Cours de programmation orientée-objet
Examen du 4 juin 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 brin d'acide désoxyribonucléique (plus connu sous l'appellation d'ADN) peut être caractérisé par une séquence finie de nucléotides. On dénombre 4 nucléotides différents communéments représentés par les lettres A, T, G et C. Par exemple, la séquence AATTGCCGT représente un brin d'ADN.

   Sachant qu'un brin d'ADN peut avoir un nombre très élevé d'occurrences de nucléotides, afin de le représenter efficacement, une compression Run-Length Encoding (RLE) peut être utilisée. La compression RLE consiste à remplacer une séquence d'occurences successives d'un même élément en une paire $< l, e >$ où l est le nombre d'occurrences et e est l'élément lui-même. Par exemple, la séquence AAAAAATTT peut être compressée par la séquence des deux paires $< 6, A >$ et $< 3, T >$.

   On souhaite programmer en Java deux classes intitulées `FullDNA` et `CompressedDNA` permettant de représenter un brin d'ADN (autrement dit, une séquence de nucléotides) respectivement sous sa forme complète, et sous une forme compressée à l'aide de la méthode RLE.

   Les deux classes doivent implémenter l'interface suivante :

```
public interface DNA
{
  Object getNucleotide(int i);
  int getSize();
}
```

   où la méthode `getNucleotide` renvoie un objet encapsulant le $i^{\text{ème}}$ nucléotide d'un brin (en partant de 0) et où la méthode `getSize` renvoie la longueur de la séquence caractérisant un brin (c'est à dire, le nombre de nucléotides appartenant à ce brin).

   Ajoutons que les deux classes demandées doivent respecter les propriétés suivantes :

   – La séquence caractérisant un brin d'ADN doit pouvoir être affichée sous sa forme complète sur la console, pour chacune des deux classes demandées.
   – Deux brins d'ADN doivent pouvoir être comparés à l'aide du mécanisme d'équivalence, même s'ils sont représentés différemment.
   – Les deux classes demandées doivent être instanciées à l'aide d'une séquence de nucléotides fournie sous la forme d'une chaîne de caractères. On considère que cette chaîne contient exclusivement des lettres majuscules.

   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'erreurs.

2. Répondez aux questions suivantes en justifiant. En Java :
   (a) Qu'est-ce que le principe d'encapsulation ? De quels mécanismes dispose-t-on pour l'implémenter ?
   (b) Quelle différence existe t-il entre lien statique et lien dynamique ? À quoi s'appliquent-ils ? Illustrez votre réponse à l'aide d'un exemple concret.
   (c) Citez deux manières différentes d'acquérir un verrou sur un objet.
   (d) Quelles sont les conditions préalables à l'utilisation des méthodes de synchronisation `wait`, `notify` et `notifyAll` ? Décrivez *brièvement* le rôle de ces trois méthodes.
   (e) De quelle manière est-il possible de déclarer qu'une variable ne doit pas être considérée lors de la sérialisation d'un objet ?

3. En considérant l'extrait de code ci-dessous, répondez aux questions suivantes en justifiant :

```
public class Person
{
  private String name;
  private Address[] addresses;

  ...

  public Object clone()
  {
    try
    {
      return super.clone();
    }
    catch (CloneNotSupportedException e)
    {
      throw new InternalError();
    }
  }
}
```

   (a) Le clonage que tente d'opérer la méthode `clone` est-il *superficiel* ou *en profondeur* ?
   (b) En l'état, quel est le résultat d'un appel à la méthode `clone` d'une instance de `Person` ?
   (c) Que faudrait-il changer dans la classe `Person` afin que la méthode `clone` fonctionne ?
   (d) Donnez une version modifiée de la méthode `clone` afin que le clonage alternatif à celui décrit dans votre réponse au point (a) soit effectué. Vous pouvez considérer que la classe `Address` dispose déjà d'un mécanisme de clonage adéquat afin de répondre à la question.

Documentation Java

Il est possible que vous souhaitiez utiliser au cours de la résolution de l'exercice 1 des classes fournies par la bibliothèque standard Java. Ici se trouve la documentation de quelques méthodes de certaines classes pouvant éventuellement de révéler utiles.

Note : vous n'êtes pas obligés de vous servir de chacune des méthodes décrites ci-dessous. Utilisez-les uniquement en fonction de vos besoins !

Classe `Character`

- `static String toString(char c)` : renvoie le caractère `c` sous forme de chaîne de caractères.
- `static char toLowerCase(char c)` : renvoie la version minuscule du caractère `c`.
- `static char toUpperCase(char c)` : renvoie la version majuscule du caractère `c`.

Classe `Vector`

- `void addElement(Object o)` : ajoute la référence `o` vers un objet en fin de file.
- `Object elementAt(int i)` : renvoie la référence vers un objet stockée à l'indice `i`.
- `boolean isEmpty()` : indique si la file est vide.
- `int size()` : renvoie la taille de la file.

Classe `String`

- `char charAt(int i)` : renvoie le caractère à l'indice `i` de la chaîne de caractères.
- `boolean isEmpty()` : indique si la chaîne de caractères est vide.
- `int length()` : renvoie la taille de la chaîne de caractères.

Manipulation de tableaux

- La variable publique `length` contient la taille du tableau.