Exercices dirigés
séance n°7 - corrigé

Exercice 1 : Matrice creuse

Une matrice creuse est une matrice d'entiers essentiellement composée de 0. Les valeurs sont comprises entre 0 et 10 ([0,10[ ).

0	2	0	0	0
1	0	3	0	0
0	0	0	0	0
0	0	0	6	0

`l'élément situé ligne 4,colonne 4 vaut 6`

On souhaite représenter une matrice creuse par une liste chaînée, contenant uniquement les éléments non nuls, avec leurs indices.

[(2,(1,2)); (1,(2,1)); (3,(2,3)); (6,(4,4)); (0, (4,5))]

Les éléments sont rangés par indice croissant (ligne puis colonne).
Le dernier élément de la liste permet de connaître l'indice maximal de la matrice même si celui-ci est nul ((0, (4,5)) dans l'exemple).

Question 1
Définir la classe `Element`, c'est à dire ses variables d'instance, un constructeur qui les initialise et au moins la méthode `println()` pour afficher l'état d'une instance.

Question 2
Définir la classe `ListeCreuse`. en s'inspirant de la classe `Liste` vue en cours et en ED. On y ajoutera une méthode d'affichage de la liste.

Question 3
Définir la classe `MatriceCreuse` comme un tableau à 2 dimensions. On initialisera cette matrice de manière aléatoire en utilisant la méthode `Math.random()` du package `java.lang` qui retourne un `double` compris entre `0` et `1` ([0,1[ ).

Question 4
Définir une fonction `ajouter` qui ajoute un élément en tête à la liste.

Question 5
Écrire la fonction `transformer` qui transforme une matrice creuse en une liste chaînée selon le modèle présenté ci-dessus.

Question 6
Ecrire un petit programme de test qui crée une matrice creuse, l'affiche, la transforme en une liste creuse et affiche cette dernière.

--- Exercice 1 ------ Solutions ----------

Question 1

```
public class Element{
   private int valeur;
   private int ligne, colonne;

   public Element( int valeur, int ligne, int colonne ){
        this.valeur = valeur;
        this.ligne = ligne;
        this.colonne = colonne;
   }

   public int getValeur(){
        return valeur;
    }

   public int getLigne(){
        return ligne;
   }

  public int getColonne(){
        return colonne;
  }

  public void println(){
      System.out.println("["+valeur+" ("+ligne+","+colonne+")]");
  }
}
```

Question 2

```
import static java.lang.System.*;

public class ListeCreuse{
   private Element valeur;
   private ListeCreuse suivant;

   public ListeCreuse(){}

   public ListeCreuse(Element premier, ListeCreuse reste){
        valeur = premier;
        suivant = reste;
    }

   public ListeCreuse ajouter( Element item ){
        ListeCreuse ref = this;
        ref = new ListeCreuse( item,this );
        return ref;
   }

    // récupération de la valeur de la tête de liste
   public Element tete(){
        return this.valeur;
   }
    // récupération de la liste privée de son premier élément
   public ListeCreuse queue(){
        return this.suivant;
   }
```

```
    public void println(){
       ListeCreuse ref = this;
       while( ref!=null ){
           Element item = ref.tete();
           item.println();
           ref = ref.queue();
      }
      out.println();
   }
}
```

