ÉCOLE POLYTECHNIQUE DE MONTRÉAL

DÉPARTEMENT DE GÉNIE INFORMATIQUE

LOG2410 — Conception logicielle

Examen final — Hiver 2009
Documentation : aucune
Calculatrice : aucune
Date : 30 avril 2009
Cet examen comprend 4 questions sur 5 pages
Répondre sur le cahier de réponses

Question 1 — Analyse (16 pts)

Le comité étudiant Poly-Sport de Polytechnique vous approche, vous et votre équipe de concepteur, pour le développement d'un outil permettant à ses membres de comparer leurs pronostiques sur différents événements sportifs dont les séries de la ligue nationale de hockey. Cet outil permettra à des usagers de créer des équipes virtuelles à partir des joueurs des équipes réelles (pensez aux « pools » d'hockey). Les usagers seront classés selon le résultat de leur équipe virtuelle.
Poly-Sport vous demandent donc de faire l'analyse et la conception de ce système qui portera le nom de POLY_POOL. Certains membres de votre équipe ont rencontré les principaux intervenants du projet pour définir les différentes caractéristiques de leur nouvel outil :

I – Configuration matérielle

Le système sera accessible à travers une interface Web, qui lui donnera une plus grande flexibilité. Un serveur central permettra d'identifier les usagers et d'accéder à une base de données contenant différentes statistiques sur les joueurs et sur les équipes réelles. Cette base de données contiendra également les informations personnelles sur les usagers. Le système POLY_POOL sera utilisé avant, pendant et après les compétitions sportives.

II – Utilisation du système pour la préparation d'une nouvelle compétition

Avant le début officiel d'une compétition sportive réelle, certains usagers désignés par Poly-Sport auront accès à une interface administrateur pour créer une compétition virtuelle et imposer certaines règles que les usagers devront suivre. Par exemple, le créateur d'une compétition pourrait imposer que les joueurs des équipes de hockey ne puissent être associés qu'à une seule équipe virtuelle. Par la suite, des usagers pourront s'inscrire et composer leurs équipes. Pour les aider dans leur choix, les usagers auront

accès aux statistiques de tous les joueurs disponibles. Les usagers pourront également modifier leurs informations personnelles, tel que leur nom d'usager et leur mot de passe. Finalement, les administrateurs pourront faire une gestion des participants d'une compétition en cas de litige.

III – Utilisation du système pendant la compétition

Les administrateurs pourront mettre à jour les statistiques des joueurs au fur et à mesure du déroulement de la compétition (nombre de points marqués, de parties jouées…). Ces statistiques détermineront le classement des usagers participant à une compétition virtuelle. Le classement et les statistiques seront accessibles par tous les usagers.

IV – Utilisation du système après la compétition

Lorsqu'une compétition prend fin, les administrateurs déclarent officiellement quel usager a obtenu les meilleurs résultats et l'inscrivent aux tableaux des grands gagnants.

Sur la base de ces notes, et en vous fiant à votre expertise en matière d'élection, on vous demande de :

a) (5 pts) Tracer un diagramme de cas d'utilisation de haut niveau (diagramme de contexte) pour le système POLY_POOL.

Réponse:

