Fiche méthodologique de projet

C.12 Mettre en oeuvre des patterns en conception détaillée

Objectif

Utiliser des patterns GOF pour la conception détaillée.

Préambule

Une fois l'analyse terminée, on sait donc « quoi faire ». Maintenant en conception, on doit déterminer « comment le faire ». L'étape de conception peut se décomposer en deux parties : la conception préliminaire et la conception détaillée.

Les patterns

Dans le monde de l'orienté-objet, les design patterns se présentent comme un catalogue de méthodes de résolution de problèmes récurrents.

Un pattern ou motif de conception est un document qui décrit une solution générale à un problème qui revient souvent.

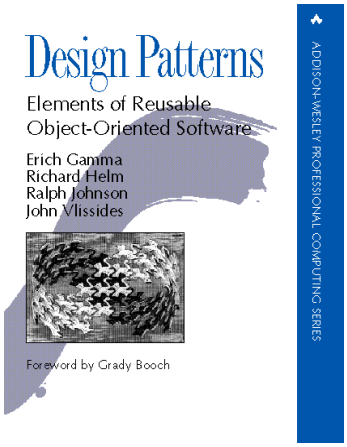
Les patterns GOF

Le concept de design patterns a été développé dans un ouvrage de référence publié en 1995 par quatre auteurs, le "Gang of Four" : Erich Gamma, Richard Helm, Ralph Johnson, et John Vlissides. Il y identifiait 23 motifs de conceptions, chacun offrant une solution à un problème récurrent de la conception orientée-objet. En voici quelques uns répartis en trois catégories :

Création : Abstract Factory, Builder, Factory Method, Prototype, Singleton

Structure : Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy

Comportement : Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor



Méthode

Il est conseillé de passer un peu de temps à étudier les fameux « design patterns ». N'abusez cependant pas des patterns au risque de complexifier inutilement vos conceptions.

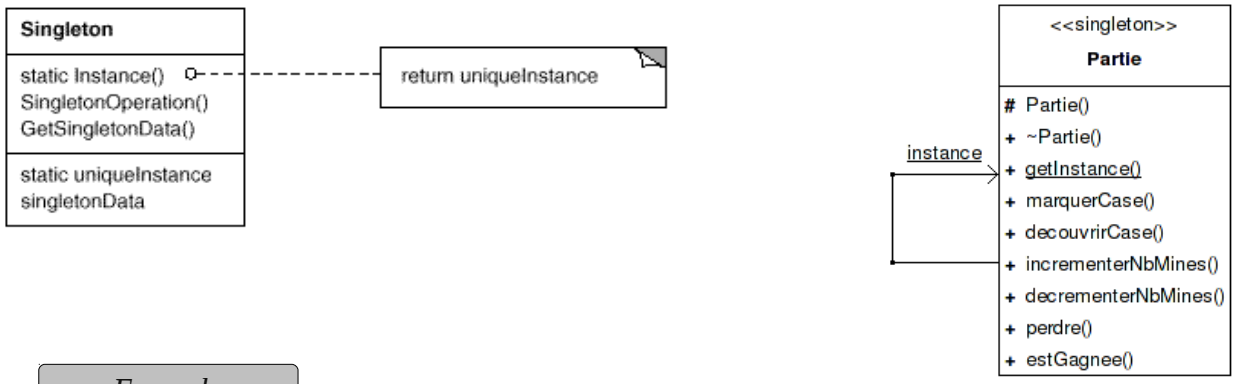
Il est toutefois vraisemblable que vous risquez de rencontrer un de ces patterns : Composite, Observateur, Stratégie, État, Singleton, Template Method, Adaptateur, Façade et Fabrique abstraite ...

Description d'un pattern

Nom et classification : par exemple observateur/comportement
Intention : description générale et succincte
Alias : autres noms connus pour le pattern
Motivation : exemples qui montrent pourquoi on a besoin du pattern
Indications d'utilisation : une liste des situations qui justifient l'utilisation du pattern
Structure : un diagramme de classe
Constituants : explication des différentes classes qui interviennent dans la structure du pattern
Implémentation : les principes, pièges, astuces, techniques pour implanter le pattern dans un langage objet
Utilisations remarquables : des programmes réels dans lesquels on trouve le pattern
Limites : les limites concernant l'utilisation du pattern

Pattern Singleton

L'objectif du pattern Singleton est de garantir qu'une classe n'a qu'une seule instance. La meilleure solution consiste à confier à la classe elle-même la responsabilité d'assurer l'unicité de son instance.



Exemple

Plusieurs classes vont avoir besoin d'un accès au contrôleur Partie, pour incrémenter et décrémenter le compteur de mines, pour signifier la fin de la partie, ... La classe Partie ne doit avoir qu'une seule instance à la fois : on joue une partie unique. C'est un problème connu de la programmation OO et la solution est d'utiliser un pattern Singleton. C'est le plus connu et le plus simple des patterns GoF.

On va appliquer ce modèle à la classe Partie :

```
class Partie
{
private:
    static Partie* instance; //membre static pour exister indépendamment
protected:
    Partie(); //interdit l'accès au constructeur de l'extérieur de la classe
public:
    ~Partie();
    static Partie* getInstance(); //pour accéder au membre static instance
};
Partie* Partie::instance = NULL; //initialisation de l'instance

Partie* Partie::getInstance() {
    if(instance == NULL) //aucune instance ?
        instance = new Partie(); //alors on instancie
    return instance; //on retourne l'instance
}
```

Et pour les classes qui auront besoin d'accéder au contrôleur Partie :

```
Partie *partie = Partie::getInstance(); //récupère l'instance unique
```