EXAMEN MASTER 1

Seconde session

MODULE ACO

Mardi 16 juin 2015
Durée : 2 heures
Tous documents autorisés

Description du sujet

Le sujet comporte deux parties :

1. un questionnaire, comportant des questions à choix multiples, où vous devez indiquer une ou plusieurs bonnes réponses ;
2. un exercice de conception et de développement à objet à réaliser en utilisant la notation UML et le langage Java.

Le barème est donné à titre indicatif et n'est pas définitif. Il est conseillé de commencer par le questionnaire.

Questionnaire (8 points)

Vous indiquerez vos réponses sur votre copie anonymée sous la forme n° de question- n° du choix correct. Exemple : si pour la question *l* la possibilité correcte est la réponse *a* vous indiquerez *1–a* sur votre copie d'examen. Certaines questions comportant plusieurs possibilités correctes, il faut les indiquer *toutes* pour que la réponse à la question soit considérée globalement correcte, par exemple : *2–cd*. Barème indicatif :

– une réponse globale correcte à une question vaut 1 point ;

– une réponse incorrecte à une question vaut –0,25 points ;

– l'absence de réponse vaut 0 points.

Questions

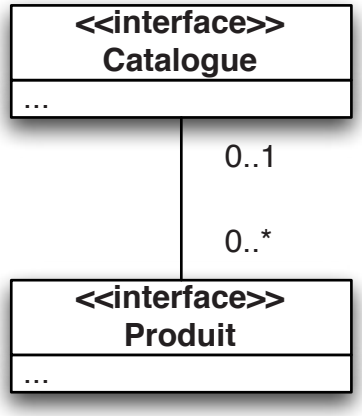
1. Un nom d'interface Java peut être employé à la place d'un nom de classe
   a. *partout*
   b. *seulement après un new*
   c. *sauf après un new*
   d. *sauf après un new et sauf en paramètre d'opération*
2. Dans le patron Command, une classe qui joue le rôle d'Invoker peut choisir l'opération execute() en fonction de la classe concrète de l'objet destinataire du message execute()
   a. *vrai*
   b. *faux*
3. Le patron Visitor regroupe le code de traitement de la structure
   a. *par type de nœud dans la structure*
   b. *par type de traitement*
4. Dans le domaine du test de logiciel, les cas de test sont produits par l'oracle
   a. *vrai*
   b. *faux*
5. Le test structurel repose sur l'emploi exclusif des spécifications du système à tester
   a. *vrai*
   b. *faux*
6. Si lors de l'exécution d'une opération op1 une postcondition de op1 est évaluée à faux alors il s'agit
   a. *d'une faute due uniquement au développeur de la méthode associée à op1 dans tous les cas*
   b. *d'une faute due au développeur qui appelle op1 dans tous les cas*
   c. *d'une faute due au développeur qui appelle op1 uniquement si les préconditions de op1 sont toutes évaluées à faux*
   d. *d'une faute due au développeur de la méthode associée à op1 uniquement si les préconditions de op1 sont toutes évaluées à vrai*
7. Lors de l'application du patron de conception Visitor, le contrôle du parcours dans la structure est dirigé par
   a. *le Visitor*
   b. *le ConcreteVisitor*
   c. *les ConcreteElement*
   d. *le Client et le Visitor*
8. Quelle est la meilleure mise en œuvre Java de l'architecture présentée dans la figure ci-après ?

a.
```
interface Catalogue {
    public ArrayList<Produit> getProduits();
}
```

b.
```
interface Catalogue {
    public Collection<String> getNomProduits();
}
```

c.
```
interface Catalogue {
    public Iterator<Catalogue> getProduits();
}
```

d.
```
interface Catalogue {
    public Iterator<Produit> getProduits();
}
```



Conception (12 points)

On considère dans cette partie les meilleurs moyens de préserver l'encapsulation des objets composites (par exemple les listes). Une liste est une collection ordonnée, et qui permet donc d'accéder aux éléments en utilisant leur rang ou bien en utilisant un itérateur.

