Cours de programmation orientée-objet
Examen du 27 août 2013

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. Un chercheur souhaite calculer certaines valeurs statistiques (comme par exemple, la moyenne arithmétique) pour un ensemble d'échantillons de nombres entiers. Un ensemble se présente sous la forme d'un tableau d'entiers dont une référence doit être fournie au constructeur d'une classe intitulée `SampleSet`, implémentée par vos soins et permettant d'encapsuler un tel ensemble.

   Au minimum, cette classe doit permettre par l'intermédiaire de méthodes :

   – D'accéder au i$^{\text{ème}}$ élément (en partant de 0) de l'ensemble encapsulé.
   – De comparer deux ensembles en mettant en œuvre le mécanisme d'équivalence de Java.
   – De calculer une valeur statistique au choix de l'utilisateur sur ce même ensemble.

   Une contrainte du problème est que le calcul d'une valeur statistique doit être modulaire, dans le sens où l'on souhaite que l'ajout d'une nouvelle fonction statistique soit le plus simple possible.

   Afin de respecter ces prérequis, votre classe `SampleSet` doit répondre aux exigences de l'interface suivante (qui exclut l'équivalence pour laquelle vous ajouterez vous-même la/les méthode(s) adéquate(s) dans la classe `SampleSet`) :

```
public interface SampleSetInterface
{
  int getElement(int i);
  double computeStatistic(Object statistic);
}
```

   La méthode `getElement` permet de récupérer le i$^{\text{ème}}$ élément de l'ensemble (en partant de 0) et la méthode `computeStatistic` permet de calculer une valeur statistique, ce calcul étant délégué à l'objet statistique fourni en paramètre. Ici, il est donc hors de question de modifier la classe `SampleSet` ou d'en définir une sous-classe pour ajouter une nouvelle valeur statistique calculable (cette affirmation n'empêche pas l'utilisation du mécanisme d'héritage avec des classes extérieures).

   Afin de montrer que votre programme est suffisamment modulaire, il vous est demandé d'implémenter le calcul des valeurs statistiques suivantes : la valeur maximum, la valeur minimum, la moyenne arithmétique et la moyenne harmonique (étant définie comme l'inverse de la moyenne arithmétique de l'inverse des termes). Il vous est également demandé de fournir une méthode `main` illustrant le calcul de ces quatre valeurs sur un ensemble d'échantillons composé des entiers 4, 8, 15, 16, 23 et 42.

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

2. Répondez aux questions suivantes, **en justifiant**. En Java :
   (a) Comment un constructeur peut en invoquer un autre de la même classe ? Quelle est la condition à respecter pour que cela soit possible ?
   (b) Citez le principe de substitution relatif à l'héritage.
   (c) Quelle(s) différence(s) existe(nt) entre un bloc `catch` et un bloc `finally` ?
   (d) Citez deux manières d'invoquer à l'intérieur d'une méthode d'une classe quelconque les méthodes statiques d'une autre classe située dans un groupe de classes (package) différent.
   (e) Dans le contexte du parallélisme, quel est l'intérêt de déclarer une variable `volatile` ?

3. En considérant les deux extraits de code :

```
public class ClassA
{
  public int v = 1;
  public int m() { return 1; }
  public static int ms() { return 1; }
}
```

```
public class ClassB extends ClassA
{
  public int v = 2;
  public int m() { return 2; }
  public static int ms() { return 2; }
}
```

et les déclarations :

```
ClassA a = new ClassA();
ClassA b = new ClassB();
```

donnez **en justifiant** la valeur renvoyée par les instructions suivantes :

   (a) `a.v;`
   (b) `b.v;`
   (c) `a.m();`
   (d) `b.m();`
   (e) `b.ms();`
   (f) `b instanceof ClassA;`
   (g) `a instanceof ClassB;`