Algorithmique et programmation
Examen partiel

Nous nous proposons de définir une classe *TableauATrous*, qui possède des méthodes permettant d'ajouter des éléments, supprimer des éléments, rechercher un élément ou accéder au $i^{ème}$ élément. La représentation du tableau se fait de façon non contiguë : c'est à dire que lorsqu'on supprime un élément, on crée un « trou » sans déplacer tous les éléments qui suivent l'élément supprimé. De la même façon lorsqu'on ajoute un élément, on l'ajoute dans le premier « trou » disponible.

Pour représenter un « trou » nous définissons la classe *Trou* : l'attribut *trouSuivant* est une référence au trou suivant dans le tableau. Pour le dernier trou du tableau, *trouSuivant* vaut *null*. *indice* est l'indice du trou dans le tableau.

```
class Trou{
      private Trou trouSuivant;
      private int indice;
      public  Trou (Trou trouSuivant, int indice)
      public int getIndice()
      public void setIndice(int i)
      public Trou getSuivant()
      public void setSuivant(Trou t)
      public String toString()
};
```

Question 1.(3 points)

Définir le constructeur et les méthodes de *Trou*. La méthode *toString* retourne une chaîne de caractère formée de *T* et entre parenthèses l'indice du trou.

Dans la classe *TableauATrous*, l'attribut *premierTrou* est une référence au premier trou du tableau. S'il n'y a pas de trou dans le tableau cette référence vaut *null*. Un tableau qui n'est pas plein contient au moins un trou. Tout élément du tableau qui est un trou contient une référence à l'instance de la classe *Trou* qui le représente.

```
public class TableauATrous{
   private Object[] rep ;
   private int nbElements;
   private Trou premierTrou;
   public TableauATrous ()
   public TableauATrous ( int capaciteInitiale )
   public void ajout ( Object o )
   private int indiceReel( int i)
   public Object elementAt (int i)
   public int indexOf (Object o);
   public void supprimer ( Object o )
   public void supprimer ( int  i)
   public String toString()
   public void tasser()
};
```

Question 2. (3 points)

Définir les constructeurs de la classe *TableauATrous*. Le constructeur par défaut construit un *tableauATrous* de capacité initiale 10.

Question 3. (4 points)

Définir la méthode *void ajout(Object o)* qui ajoute un élément dans le premier « trou » disponible dans la représentation du tableau. Si le tableau est plein on augmente sa taille de la moitié de sa taille plus 1.

Question 4. (4 points)

Définir la méthode *int indiceReel(int i)* Qui retourne l'indice réel dans le tableau du i^{ème} élément du tableau. Cet indice est égal à *i* plus le nombre de trous qui précèdent l'élément de rang *i*.

Définir la méthode *Object elementAt(int i)* Qui retourne l'élément à l'indice *i.* Les « trous » ne font pas partie des éléments du tableau. On lèvera une exception de la classe *IndexOutOfBounds* si l'indice *i* n'est pas correct.

Définir la méthode *int indexOf(Object o)* Qui retourne l'indice de la première occurrence de *o* dans le tableau, et *-1* si *o* n'est pas dans le tableau.

Question 5. (3 points)

Ecrire la définition des méthodes *supprimer(Object o)* et *supprimer(int i)* qui suppriment respectivement la première occurrence de *o* dans le tableau et l'élément d'indice *i*.

Question 6. (3 points)

Ecrire la définition de la méthode *tasser()* qui déplace les éléments du tableau de façon à avoir tous les éléments non trou en début de tableau, et tous les éléments trou en fin de tableau. Lorsqu'un trou est déplacé, il faut changer la valeur de son attribut *indice*.