Examen de Java

Session : ING1 - Juin 2008 - session 1
Durée : 2h30

Note : la matière Eclipse sera évaluée sur le rendu de projet 2° semestre.

Conditions : Examen sur machines EISTI. Vous disposez d'un accès à la Javadoc 1.5 en interne, à l'URL : http://www.eisti.fr/javadoc/

En fin d'examen, vous devez rendre :

- L'ensemble des classes sources, dans une arborescence "src", inscrite dans un (des) paquetage(s)
- Les classes compilées (même si les écritures ne sont pas complètes, l'état de votre source DOIT pouvoir compiler), dans un répertoire "bin".
- La génération de la javadoc.
- Le tout dans une archive ZIP, ou TGZ à la rigueur (aucun autre format)
- Vous pouvez aussi poster une archive JAR

Commande de livraison : `renduprog2` sur votre compte deathstar.

Thème : les réseaux sociaux

L'actualité du Web est très prolixe en ce moment sur les thèmes des réseaux de personne. Nous utilisons ce prétexte conjoncturel pour vous faire réfléchir sur les nécessités "programmatiques" d'applications qui les utilisent.

Le but du devoir est de modéliser la façon dont une application de type Facebook peut organiser et gérer les mutiples petites applications qu'elle propose aux utilisateurs.

La programmation doit être claire.

La programmation doit faire usage des collections, et ne pas réimplémenter des algorithmes triviaux.

Vous devez mettre en oeuvre toutes vos connaissances Java pour proposer des solutions aux questions posées.

COMMENTEZ ETJUSTIFIEZ VOS INTENTIONS.

Questions de cours POO (4 points)

Pourquoi une classe finale ET abstraite n'a aucun sens ?

En quoi l'utilisation exclusive de membres publics ne tire aucun avantage de l'encapsulation ?

Peut-on réaliser la même chose en programmation procédurale et en programmation objet ?

Réexprimez les trois concepts de classe, d'objet et d'instance.

Conception de classes (6 points)

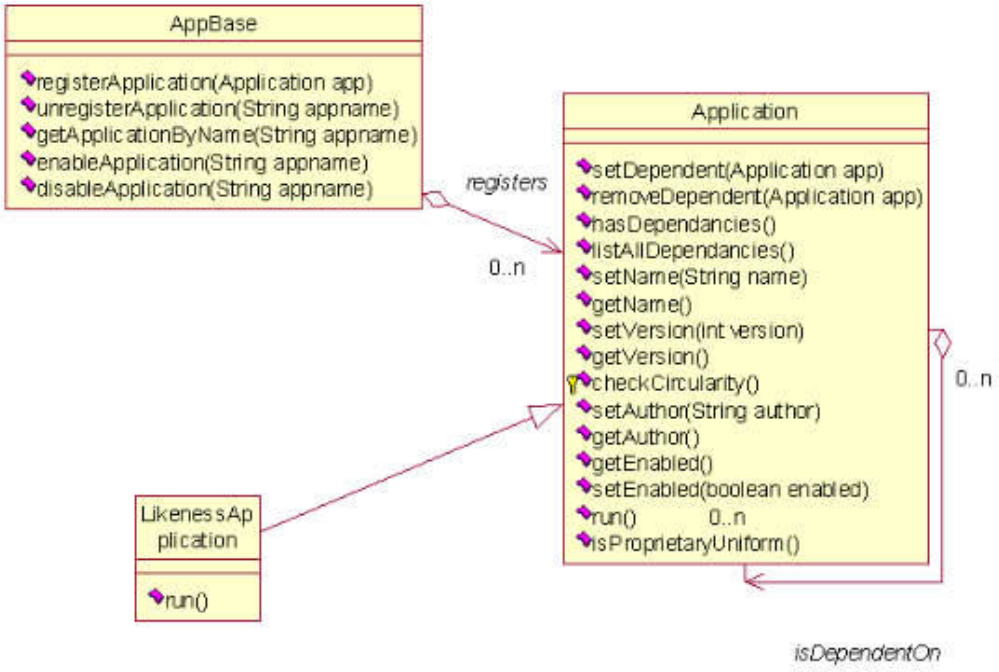
Une application de type Facebook est une application qui utilise d'autres applications créées par les développeurs/utilisateurs/contributeurs. Il s'agit de déteminer un ensemble de classes qui permet la "gestion" de ces sous-applications.

1. Déterminez la classe Application représentant une application Facebook proposée par un utilisateur connu. Fournissez entre trois et cinq champs descriptifs. La classe Application doit être totalement mutable et comporte un numéro de version. **La classe Application est abstraite** car on ne connait pas son fonctionnement (de l'application, matérialisée par la méthode "run()"). Les applications peuvent être activées ou désactivées, installées ou simplement identifiées.
2. Déterminez une façon d'enregistrer les dépendances entre les différentes applications. Identifiez (la ou) les collections nécessaires pour que :

- On puisse déclarer qu'une application "dépend" d'autres applications.
- Il ne peut y avoir de boucles de dépendance : si A dépend de B et B dépend de A, directement ou indirectement, la maintenance de B et de A est indéterminable.
- Il peut y avoir plusieurs fois la même application dans la chaîne de dépendance, mais toujours en versions décroissantes :
  B(2) => A(1) => B(1) est légitime, A(2) => A(1) aussi, C(1) => B(3) => C(4) est interdit.

3. Constituez les classes sous forme de carcasses SANS ECRIRE LE CODE DES METHODES. Votre objectif est de bien décider des membres et des méthodes utiles pour manipuler ce modèle et de savoir le proposer en JAVA.
4. La base d'applications est connue en tant que tel (classe explicite AppBase). Elle doit apparaître comme une identité explicite et implémente donc un comportement de container (Collection à organiser).
5. La base d'applications doit être manipulable, c'est-à-dire que toutes les méthodes qui permettent de la construire à partir d'une base vide doivent être présentes => réviser et compléter les signatures si nécessaire.
6. On désire pouvoir savoir si une application est uniformément propriétaire : c'est le cas si toutes les dépendances (à tous les niveaux de distance) appartiennent à la même personne. Trouvez les méthodes à définir pour que ce calcul soit possible, et expliquez la chaîne d'utilisation (le diagramme de séquence). Complétez les signatures de ces méthodes dans la/les classe(s).

Pour vous aider dans votre implémentation, on vous fournit le shéma UML suivant (pas nécessairement complet) :



On remarque que le problème énoncé dans le sujet ne demande pas nécessairement une grande quantité de classes.

Barème :

- Explications et analyse du problème
- carcasse de la classe abstraite Application
- carcasse de la classe LikenessApplication réalisant l'abstraction "par défaut".
- carcasse de la classe AppBase

L'utilisation de génériques (collections typées) sera évidemment appréciée.

Programmation de méthodes (5 points)

1. Ecrivez le code de la méthode citée en 7 et qui détermine si une application Facebook est uniformément propriétaire.
2. Ecrivez le code d'une méthode qui fournit la liste de toutes les dépendances d'une application donnée.
3. Ecrivez les méthodes toString() permettant d'exprimer une représentation lisible des

instances.

Barème :

- 2 point pour la première,
- 1 point pour la deuxième (réutiliser le principe de la première),
- 1 point pour les impressions.

Programmation d'application (5 points)

1. Ecrivez les constructeurs des classes ci-dessus et
2. un programme principal qui construit un jeu de test et imprime la construction.

Barème :

- 1 point par constructeur
- 1 point pour la classe principale portant le main().