Question 3,4,5

```
import static java.lang.System.*;
public class MatriceCreuse{
    private int[][] matrice;

    public MatriceCreuse( int i, int j ){
        matrice = new int[i][j];
        for(int a=0;a<i;a++)
            for(int b=0;b<j;b++){
              double x = Math.random();
              if( x>0.1) matrice[a][b]=0;
              else matrice[a][b]=(int)(java.lang.Math.random()*10);
            }
    }

    public ListeCreuse transformer(){
        int nl = matrice.length-1;
        int nc = matrice[0].length-1;
        Element item = new Element(matrice[nl][nc],nl,nc);
       // le dernier élément est placé que sa valeur soit nulle ou pas
        ListeCreuse liste = new ListeCreuse(item,null);
        for( int i=nl;i>=0;i--)
            for( int j=nc;j>=0;j--){
                item = new Element( matrice[i][j],i,j );
                if( matrice[i][j]!=0 )
                  // le dernier élément est déjà ajouté,
                  // inutile de la placer à nouveau
                    if( i!=nl && j!=nc)
                      // on ajoute devant
                        liste=liste.ajouter(item);
            }
        return liste;
    }

    public void println(){
        for(int a=0;a<matrice.length;a++){
            out.print("[ ");
            for(int b=0;b<matrice[0].length;b++)
               out.print(matrice[a][b] +" ");
            out.println("]");
        }
    }
}
```

Question 6

```
public class TestMatrice{
   public static void main( String[] args ){
       MatriceCreuse matrice = new MatriceCreuse(10,10);
       matrice.println();
       ListeCreuse liste = matrice.transformer();
       liste.println();
    }
}
```

Exemple de résultat d'un test

```
[ 0 0 0 0 0 0 0 0 0 0 ]
[ 0 0 4 0 0 0 0 0 0 0 ]
[ 0 0 0 0 0 0 0 0 0 0 ]
[ 0 0 0 0 0 0 0 3 2 0 ]
[ 0 0 0 0 0 0 0 0 0 0 ]
[ 0 0 0 0 5 0 0 0 0 0 ]
[ 0 0 0 0 0 7 0 0 0 0 ]
[ 0 0 0 0 0 0 0 7 0 0 ]
[ 0 0 0 0 0 0 0 0 0 0 ]
[ 0 0 0 0 0 0 0 0 0 0 ]
[4 (1,2)]
[3 (3,7)]
[2 (3,8)]
[5 (5,4)]
[7 (6,5)]
[7 (7,7)]
[0 (9,9)]
_
```

Exercice 2 : vente par correspondance

Un magasin de vente par correspondance souhaite disposer d'un logiciel de gestion de ses ventes. Chaque article disponible dans le magasin est décrit par sa référence (entier), par sa dénomination (une chaîne de caractères) et par son prix. L'ensemble des articles (le stock) est représenté par un tableau.

Question 1

Définir la classe `Article` et la classe `Stock`.

Question 2

Les commandes des clients sont également représentées par un tableau. Une commande groupe l'ensemble des achats effectués par un client. Elle est caractérisée par un numéro de commande, le nom du client, les achats et le solde. Un achat est caractérisé par un article, le nombre d'exemplaires commandés et le nom du client. Les achats du client sont représentés par une liste chaînée.
Définir la classe `Achat`, c'est à dire variables d'instance, constructeur avec initialisation et accesseurs (getter et setter)
Définir la classe `Commande`.
On utilisera la classe `java.util.Hashtable`, pour réunir au sein d'une structure commune tous les couples (référence, article).

Exemple de `Hashtable` :

```
Hashtable<String,Integer> numbers = new Hashtable<String,Integer>();
numbers.put("one", 1);
numbers.put("two", 2);
numbers.put("three", 3);
// parcours de la table
Integer i = null;
for(Enumeration<Integer> iter=numbers.elements();iter.hasMoreElements();)
      i = iter.nextElement();
```

Question 3

Ecrire une méthode qui permet d'ajouter un achat à une commande.

Question 4

Ecrire une méthode qui modifie, dans une commande, le nombre d'exemplaires d'un article acheté connaissant sa référence.

Question 5

Ecrire une méthode qui retourne le nombre d'articles achetés lors d'une commande.

Question 6

Ecrire l'algorithme puis la méthode qui retourne le solde d'une commande.