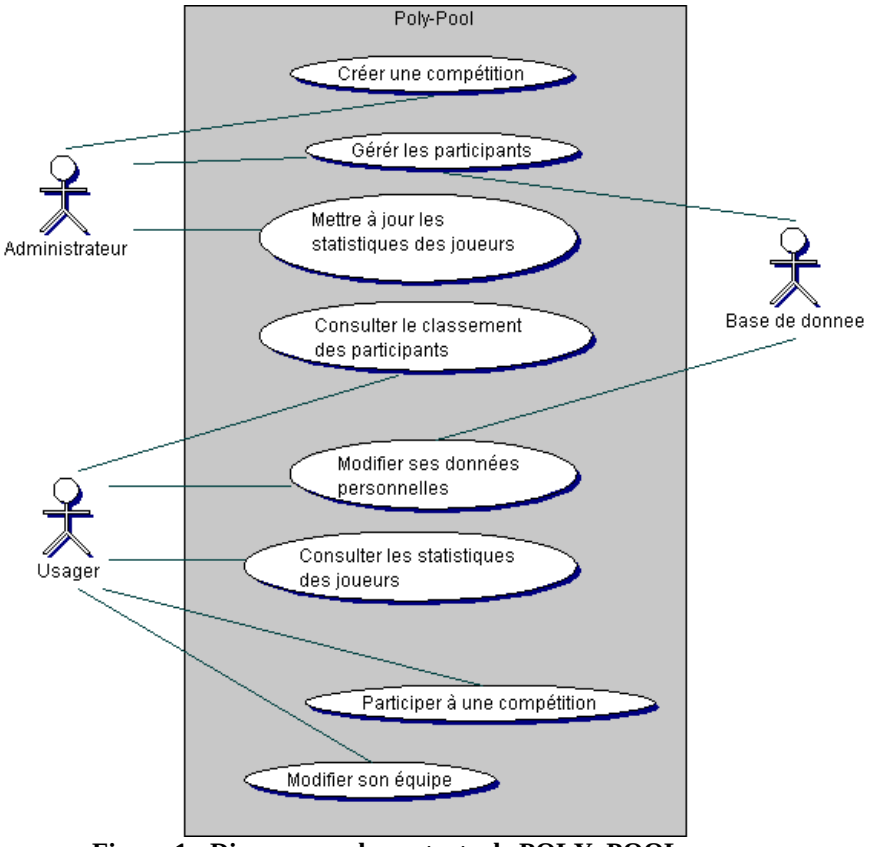

Figure 1 : Diagramme de contexte de POLY_POOL

b) (5 pts) Tracer un diagramme de concepts pour le système POLY_POOL.

Réponse:

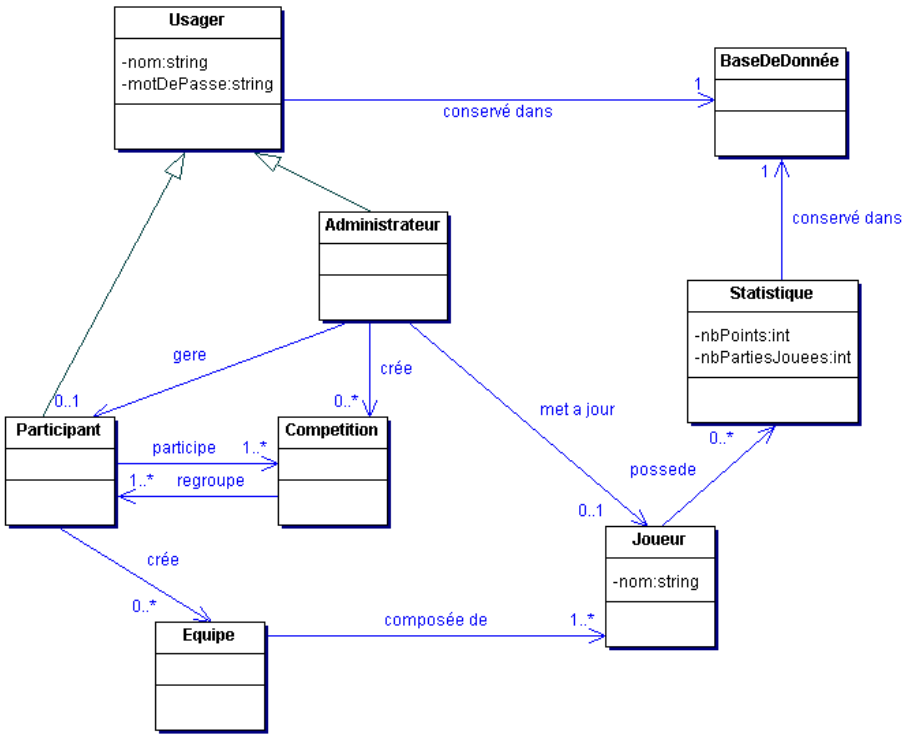

Figure 2 : Diagramme de concepts de POLY_POOL

c) (4 pts) Plusieurs types de relations peuvent être utilisés pour lier des concepts ou des classes. Identifiez quatre types de relations différentes en donnant un exemple concret illustrant dans quelle situation elles s'appliquent.

