du logiciel sera Orientée objet.
- Le codage doit respecter le standard C/C++ en cours dans la section
- Un utilitaire de compilation automatisé de type « make » sera utilisé
- Le gestionnaire de gestion de versions utilisé sera **subversion**

Exigences qualité sur la documentation à produire

Les exigences qualité à respecter, relativement aux documents, sont :

- sur leur forme : respect de normes et de standards de représentation, homogénéité, lisibilité, maintenabilité ;
- sur leur fond : complétude, cohérence, précision.

Exigences qualité sur la livraison

Les produits à mettre à disposition du client sont :

- la documentation (fichiers `Changelog`, `TODO`, `README` et `configuration.txt`) ;
- les codes sources de l'application de la dernière version livrable, ainsi que le fichier de type Makefile.

Ces produits seront livrés sous forme informatique regroupés dans une archive au format `tar.gz` ou `zip`. Le nom de l'archive sera formaté de la manière suivante : **`mp1-teamN-vX.Y.tar.gz`** ou **`mp1-teamN-vX.Y.zip`** où :

- N représente le numéro de l'équipe de développement (numéro donné par l'enseignant)
- X le numéro de version majeur de l'application
- Y le numéro de version mineur de l'application (0 par défaut)

Exigences qualité sur l'environnement d'exploitation

Aucune

Planification des tâches

Ref.	Description	L	M	M	J	V	S	D	L	M	M	J	V
T2.1	Interprétation des spécifications logicielles et identification des fonctions principales												
T2.3	Conception de l'architecture des interfaces homme - machine												
T2.8	Utilisation des diagrammes de classe pour produire une maquette de l'application												
T3.3	Codage et assemblage des modules logiciels (dans le respect des standards entreprise)												
T3.4	Fabrication de modules logiciels réutilisables et réalisation de la documentation associée												
T3.7	Élaboration de documents de suivi de réalisation et de codage												
T8.1	Intégration et travail dans une organisation par projet												
T8.5	Renseignement des indicateurs permettant le suivi d'un projet												
T8.7	**Gestion des évolutions des versions successives des logiciels et des documents produits**												
T9.1	Intégration et travail en équipe												
T9.2	Exposé et argumentation des choix de conception, des choix techniques et des résultats de travaux auprès du service concerné ou du client (communication écrite et orale)												

Travail à réaliser

Étape n°1 : communiquer, organiser et planifier

Après avoir formé des équipes de 3 personnes, la première étape va consister à désigner un chef de projet. Cette personne sera responsable de la gestion de projet et servira d'intermédiaire avec le client. Une fois le chef de projet désigné, vous devez réfléchir au projet, aux fonctionnalités et aux choix techniques (structure de données, algorithmes, ...).

Vous devez aussi définir une organisation de travail (dépôt Subversion, gestion des *bugs*, convention de nommage, ...). Enfin, vous devez planifier les tâches et les attribuer aux membres de l'équipe de projet.

Toutes ces informations devront être écrites sous format informatique (les fichiers `Changelog`, `TODO` et `README`) et devront être disponibles à chaque membre du projet.

A la fin du temps attribué à cette étape, le chef de projet présentera brièvement son plan d'action et des questions pourront être posées au client concernant les choix techniques, le besoin, etc.

Production attendue à la fin de cette étape :

- Contenu minimal du fichier `README` à la fin de cette étape :
  - Nom du logiciel :
  - Date de début :
  - Objectif :
  - Équipe de développement : nom, prénom et <adresse courriel> de chaque membre
  - Ce que le logiciel doit faire : description détaillée des fonctionnalités offertes

Au fur et à mesure de l'avancement, vous ajouterez les informations suivantes :

  - Numéro de version du logiciel :
  - Date de cette version du logiciel :
  - Ce que le logiciel fait dans cette version :
  - Défauts constatés non corrigés :

- Contenu minimal du fichier `TODO` à la fin de cette étape : la liste des tâches et leur attribution

- Contenu minimal du fichier `Changelog` à la fin de cette étape : aucun

Étape n°2 : préparer et installer

Vous devez préalablement installer et préparer votre environnement de développement (éditeur et chaîne de fabrication `make`/`g++`).

Pour la mise en œuvre du logiciel de gestion de versions (subversion), vous devez suivre le tutoriel fourni. Une fois les comptes créés, vous devez les communiquer à l'administrateur.

Vous devez assurer une gestion de configuration. Pour cela, vous allez créer un fichier `configuration.txt` contenant pour chaque membre de l'équipe l'identification de votre plate-forme de développement : indiquer précisément le nom et la version de chaque outil utilisé ainsi que votre système d'exploitation.

Le chef de projet pourra alors mettre à jour dans votre dépôt subversion les fichiers `Changelog`, `TODO` et `README` créés à l'étape n°1 et le fichier `configuration.txt`.

Production attendue à la fin de cette étape :

- la chaîne de fabrication est fonctionnelle
- le dépôt subversion est utilisable par chaque membre de l'équipe
- le dépôt subversion est à jour (fichiers `Changelog`, `TODO`, `README` et `configuration.txt`)

Étape n°3 : concevoir et réaliser

Cette étape va consister à réaliser le projet en suivant un développement itératif et incrémental.