Listes immuables

Soit la mise en œuvre suivante :

```
import java.util.List;
public interface Catalogue {
        public List<Produit> getProduits();
}
// -------
import java.util.ArrayList;
import java.util.List;

public class ImplCatalogue1 implements Catalogue {

        private List<Produit> produits;

        public List<Produit> getProduits() {
                return this.produits;
        }

        public ImplCatalogue1() {
                super();
                produits = new ArrayList<Produit>();
        }

}
```

Question 1. *Expliquez en détails les problèmes soulevés par cette mise en œuvre* ImplCatalogue1, par rapport aux bonnes pratiques vues en cours.

On développe maintenant une autre mise en œuvre fondée sur les listes immuables. La bibliothèque standard Java contient une telle mise en œuvre de listes immuables, présentée en cours.

Question 2. *Expliquez en quoi les choix faits pour la mise en œuvre de listes immuables par les concepteurs de la bibliothèque standard Java posent certains problèmes aux utilisateurs.*

On veut mettre en œuvre explicitement un système de liste immuables.

Soit la déclaration suivante :

```
public interface ImmutableList<T> {

        public T getElement(int position);

        public int size();

}

public class ImmutableListImpl<T>  implements ImmutableList<T> {
        // Crée une liste immutable à partir d'une liste mutable
        public ImmutableListImpl<T>(List<T> original);
// À compléter !

}
```

Question 3. *Définissez une architecture décrite en UML, comportant une classe concrète appelée* ImmutableListImpl *qui met en œuvre l'interface* ImmutableList *(voir début de définition ci-dessus) en employant la délégation et employant les types génériques* List *et* ArrayList *de la bibliothèque standard Java.*

Question 4. *Donnez une mise en œuvre en Java de cette architecture.*

Question 5. *Expliquer pourquoi le sous-typage direct de la classe* ArrayList *n'est pas utilisable ici.*

Contrôle des modifications

On veut maintenant réaliser une variante où une liste (mutable cette fois-ci) peut accepter ou refuser des modifications de son contenu.

Lors de l'exécution d'une opération de modification (ajout ou retrait d'éléments dans la liste), une opération de contrôle est appelée. Si cette opération retourne true alors la modification est faite sinon la liste n'est pas modifiée.

On envisage deux mises en œuvre différentes de ce concept :

1. une mise en œuvre statique, où la définition de l'opération de contrôle est permise par le sous-typage de la classe ArrayList ou autre ;
2. une mise en œuvre dynamique où l'opération de contrôle est définie par l'emploi d'un objet séparé contenant l'opération de contrôle.

Question 6. *Indiquez les avantages et inconvénients respectifs des deux techniques mentionnées ci-dessus.*

Dans la suite de l'énoncé on ne s'intéressera qu'à la deuxième technique de mise en œuvre (par contrôle dynamique de la possibilité de mise à jour).

Soit la déclaration suivante :

```
public interface TestMiseAJour {
        public boolean majPossible();
}
```

L'interface TestMiseAJour permet de déclarer des classes concrètes qui fourniront diverses mises en œuvre de l'opération de contrôle lors d'une mise à jour de liste. Le principe mis en œuvre dans la suite repose sur des opérations permettant de changer cette opération de contrôle, qui sera exécutée lors de chaque opération de mise à jour de la liste.

Question 7. *Définissez une architecture UML montrant le fonctionnement du contrôle dynamique fondée sur l'interface* TestMiseAJour. *Une nouvelle interface* ListeContrôlée *est définie partiellement ci-après. Modifiez l'opération* majPossible *si besoin. Expliquez l'architecture au moyen de diagrammes UML statiques, dynamiques et de patrons de conception.*

```
public interface ListeContrôlée<T> {

        public void setTestMajListe(TestMiseAJour t);

        public TestMiseAJour getTestMajListe();

        public T getElement(int position);
        public void setElement(int position, T valeur);
        // Compléter si besoin
}
```

Question 8. *Donnez une mise en œuvre en Java de la classe* ListeContrôléeImpl *qui met en œuvre l'interface* ListeContrôlée.