Réponse:

Tableau 1 : Type de relation entre deux classes

Association (agrégation, composition)	Relation de possession entre deux objets (ex. attribut de classe)
Généralisation	Relation de généralisation entre deux classes
Dépendance	Une dépendance peut représenter l'utilisation d'un type d'objet dans la signature d'une fonction.
Réalisation	Une classe réalisant la définition d'une interface.

d) (2 pts) Tracer un diagramme de déploiement pour le système POLY_POOL.

Réponse:

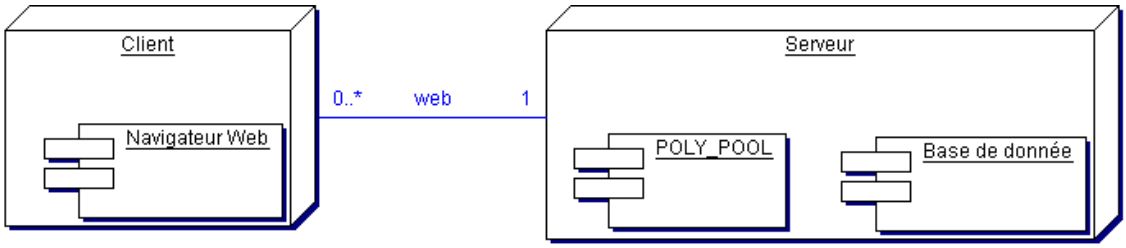

Figure 3 : Diagramme de déploiement de POLY_POOL

Question 2 – Patrons de conception (18 pts)

En considérant la mise en situation de la question précédente, vous constatez que les usagers peuvent s'inscrivent à plusieurs compétitions et qu'une compétition regroupe plusieurs usagers. Par souci d'efficacité, vous réaliser qu'il faut maintenir une relation bidirectionnelle entre les usagers et les compétitions. Par contre, ce type de référence est difficile à changer de façon non invasive.

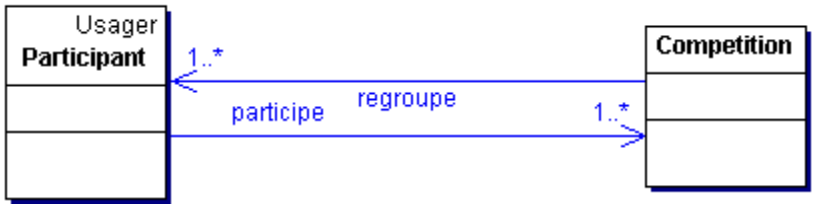

Figure 4 : Diagramme de classe réduit

Pour faciliter votre travail, vous décider d'utiliser le patron Médiateur.

a) (1 pt) Donnez, en vos propres mots, l'intention du patron Médiateur

Réponse:

Définir un objet qui encapsule comment un ensemble d'objets interagissent afin de promouvoir un couplage faible et de laisser varier l'interaction entre les objets de façon indépendante.

b) (1 pt) La classe agissant comme médiateur est souvent implantée en tant que singleton. Donnez, en vos propres mots, l'intention du patron Singleton.

Réponse:

S'assurer qu'il ne soit possible de créer qu'une seule instance d'une classe, et fournir un point d'accès global à cette instance.

c) (3 pts) Modifiez le diagramme de classe fourni pour y intégrer les patrons Médiateur et Singleton.