Pour l'utilisation du logiciel de gestion de versions (subversion), vous devez vous référer au tutoriel fourni.

Lorsqu'une version est terminée, vous pouvez contacter le client qui jouera au juste prix. Si des défauts (*bugs*) sont trouvés lors de l'utilisation, ils devront être corrigés dans les plus brefs délais. Si plus aucun défaut (*bug*) n'est trouvé, la version sera acceptée définitivement et la version suivante pourra être testée.

Ainsi, une version possède trois états :

- en développement : la version est en cours de développement (***trunk***).
- livrée : la version est livrée chez le client qui l'utilise (***tags***).
- acceptée : le client a utilisé la version livrée et n'a plus détecté de problèmes d'utilisation. Lorsqu'une version est acceptée, la version suivante peut être livrée si elle est terminée.

Il vous faudra maintenir continuellement les fichiers `Changelog`, `TODO` et `README` (ces fichiers faisant partie d'une version livrable au client). La description de ces fichiers est fournie en Annexe.

Production attendue à la fin de cette étape :

- un version livrable a été au minimum produite par l'équipe de développement
- le dépôt subversion est à jour (fichiers sources, `Changelog`, `TODO`, `README` et `configuration.txt`)

Étape n°4 : présenter oralement

Vous présenterez oralement :

- le diagramme de classes final
- les fonctionnalités validées
- les fonctionnalités non implémentées
- les défauts constatés non corrigés
- une démonstration de l'application

Grille d'évaluation

Activité :		Livraison finale		
Équipe n°__ :		Version : ____		
Critères		-	0	+
Gestion de projet	Respect de la méthodologie de développement itérative			
	Respect du travail en équipe			
	Respect du cahier des charges			
	Respect et suivi de la planification			
	Les fichiers README, TODO et Changelog existent et sont correctement renseignés			
	Les fichiers (sources, TODO, …) sont accessibles et à jour sur le serveur **subversion**			
Oral	Qualité de la présentation, précision, rigueur, clarté			
	Utilisation des moyens mis à la disposition et du vidéo-projecteur			
Démonstration	Qualité de la démonstration orale : précision, rigueur, clarté, ...			
Application	L'application livrée respecte les contraintes			
	L'application livrée respecte les exigences qualité			
Entretien	Capacité à répondre avec pertinence, précision et exactitude			
	Capacité à rechercher et à exploiter une documentation			
	Capacité à argumenter et à réagir aux objections			
Bilan	État d'avancement			
Remarques :				

Annexe : les fichiers de gestion de projet

Changelog

Un ChangeLog, littéralement **Journal des modifications** (en anglais), désigne souvent un fichier qui contient l'énumération de ce que les personnes collaborant à un projet ont effectué comme travail sur ce dernier. Ce fichier est souvent un simple fichier texte, assez brut, avec éventuellement des sections correspondant aux différentes sous-parties du projet. On peut également y trouver des noms de personnes qui ont réalisé ces tâches. Ce terme est directement issu du monde des développeurs de logiciels, notamment celui des développeurs de logiciels libres, afin que tout le monde puisse savoir dans quelle direction le projet a évolué à travers le temps, quelle est sa vitalité (s'il avance beaucoup et vite, s'il est en plein essor ou abandonné depuis un an, ou seulement en phase de corrections de bugs). Dans tous les cas, cela constitue également une invitation à contribuer en sachant quelles sont les dernières évolutions, ce qu'il y a à tester...

TODO

Une *todo list* (ou **une liste des choses à faire**) est un procédé simple et efficace qui permet de se concentrer sur une tâche d'un projet sans pour autant perdre de vue les autres tâches à accomplir. Les listes à faire se déclinent de multiples façons : par exemple, un chef de projet qui note les bogues à corriger et les fonctionnalités à programmer construit une *todo list*. Plus trivialement, un post-it avec une liste de courses à faire est aussi une *todo list*.

Une *todo list* désigne parfois un fichier qui contient l'énumération de ce que les personnes se fixent comme tâches à réaliser. Ce fichier est souvent un simple fichier texte, assez brut, avec éventuellement des sections correspondant à différents domaines d'action ou aux différentes sous-parties d'un projet.

Pour le travail en équipe, on peut également y trouver des noms de personnes à qui ces tâches échoient.

Les éléments présents dans une *todo list* sont généralement biffés une fois réalisés, afin de mesurer rapidement l'avancement global. Pour des projets logiciels, ces éléments devraient à terme se retrouver dans le journal des améliorations et modifications apportées au programme ChangeLog.

README

Un fichier readme (**lisezmoi** en français) est, en informatique, un fichier contenant des informations sur les autres fichiers d'un répertoire.

Un tel fichier est généralement un fichier texte appelé « README.TXT », « README.1ST », « READ.ME » ou simplement « **README** », parfois localisé dans les distributions françaises en « LISEZMOI.TXT », etc. Son contenu varie mais inclut d'ordinaire des instructions d'exploitation, une liste des noms et utilités des autres fichiers, des informations sur la personne les ayant créés, la version de l'application, sa description, voire la licence applicable.