--- Exercice 2 ------ Solutions ----------

Question 1

```
public class Article{
    private int reference;
    private String denomination;
    private double prix;

    public Article(){}
    public Article(int reference, String denomination, double prix){
        this.reference = reference;
        this.denomination = denomination;
        this.prix = prix;
    }

    public int getReference(){
        return reference;
    }

    public String getDenomination(){
        return denomination;
    }

    public double getPrix(){
        return prix;
    }

    public void setReference( int reference ){
        this.reference = reference;
    }

    public void setDenomination( String denomination ){
        this.denomination = denomination;
    }

    public void setPrix( double prix ){
        this.prix = prix;
    }

    public void println(){
        System.out.print("[référence : "+reference);
        System.out.print(", dénomination : "+denomination);
        System.out.println(" , prix : "+prix+"]");
    }
}

public class Stock{
    private ArrayList stock;

    public Stock(){stock = new ArrayList()}
    public Stock(ArrayList stock ){
        this.stock = stock;
    }

    public ArrayList getStock(){
        return stock;
    }
}
```

Question 2

```
public class Achat{
    private Article article;
    private int nbExemplaires;
    private String client;

    public Achat(){}
    public Achat( Article article, String client){
        this.article = article;
        this.client = client;
    }

    public int getNbExemplaires(){
        return nbExemplaires;
    }

    public int getReference(){
        return article.getReference();
    }

    public Article getArticle(){
        return article;
    }

    public void incNbExemplaires( int nb ){
        nbExemplaires = nbExemplaires + nb;
    }
}

public class Commande{
    private Hashtable<Integer,Achat> achats;
    private int numero;
    private String client;
    private double solde = 0;

    public Commande(){
        achats = new Hashtable<Integer,Achat>();
    }

    public Commande(int numero,
                    String client,
                    Hashtable<Integer,Achat> achats ){
        numero = numero;
        client = client;
        this.achats = achats;
    }

    public void addAchat( Achat achat ){
        achats.put( new Integer(achat.getReference()),achat );
    }

    public void setNbExemplairesAchat( int nb, int reference ){
        Achat achat = achats.get( new Integer(reference) );
        achat.incNbExemplaires( nb );
    }

    public int nbExemplairesArticlesAchetes(){
        int nbAchats = achats.size();
        int nb = 0;
        Achat achat = null;
```

```
        for( Enumeration<Achat> iter = achats.elements();
                                    iter.hasMoreElements(); ){
            achat = iter.nextElement();
            nb = nb+achat.getNbExemplaires();
        }
        return nb;
    }

    public double getSolde(){
        int nb = 0;
        double prix =0.0;
        Article article;
        double total = 0.0;
        Achat achat = null;
        for( Enumeration<Achat> iter = achats.elements();
                                    iter.hasMoreElements(); ){
            achat = iter.nextElement();
            nb = nb+achat.getNbExemplaires();
            article = achat.getArticle();
            prix = article.getPrix();
            total = total+nb*prix;
        }
        return total;
    }
}
```

Question 3

```
public void addAchat( Achat achat ){
        achats.put( new Integer(achat.getReference()),achat );
}
```

Question 4

```
public void setNbExemplairesAchat( int nb, int reference ){
        Achat achat = achats.get( new Integer(reference) );
        achat.incNbExemplaires( nb );
}
```

Question 5

```
public int nbExemplairesArticlesAchetes(){
        int nbAchats = achats.size();
        int nb = 0;
        Achat achat = null;
        for( Enumeration<Achat> iter = achats.elements();
                                     iter.hasMoreElements(); ){
            achat = iter.nextElement();
            nb = nb+achat.getNbExemplaires();
        }
        return nb;
 }
```

Question 6

```
public double getSolde(){
        int nb = 0;
        double prix =0.0;
        Article article;
        double total = 0.0;
        Achat achat = null;
```

```
        for( Enumeration<Achat> iter = achats.elements();
                                    iter.hasMoreElements(); ){
            achat = iter.nextElement();
            nb = nb+achat.getNbExemplaires();
            article = achat.getArticle();
            prix = article.getPrix();
            total = total+nb*prix;
        }
        return total;
}
```