1 Interfaces

Question 1 *Extraction d'une interface. On suppose connue une classe* `Personne` *et ci-dessous on vous donne le code d'une classe représentant des files d'attente de personnes. Proposez une interface décrivant le type de cette classe.*

```
public class FileAttente
{
     private String nomFile;
     private static int nbPersonnesEntreesTotal = 0;
     public static final String nomAnglais = "queue";
     private ArrayList<Personne> contenu;
     public FileAttente(){contenu=new ArrayList<Personne>();}
     public void entre(Personne p){contenu.add(p); nbPersonnesEntreesTotal++;}
     public Personne sort()
          {
               Personne p=null;
               if (!contenu.isEmpty())
                    {p=contenu.get(contenu.size()-1);
                    contenu.remove(contenu.size()-1);}
               return p;
          }
     public int nbElements(){return contenu.size();}
     public boolean estVide(){return contenu.isEmpty();}
     public String toString(){return ""+descriptionContenu();}
     private String descriptionContenu()
     {
          String resultat = "";
          for (Personne p:this.contenu)
               resultat += p + " ";
          return resultat;
     }
}
```

Question 2 *Comment modifiez-vous la classe* `FileAttente` *pour qu'elle implémente l'interface que vous avez créée ?*

Question 3 *Proposez un type (une interface) file d'attente avec des statistiques. Une opération supplémentaire permet de connaître le nombre total d'opérations réalisées (en cumulant les entrées et les sorties). Puis proposez une classe implémentant cette interface.*

Question 4 *Proposez une autre implémentation* `Rectangle_tab` *de l'interface* `Irectangle` *dans laquelle on stocke la hauteur et la largeur dans un tableau de* `double` *de taille 2. Voyez-vous des parties communes dans cette implémentation qui pourraient suggérer une super-classe des deux implémentations ? Si oui, écrivez-la.*

Analyser le programme suivant avec les éléments que vous avez créés et la classe `RectangleAT` *du cours.*

```
public static void main(String[] arg)
{
    Irectangle r1, r2, r3;
    r1 = new Irectangle();
```

```
        r1 = new Rectangle();
        r2 = new RectangleAT(2,4);
        r3 = new Rectangle_tab(3,5);
        System.out.println(r1+"\n"+r2+ "\n"+ r3);
        System.out.println(r1.perimetre()+"\n"+r2.perimetre()+ "\n"+ r3.perimetre());
        System.out.println(r1.getLargeur()+"\n"+r2.getLargeur()+ "\n"+ r3.getLargeur());
    }
```

QUESTION 5 *Ecrivez une classe pour représenter des stocks de* `Irectangle` *et une méthode retournant la somme de leurs périmètres. Qu'en déduisez-vous sur l'un des intérêts des interfaces ?*

2 Listes

Le code ci-dessous est un extrait très simplifié de l'interface décrivant les listes.

```
public interface List extends Collection
{
    void add(Object element);
    boolean contains(Object o);
    Object get(int index);
    int size();
}
```

L'implémentation basée sur les tableaux consiste à stocker les éléments dans un tableau de plus en plus grand au fur et à mesure des besoins. On mémorise dans un attribut `nbrElements` le nombre d'éléments insérés. Le tableau est créé d'une certaine taille (`tailleInitiale`) et les cases du tableau sont remplies entre 0 et `nbrElements`-1. On ne retire pas d'élément dans cette première version. On décide que lorsque le tableau est plein, on l'agrandit d'une quantité stockée dans un attribut `incrementTaille`. Testez vos méthodes au fur et à mesure.

QUESTION 6 *Proposez une implémentation de liste basée sur un tableau*

QUESTION 7 *Ecrivez une méthode* `equals` *qui vérifie l'égalité de contenu entre deux listes (attention au type du paramètre de* `equals` *de manière à bien profiter de la liaison dynamique).*

QUESTION 8 *Ecrivez une interface* `ListeAvecStatistiques` *qui dispose de 4 méthodes retournant le nombre d'appels de chacune des méthodes* `add`, `contains`, `get`, `size` *effectués dans un programme. Ecrivez une sous-classe* `ListeTableauStatistiques` *implémente* `ListeAvecStatistiques`*. Proposez une méthode* `equals` *qui vérifie l'égalité de contenu et de statistiques de deux listes avec statistiques (attention au type du paramètre de* `equals` *de manière à bien profiter de la liaison dynamique).*

QUESTION 9 *Vérifiez le comportement des instructions suivantes et notamment quelle est la méthode* `equals` *appelée dans chaque cas.*

```
ListeTableau l1 = new ListeTableauStatistiques();
ListeTableau l2 = new ListeTableauStatistiques();
ListeTableauStatistiques l3 = new ListeTableauStatistiques();
ListeTableauStatistiques l4 = new ListeTableauStatistiques();
l1.equals(l2); l1.equals(l3); l3.equals(l1); l3.equals(l4);
```