Réponse:

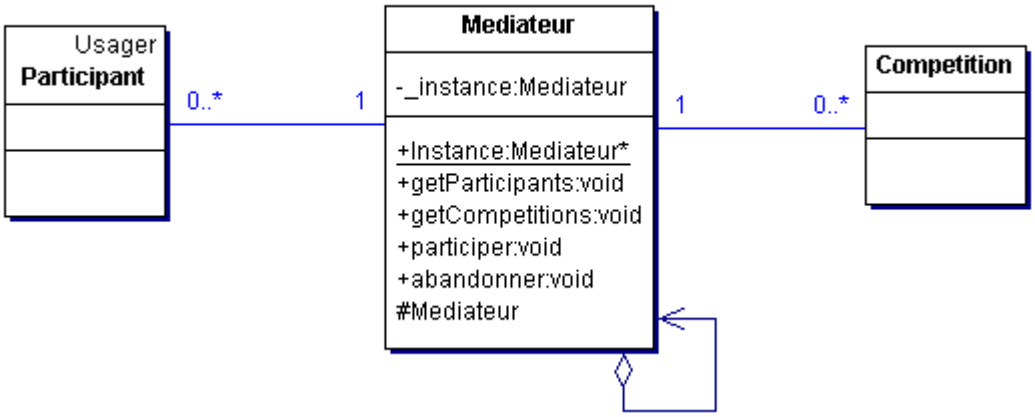

Figure 5 : Ajout du médiateur

d) (3 pts) Donnez deux avantages pour chacun de ces patrons.

Réponse:

Médiateur : gestion des communications simplifiée, recherche simplifiée

Singleton : point d'accès global, contrôle du nombre de médiateur

Ensemble de formes géométriques

Considérez le diagramme de classe de la figure 6 représentant une hiérarchie de formes géométrique. Le patron Visiteur pourrait être utilisé pour regrouper dans une seule classe les fonctionnalités d'affichage des formes géométriques.

a) (1 pt) Donnez, en vos propres mots, l'intention du patron Visiteur.

Réponse:

Représenter une opération qui doit être appliquée sur les éléments d'une structure d'objets. Un Visitor permet de définir une nouvelle opération sans modification aux classes des objets sur lesquels l'opération va agir.

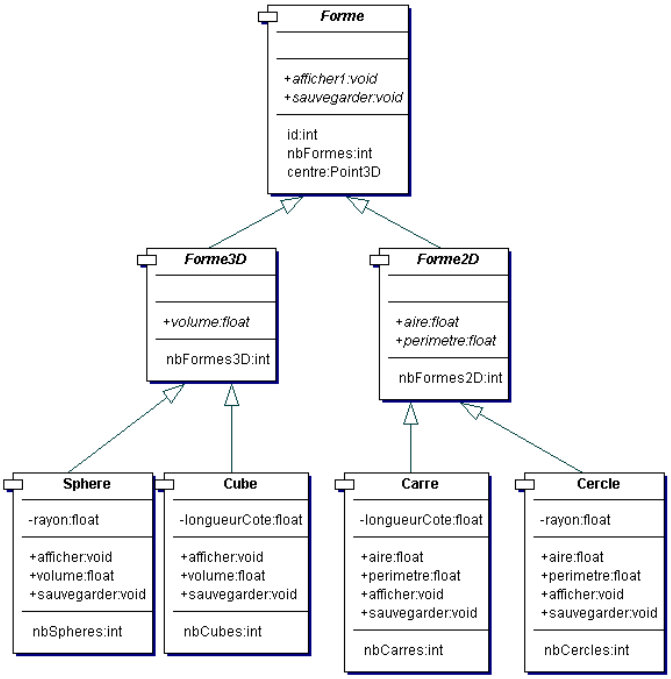

figure 6 : Ensemble de formes géométriques

b) (3 pts) Modifiez le diagramme de classe de la figure 6 pour y intégrer le patron Visiteur.

