Université de Montpellier / Faculté Des Sciences

Questions sur le cours d'introduction à la généricité en Java

Structures de données — FMIN 220

8 février 2015

QUESTION 1 *Nous disposons d'une classe* `Oeuf` *et nous avons également défini une classe* `Casier` *qui représente des structures pour stocker et transporter des oeufs. Une première mise en œuvre est présentée ci-dessous.*

```
public class Oeuf{
      //....
      public double taille(){return 9;}
}

public class Casier6 {
      private Oeuf[] contenu;

      public Casier6(Oeuf[] structure)
      {
            if (structure.length==6)
                  this.contenu = structure;
            else System.out.println("structure mal construite");
      }

      public void place(int indice, Oeuf oeuf)
      {
            if (indice >=0 && indice <6)
                  if (contenu[indice] == null)
                        contenu[indice]=oeuf;
                  else System.out.println("place occupee");
            else System.out.println("indice incorrect");};

      public String toString(){
            String s="";
            for (Oeuf o:this.contenu) s+=o+" ";
            return s;
      }

      public static void main(String[] args) {
            Casier6 boiteOeufs = new Casier6(new Oeuf[6]);
            System.out.println(boiteOeufs);
            boiteOeufs.place(0, new Oeuf());
            boiteOeufs.place(7, new Oeuf());
            boiteOeufs.place(0, new Oeuf());
            boiteOeufs.place(1, new Oeuf());
            boiteOeufs.place(2, new Oeuf());
```

```
            boiteOeufs.place(5, new Oeuf());
            System.out.println(boiteOeufs);
      }
}
```

1. *Que va afficher ce programme ?*
2. *Ecrivez une classe* `Bouteille` *succincte munie d'un attribut de type chaîne de caractères représentant son contenu et une méthode retournant ce contenu. Ecrivez une classe représentant les casiers capable de contenir 6 bouteilles. Ecrivez un programme mettant des bouteilles dans ce casier. Ajoutez au casier de bouteille une méthode* `toString`*.*
3. *Ecrivez une classe représentant les casiers capable de contenir 6 objets quelconques. Ecrivez un programme mettant des oeufs et des bouteilles dans un même casier d'objets.*
   (a) *Complétez le programme de manière à parcourir le casier et à afficher, pour un objet stocké, la taille s'il s'agit d'un oeuf et le contenu s'il s'agit d'une bouteille.*
   (b) *Il n'est en réalité pas très sain de mélanger des oeufs et des bouteilles dans un casier. Comment peut-on l'éviter ?*

QUESTION 2 *Transformer la classe* `Casier` *en classe générique.*

QUESTION 3 *Transformer la classe* `FileAttente` *en classe générique, puis écrivez un programme qui crée et remplit une file d'attente de personnes, puis une file d'attente de rectangles.*

```
public class FileAttente
{
     private String nomFile;
     private static int nbPersonnesEntreesTotal = 0;
     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 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 4

— *Ecrivez pour la classe générique* ***File d'attente*** *une méthode statique permettant de retourner le nombre d'éléments entrés dans toutes les files d'attente (utilisez l'attribut existant* ***nbPersonnesEntreesTotal*** *en le renommant* ***nbElementsEntresTotal****).*

— *Ecrivez pour la classe générique* ***File d'attente*** *une méthode statique prenant en paramètre deux files d'attente contenant des objets de même type et retournant vrai si elles contiennent les mêmes objets.*

— *Ecrivez pour la classe générique* ***File d'attente*** *une méthode non statique prenant en paramètre une file d'attente contenant des objets du même type que la file receveur et retournant vrai si les deux files contiennent les mêmes objets.*

— *Ecrivez un programme qui montre comment appeler ces méthodes.*

Question 5 *Ecrivez pour la classe générique* ***File d'attente*** *une méthode non statique prenant en paramètre une file d'attente avec des éléments qui ne sont pas forcément du même type que la file receveur et retournant vrai si les deux files sont de la même longueur. Ecrivez un programme qui montre comment l'appeler sur une file de rectangles et sur une file de personnes.*

Question 6 *Ecrivez un programme créant une file d'attente de personnes et une file d'attente de rectangles. Mettez une personne dans la première et un rectangle dans la seconde. A votre avis, quelle est la valeur de l'attribut static* ***nbPersonnesEntreesTotal*** *après ces deux opérations ? Comment l'expliquez-vous ?*

Question 7 *Pleins de bonnes intentions, nous voudrions transformer la classe implémentant la liste avec un tableau en classe générique, et nous écrivons le code (partiel) ci-dessous. Malheureusement, cela va mal se passer. Essayez de comprendre quelle est la partie du code en cause.*

