EXAMEN MASTER 1 INFORMATIQUE

Première session

MODULE ACO

Lundi 13 décembre 2010
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 (10 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 –0,5 points ;

– l'absence de réponse vaut 0 points.

Questions

1. Dans un diagramme de conception détaillée en UML, un attribut du type String
   a. *doit être déclaré public*
   b. *peut être déclaré public et être associé à un getter et un setter*
   c. *doit être déclaré privé et associé à un getter et à un setter*
   d. *doit être déclaré privé , être associé à un getter et éventuellement un setter.*
2. Les patrons de conception ont pour but
   a. *d'augmenter le couplage entre des solutions pour les rendre plus robustes*
   b. *de diminuer le couplage entre des solutions pour les rendre plus robustes*
   c. *de résoudre plusieurs problèmes à l'aide d'une solution unique*
   d. *d'éviter la définition de cas de test*
3. L'interface Iterator de la bibliothèque standard Java sert à parcourir des tableaux
   a. *vrai*
   b. *faux*
   c. *seulement si le tableau est de type primitif (par exemple tableau d'entiers)*
   d. *seulement si le tableau est un tableau d'objets (par exemple tableau de String)*
4. Quel(s) inconvénient(s) présente(nt) le patron de conception Visitor ?
   a. *il faut utiliser les mécanismes d'introspection*
   b. *il faut modifier l'arbre des éléments pour rajouter l'opération visit*
   c. *il faut faire au minimum deux passes dans l'arbre avec le visiteur concret*
   d. *l'état interne des éléments doit être accessible au visiteur concret*
5. Parmi les usages ci-après, indiquer ceux pour lesquels OCL est adapté
   a. *implémentation (mise en œuvre)*
   b. *navigation dans les modèles*
   c. *définition de contraintes*
   d. *définition de requêtes sur un modèle*
6. Parmi les mises en œuvre ci-dessous de la classe Block vue en travaux dirigés, quelles sont la ou les versions correctes du point de vue des bonnes pratiques de conception appliquées en ACO ?
   a. *class Block {*

```
private List<Statement> statements = new
ArrayList<Statement>();
public List<Statement> getStatements() { return this.statements; }
```

}

b. *class Block {*
private List<Statement> statements = new ArrayList<Statement>();
public Iterator<Statement> getStatementsIterator() { return this.statements.iterator(); }
}

c. *class Block {*
public ArrayList<Statement> statements = new ArrayList<Statement>();
}

d. *class Block {*
private List<Statement> = new ArrayList<Statement>();
public getStatements() { return new ArrayList<Statement>(this.statements); }
}

7. Quand la précondition d'une méthode est fausse lors d'une exécution alors
   a. *la méthode peut lever une exception*
   b. *la méthode doit lever une exception*
   c. *la méthode doit continuer son exécution et garantir que la postcondition sera vraie*
   d. *la méthode doit toujours évaluer le prédicat de précondition*
8. Le test fonctionnel est nécessairement du type
   a. *test boîte blanche*
   b. *test boîte noire*
9. Le test d'intégration
   a. *permet de vérifier que l'interface utilisateur est conforme au cahier des charges*
   b. *permet de vérifier que les modules fonctionnent correctement ensemble*
   c. *doit être réalisé avant les tests unitaires des modules*
   d. *doit être réalisé après les tests unitaires des modules*
10. La technique nommée « test aux limites » a pour but de tester le fonctionnement d'un système dans des conditions extrêmes
    a. *vrai*
    b. *faux*

Conception (10 points)

On considère un système de gestion de graphe. Les types élémentaires de ce système sont définis à la figure en fin d'énoncé.

Question 1. Compléter le diagramme donné en fin d'énoncé pour rajouter des classes implémentant les interfaces ci-dessus. Compléter les opérations d'accès. Donner une définition du type marqué ADefinir dans le diagramme ci-dessus. La classe mettant en œuvre Graphe sera nommée GrapheImpl.

Question 2. Donner un diagramme d'objets conforme à votre diagramme de classe et représentant le graphe ci-après.

nœuds = ("Soleil", "Terre","Lune","Mars")

arcs = (("Soleil", 150000, "Terre"), ("Soleil", 228000 , "Mars"), ("Terre",384,"Lune"))

On veut maintenant garantir que la propriété suivante reste vraie en permanence pour tout graphe:

Les extrémités départ et arrivée de tout arc appartiennent à l'ensemble des nœuds du graphe.

Question 3. Écrire la propriété ci-dessus sous forme d'un invariant, exprimé en langue naturelle.

Question 4. Écrire la propriété ci-dessus sous forme d'un invariant exprimé en OCL.

Gestion de l'invariant

Afin de garantir que l'invariant ci-dessus est maintenu, il faut mettre en œuvre un mécanisme de suppression automatique d'arcs qui retire les arcs connectés à un nœud lorsqu'on retire celui-ci.

En utilisant le patron de conception Observer, on veut créer une interface GrapheObservable, qui étend Graphe et qui soit capable de jouer le rôle de Subject, en notifiant tous ses abonnés lors de la suppression d'un nœud (par l'opération Graphe::supprimerNœud). Les abonnés devront obéir à l'interface ObserverGraphe (à définir) qui jouera la rôle d'Observer.

Question 5. Compléter l'architecture statique de la première question, en appliquant le patron de conception Observer et en ajoutant le ou les types nécessaires.

La mise en œuvre de l'interface GrapheObservable peut se faire par extension de la classe d'implantation de Graphe déjà existante (GrapheImpl), ou par délégation.

Question 6. Exposer les avantages et les inconvénients de ces deux possibilités.

La gestion de l'invariant sera réalisée par une classe GestionInvariant qui met en œuvre l'interface ObserverGraphe et qui étend par délégation la classe GrapheImpl de la première question.

Au moyen de diagrammes de séquence, expliquer comment la classe GestionInvariant maintient l'invariant lors de la suppression d'un nœud du graphe.

Question 7. Donner une mise en œuvre Java de GestionInvariant et d'une implantation de GrapheObservable.