Réponse:

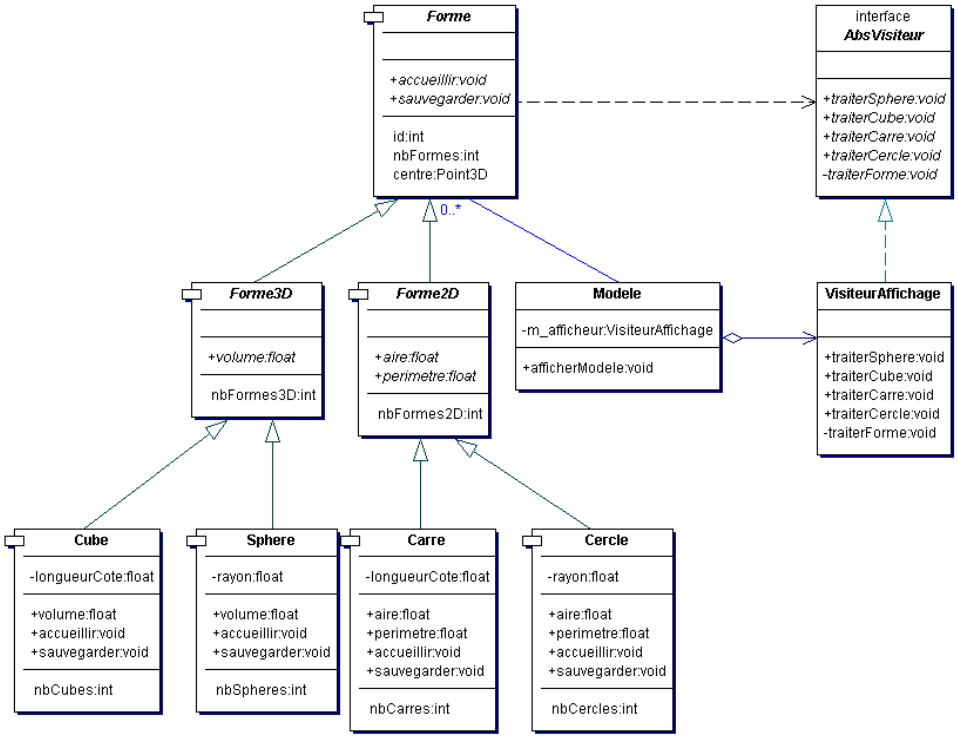

Figure 7 : Intégration du patron visiteur

c) (2 pts) Expliquez le fonctionnement du mécanisme de double invocation (« double dispatch »), qui est utilisé pour l'appel de la bonne méthode d'affichage.

d) Réponse:

Pour permettre l'affichage des formes, il faut créer le bon type de visiteur et s'assurer que la méthode *accueillir()* des formes appelle la bonne méthode de traitement. Donc, l'appel à la méthode d'affichage passe par l'appel de deux méthodes celle de la forme et celle du visiteur.

L'ensemble des formes géométriques possède également une fonction de sauvegarde permettant de conserver les attributs des formes dans un fichier. Plutôt que d'utiliser à nouveau le patron Visiteur, vous choisissez d'utiliser le patron Template Method.

e) (1 pt) Donnez, en vos propres mots, l'intention du patron Template Method.

Réponse:

Définir le squelette d'un algorithme dans une opération, et laisser lessous-classes définir certaines étapes.

f) (3 pts) Expliquez les modifications à apporter au diagramme de classes pour l'implantation de ce patron.

Réponse:

Pour intégrer le patron Template Method, il faut définir les parties fixes et variables d'un algorithme. La méthode contenant l'algorithme doit être publique et non-virtuelle. Les parties variables de l'algorithme sont placées dans des méthodes virtuelles protégées. Pour plus de clarté, le nom de ces méthodes peut commencer par le préfixe « faire » ou « do ».

Question 3 – Héritage et polymorphisme (9 pts)

Qu'est-ce que le concepteur d'une sous-classe doit déduire du fait qu'une méthode de la classe de base est ...

a) (1 pt) ... non virtuelle ?

Réponse:

Il ne faut pas surcharger cette méthode.

b) (1 pt) ... virtuelle ?

Réponse:

Il peut surcharger cette méthode si le comportement par défaut ne lui convient pas.

c) (1 pt) ... virtuelle pure ?