```
public class listeTabGenerique<E> {
private int nbrElements=0;
private int tailleInitiale=10;
private E[] contenu;

public listeTabGenerique()
{this.contenu = new E[this.tailleInitiale];}

public listeTabGenerique(int tailleInitiale)
{
      this.tailleInitiale=tailleInitiale;
      this.contenu = new E[this.tailleInitiale];
 }
 //.............
}
```

Question 8 *Les tableaux vont se révéler produire d'autres ennuis. Le code suivant compile et provoque une erreur à l'exécution. Identifiez le problème.*

```
Object[] t = new Object[3];
t[0] = "Alice";
t[1] = new Personne();
t[2] = new Rectangle_tab();

t = new String[3];
t[0] = "Alice";
t[1] = new Personne();
t[2] = new Rectangle_tab();
```

Déduisez-en la raison pour laquelle ce qui suit (la deuxième affectation) vous sera interdit (ne compilera pas) :

```
ArrayList<Object> a = new ArrayList<Object>();
a = new ArrayList<String>();
```

QUESTION 9 *Ecrivez une classe générique* ***PaireEtiquetee*** *à partir de la définition de la classe générique* ***Paire****. Une paire étiquetée est une paire à laquelle on attache une étiquette. Le type de l'étiquette est aussi un paramètre de généricité. Ecrivez un programme créant des paires étiquetées avec des étiquettes de type* ***String****, puis des paires étiquetées avec des étiquettes de type* ***Integer****.*

QUESTION 10 *Ecrivez une classe* ***EntreeAgenda*** *à partir de la définition de la classe générique* ***Paire****. Une entrée d'agenda est une paire dont le premier membre est une date et le deuxième membre est une chaîne de caractères. Ecrivez un programme créant des entrées d'agenda.*

QUESTION 11 *Ecrivez une classe générique* ***BouteilleEtiquetee*** *à partir de la définition de la classe* ***Bouteille****. Une bouteille étiquetée est une bouteille à laquelle on attache une étiquette. Le type de l'étiquette est aussi un paramètre. Ecrivez un programme créant des bouteilles étiquetées avec des étiquettes de type* ***String****, puis des bouteilles étiquetées avec des étiquettes d'une classe* ***EtiquetteBouteille*** *que vous définirez. Une étiquette de bouteille comprend plusieurs informations : le degré d'alcool, le nom du producteur, son adresse, un descriptif du contenu, la quantité contenue (en litres).*

QUESTION 12 *Ecrivez une classe générique* ***PaireHomogène*** *à partir de la définition de la classe générique* ***Paire****. Une paire homogène est une paire dont les deux membres sont de même type. Ecrivez un programme créant des paires homogènes.*

QUESTION 13 *Ecrivez une classe générique* ***PaireAvecStatistiques*** *à partir de la définition de la classe générique* ***Paire****. On mémorisera le nombre de paires créées, le nombre d'accès en lecture au premier membre (resp. au second membre), et le nombre d'accès en écriture au premier membre (resp. au second membre). Ecrivez un programme créant des paires avec statistiques, les manipulant et vérifiant que les statistiques fonctionnent bien. Comment vous y prendriez-vous pour mémoriser séparément les accès en lecture au premier membre selon le type ? Par exemple, le nombre d'accès en lecture au premier membre d'une paire lorsque c'est une* ***String*** *séparément du nombre d'accès en lecture au premier membre d'une paire lorsque c'est un* ***Integer****.*