Université Paris Diderot Langages à Objets Avancés

Projet C++ : Design Patterns et Méta-programmation en C++

November 26, 2012

1 But

Le but de ce projet et de développer une librairie c++ qui offre des solutions de programmation génériques, réutilisables et optimisés à la façon de la méta-programmation et des design patterns.

2 Programmation générique

La programmation générique passe par :

- Une implémentation efficace et typée des conteneurs
- La méta-programmation
- Les des Design Patterns

2.1 Design patterns

"Design patterns are efficient and elegant solutions to common problems in object-oriented software design. They are high-level abstract templates that can be applied to particular kinds of design problems."

2.1.1 Exemple de design patterns en c++

1. Le prototype : un prototype est une classe dont le but est d'être clonée.
2. le singleton: permet de s'assurer qu'il n'existe qu'une unique instance d'une classe donnée.

3. la fabrique : classe dont le rôle est de créer d'autres objets.

4. Les décorateurs : sont l'ensemble des classes permettant d'étendre dynamiquement le rôle d'une classe de base.

5. Le composite: est un pattern qui permet de manipuler un ensemble d'objets comme un seul.

6. L'observateur : est une classe dont le rôle est d'être averti quand l'une des classes qu'elle observe change.

7. La stratégie : pattern permettant de changer d'algorithme utilisé de façon dynamique.

8. Le Visiteur permet une séparation précise entre données et traitements. Il est le compagnon idéal du pattern Composite. (figure1)

2.2 Métaprogrammation

"La métaprogrammation est la programmation de métaprogrammes : Elle est l'écriture de programmes qui manipulent des données décrivant des programmes "

2.3 Les conteneurs

Une classe containeur est une classe qui contient d'autres objets.

3 Travail demandé

3.1 Modélisation

1. Dans un premier temps on se propose d'élaborer le diagramme de classe pour les différents design patterns cités ci-dessus.

2. Dans un deuxième temps, concevoir une librairie composée de modèles génériques (en utilisant les template) représentant les différentes relations existantes entre les classes ainsi que les concepts de généricité au sein d'une même classe notamment : L'association, la composition, l'agrégations, la spécialisation ,héritage, dérivation publique, redéfinition et surcharge, la relation d'amitié, la factorisation conceptuelle, généralisation, factorisation d'implémentation, classes abstraites, classes partielles, méthodes virtuelles, l'héritage multiple.

3.2 Implémentation

1. Implémenter vos modèles en c++.

2. Ecrire les jeux de test nécessaires.

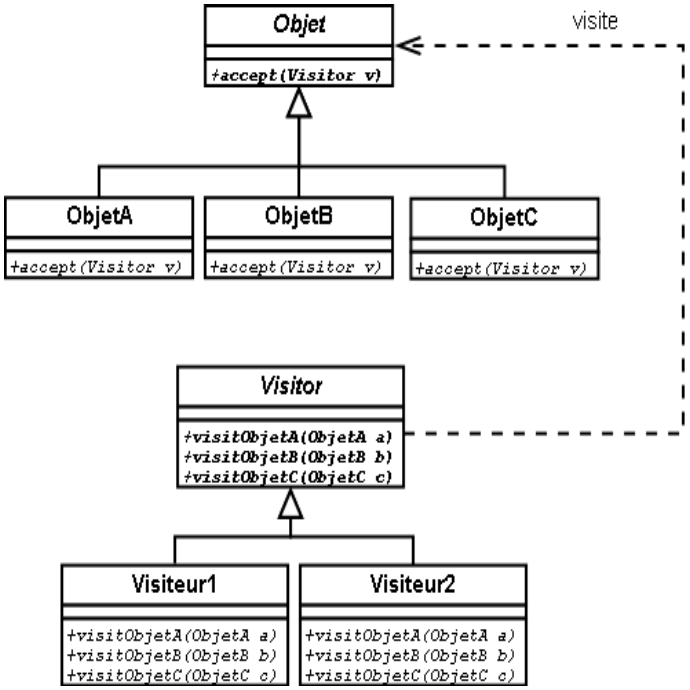

Figure 1: Le Pattern Visitor

3. Utiliser votre librairie pour développer un jeu de carte (de votre choix) permettant de mettre en avant des cas d'utilisation de votre librairie c++.

4 Bibliographie

`http://pcaboche.developpez.com/article/design-patterns/programmation-modulaire/?page=sommaire`
`http://frog.isima.fr/antoine/template.shtml` `http://pcaboche.developpez.com/article/design-patterns/programmation-modulaire/?page=page_5#L2.2.1`

5 Organisation

Il sera demandé le jour de la soutenance :

- de fournir des explications raisonnées sur les choix opérés
- de produire des schémas UML obtenus à l'aide d'un logiciel quelconque de modélisation, pas à la main de la structuration objet
- d'être capable de commenter l'implémentation réalisée

- d'être capable de faire une démonstration fonctionnelle du programme (on entend par démonstration fonctionnelle : une exécution qui ne plante pas et qui fournit un résultat raisonnable).

Le projet peut être réalisé au plus par trinôme et au moins par binôme, sans dérogation. Chaque intervenant devra rendre compte de sa contribution dans la production finale. Les équipes projet doivent être constituées à l'avance, déclarées au plus tard le vendredi 14 décembre 2012 auprès du responsable du cours (par mail faisant état des noms et prénoms des individus concernés) et, leur constitution ne pourra être modifiée sauf cas de force majeure.
Le(s) programme(s) devront impérativement être écrits en langage C++, et être exécutables sur une machine de l'UFR, ou sur son propre ordinateur portable.
Un rapport de quelques pages devra être envoyé aux chargés de TD . Ce rapport devra expliciter clairement l'ensemble des points les plus importants du projet. Il devra aussi contenir le code source du projet (ne pas hésiter à utiliser les commentaires pour expliquer vos déclarations et penser à utiliser des noms de variables assez parlants).
Tout choix d'implémentation devrait être justifié (les droits d'accès(public,protected ..), les structures de stockage , héritage ...)

La soutenance est obligatoire (sauf pour les dispenses officielles), toute absence conduira le jury à délivrer la note 0. Chaque personne devra intervenir (les silences seront jugés très négativement).