Réponse:

Il doit surcharger cette méthode.

L'héritage multiple peut mener à des situations problématiques. Par exemple, nous avons vu en cours qu'une hiérarchie en forme de losange est potentiellement dangereuse (voir Figure 8).

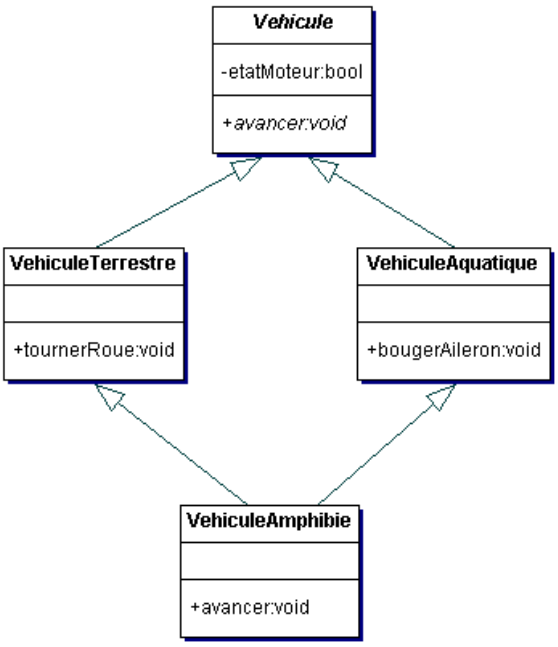

Figure 8 : Structure en losange

d) (1.5 pts) Expliquez le danger principal d'une telle hiérarchie en donnant la structure en mémoire de la classe *VehiculeAmphibie*.

Réponse:
Il existe deux copies en mémoire des attributs de la classe véhicule.

e) (1.5 pts) Donnez une solution pour corriger se problème en spécifiant les limites de cette solution.

Réponse:
Il faut modifier l'héritage pour faire de l'héritage virtuel. Par contre, il faut avoir accès aux classes *VéhiculeTerrestre* et *VehiculeAquatique* pour pouvoir les modifier. Si les classes proviennent d'une librairie, on ne peut peut-être pas faire les modifications nécessaires.

L'héritage privé d'une classe par une autre signifie : « la classe dérivée est dérivée en terme de la classe de base ».

f) (1.5 pts) Donnez deux différences entre l'héritage privé et l'héritage public.

Réponse:
Avec l'héritage privé, tous les attributs et méthodes deviennent privés dans la classe dérivée. De plus, il n'est pas possible de faire une conversion de type entre les classes de bases et les classes dérivées.

g) (1.5 pts) L'héritage privé offre plusieurs similitudes avec une relation d'agrégation. Expliquez dans quel cas une relation d'agrégation est suffisante.

Réponse:
On utilise l'héritage privé lorsqu'on doit accéder aux attributs protégés et si l'on veut surcharger une méthode virtuelle.

Question 4 – Gestion des exceptions (2 pts)

Expliquez l'utilité du traitement des exceptions disponible à même le standard C++ en le comparant aux techniques plus conventionnelles telles que les codes de retour de fonction. (Répondez en 5 à 10 lignes)

Question Bonus (1.5 pts)

Soit le code suivant :

```
class Personne
{
      int m_age;
      string m_nom;
      string m_prenom;

   public :
      inline int getAge()const {return m_age;}
      inline void setAge(int age){m_age = age;}
};


void main()
{
      const Personne usager;

      // Traitement des usagers

      // Modification suite à un anniversaire
      usager.setAge(usager.getAge()+1);
}
```

(1 pt) Expliquez la modification qu'il faut apporter à la classe *Personne* pour permettre la modification de l'attribut *m_age*, bien que l'instance *usager* soit constante.

Réponse:
Il faut définir l'attribut *m_age* comme étant mutable.

(0.5 pt) Expliquez comment on peut modifier cet attribut sans modifier la classe de base.

Réponse:
Il faut utiliser la fonction *const_cast* pour « deconstifier » la variable.