Cours de programmation orientée-objet
Examen du 2 juin 2014

Livres fermés. Durée : 3 heures 1/2.

Veuillez répondre à chaque question sur des feuilles séparées sur lesquelles figurent nom, prénom et section. Soyez bref et concis, mais précis.

1. *(10 points)* Une *matrice creuse* est une matrice dont les valeurs sont majoritairement nulles. Bien qu'une matrice creuse puisse être utilisée comme une matrice traditionnelle, il est courant de développer une structure de données lui étant spécifique. En effet, afin d'économiser la mémoire, il est uniquement nécessaire de mémoriser les valeurs non nulles et de considérer toute valeur absente en mémoire comme étant implicitement nulle.

   On souhaite programmer en Java deux classes intitulées `Matrix` et `SparseMatrix` permettant de représenter une matrice quelconque, respectivement en mémorisant toutes ses valeurs, et en ne mémorisant que ses valeurs non nulles.

   On souhaite que ces deux classes implémentent l'interface suivante :

```
public interface MatrixInterface
{
  double getValue(int i, int j);
  int getHeight();
  int getWidth();
}
```

   où la méthode `getValue()` retourne la valeur située à la ligne `i` $\in [0, M-1]$ et la colonne `j` $\in [0, N-1]$ de la matrice, de dimensions $M \times N$, et où les méthodes `getHeight()` et `getWidth()` retournent respectivement la hauteur M et la largeur N de la matrice.

   Les deux classes `Matrix` et `SparseMatrix` doivent respecter les propriétés suivantes :

   — Toute matrice doit pouvoir être clonée en profondeur.

   — Toute paire de matrices doit pouvoir être comparée à l'aide du mécanisme d'équivalence.

   — L'instanciation d'une matrice doit s'effectuer à l'aide d'un tableau bidimensionnel contenant le contenu complet de la matrice (et rien d'autre).

   Vous êtes libres de développer des classes supplémentaires nécessaires à votre solution. L'utilisation de groupes de classes (packages) **n'est pas demandée**. En revanche, veillez à utiliser des exceptions implémentées par vos soins dans les situations d'erreur.

   Note : Dans le cadre de cet exercice, les performances de l'opération de recherche d'un élément dans la matrice importent peu. Seule l'économie d'espace mémoire est importante.

2. *(6 points)* Répondez aux questions suivantes **en justifiant**. En Java :
   (a) Comment définit-on une variable de classe, une variable d'instance et une variable locale ? Dans chaque cas, où la valeur de cette variable est-elle mémorisée ?
   (b) Qu'est-ce que le chaînage des constructeurs ? Dans quel contexte s'applique t-il ? Dans quel ordre s'effectue t-il ? Par défaut, quels constructeurs sont appelés lors de l'instanciation d'une classe donnée ?
   (c) Quel est l'intérêt des classes abstraites ? Que permet d'effectuer la définition d'une telle classe ?
   (d) Comment peut-on borner le paramètre de type d'une classe générique ? Quel est l'intérêt de ce mécanisme ?

3. *(4 points)* Soient les quatre extraits de code ci-dessous

```
package a;
class A
{
  public static int m;
  int n;
}
```

```
package a;
public class B extends A
{
  protected static int o;
}
```

```
package a.b;
import a.*;
public class C extends B
{
  public static int p;
}
```

```
package a.b;
import a.*;
public class D
{
}
```

Indiquez pour chacune des six affirmations suivantes si elle est vraie ou fausse **en justifiant** votre réponse (une réponse sans justification ou avec une justification fausse ne sera pas considérée).

   (a) La classe `C` a accès à la variable `n`.
   (b) La classe `D` a accès à la variable `A.m`.
   (c) La classe `D` a accès à la variable `B.m`.
   (d) La classe `A` a accès à la variable `B.o`.
   (e) La classe `D` a accès à la variable `C.o`.
   (f) La classe `A` a accès à la variable `C.p`.

Documentation Java

Il est possible que vous souhaitiez utiliser pour la résolution de l'exercice 1 des classes fournies par la bibliothèque standard Java. Ici se trouve la documentation de quelques méthodes d'une classe pouvant éventuellement se révéler utile.

Note : vous n'êtes pas obligés de vous servir de la classe/des méthodes décrite(s) ci-dessous. Utilisez uniquement ce qu'il vous faut en fonction de vos besoins !

Classe `Vector<E>` : file de données (politique FIFO)

- `void add(E e)` : ajoute la référence `e` en fin de file.
- `E get(int i)` : retourne la référence stockée à l'indice `i`.
- `E firstElement()` : retourne la référence stockée en début de file.
- `E lastElement()` : retourne la référence stockée en fin de file.
- `E set(int i, E e)` : remplace la référence stockée à l'indice `i` par la référence `e` et retourne l'ancienne référence.
- `E remove(int i)` : supprime la référence stockée à l'indice `i`, décale tous les éléments successifs à cette référence dans le vecteur vers la gauche et retourne cette référence.
- `boolean isEmpty()` : indique si la file est vide.
- `int size()` : retourne la taille de la file.

Manipulation de tableaux

- La variable `length` d'un objet tableau contient la taille de celui-ci.