Université de Tunis El Manar
FSEGT

EXAMEN DE POO JAVA

Session : Principale 2015
Section : 2ème Année LFIG
Durée : 2 heures
Documents : Non Autorisés

Exercice 1 (14pts)

L'objectif de cet exercice est de développer une application de gestion des demandes de congé d'une entreprise.

1. **Classe Fonctionnaire**
   a. Créer une classe Fonctionnaire qui se caractérise par : Numéro de matricule, nom, prénom, date de naissance et solde des jours de congés du fonctionnaire.
   b. Créer un constructeur à 2 paramètres : nom et prénom, ce constructeur initialise le numéro de matricule (affecté de façon incrémentale par rapport au nombre des fonctionnaires), et la date d'embauche par la date système.
   c. Ajouter des accesseurs pour tous les champs.
   d. Ajouter une méthode toString () qui renvoie les informations d'un fonctionnaire.
2. **Classe Directeur**
   a. Créer une classe Directeur sous-classe de la classe Fonctionnaire, elle comporte une liste des fonctionnaires.
   Ecrire le constructeur correspondant ainsi que sa méthode toString qui renvoie les informations d'un Directeur ainsi que les noms des fonctionnaires.
   b. Ajouter une méthode ajouterFonctionnaire qui permet d'ajouter un Fonctionnaire à la liste des Fonctionnaires.
   c. Ajouter une méthode getFonctionnaireParNom qui recherche et renvoie le Numéro de matricule d'un Fonctionnaire par nom et prénom.
   d. Ajouter une méthode getFonctionnairesParAge qui renvoie la liste des fonctionnaires qui atteignent l'âge de 60 ans cette année.
3. **Classe DemandeConge**
   a. Créer une classe DemandeConge caractérisé par : code fonctionnaire, date début, durée, motif, Etat (En cours, Validé, Refusé).
   b. Ajouter un constructeur à 4 paramètres : code fonctionnaire, date début, durée et motif, et initialise l'état par la valeur « En cours ».
   c. Ajouter une méthode CalcDateFin qui calcule la date de fin à partir de la date début et durée.
   d. Ajouter des accesseurs pour tous les champs.
4. **Classe GestionConge**
   a. Créer une classe GestionConge qui se caractérise par une liste des Directeur et une liste des DemandeConge et ajouter un constructeur sans paramètre.
   b. Ajouter une méthode listeFonctionnaires qui affiche la liste des noms des Fonctionnaires qui ont demandé des congés cette année.
   c. Ajouter une méthode ajouterDemandeConge qui permet d'ajouter un objet DemandeConge à la liste des demandes.
   d. Ajouter une méthode listeFoncEnConge qui retourne les noms des fonctionnaires qui sont en congé.

e. Ajouter une méthode listeDemandeCongeParFonctionnaire qui retourne une liste des demandes de congé d'un fonctionnaire, cette méthode a comme paramètre le code d'un Fonctionnaire.

***NB.* Utiliser des tableaux dynamiques de type *ArrayList* pour modéliser les listes de fonctionnaires et des demandes de congé.**

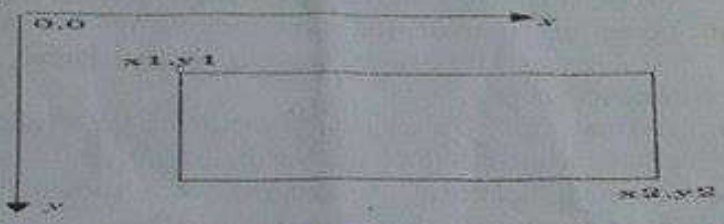
Utiliser la classe Date pour la gestion des dates :

Lire cet exemple pour pouvoir utiliser quelques méthodes utiles de la classe Date :

```
import java.util.Date;
public class TestDate {
  public static void main(String[] args) {
      Date maDate = new Date(); //récupère la date courante
      System.out.println(maDate.toString()); // affiche la date d'aujourd'hui "Thu Jan
      15 ... 2015"
      System.out.println(maDate.getFullYear()); // affiche seulement l'année "2015"
      Date dte = new Date();
      dte.setDate(dte.getDate()+3); //ajoute 3 jours à la date courante
      System.out.println(dte.toString()); // affiche la date "Sun Jan 18 ... 2015"
      if(dte.compareTo(maDate)<0)
        System.out.println("ok"); // le système n'affiche pas "ok" car dte est
        postérieure à maDate
  }
}
```

Exercice 2 (6pts)

On désire réaliser un logiciel de dessin en Java en utilisant la technique objet sans utiliser le package java.awt ou swing de Java. Un dessin est défini par les coordonnées de deux de ses points, (x1, y1, x2, y2) et la méthode **dessineToi()**. La figure suivante montre un dessin avec ses coordonnées.



On suppose que les classes suivantes sont déjà définies (en Java) :

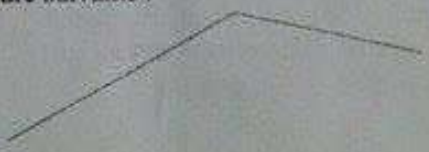
```
import java.util.*;
abstract class Dessin extends Object {
int x1, y1, x2, y2;
abstract void dessineToi(); // doit être redéfinie
Dessin(int _x1, int _y1, int _x2, int _y2) {
x1 = _x1; x2 = _x2;
y1 = _y1; y2 = _y2;
}
}

class DessinComplexe extends Dessin {
Vector dessins = new Vector();
```

```
void ajoute(Dessin d) {
dessins.addElement(d); // ajoute un dessin
}
void dessineToi() {
//à définir
}
}

class DessinElementaire extends Dessin {
}

class Rectangle extends DessinElementaire {
Rectangle(int g, int h, int b, int d) {
Dessin(g, h, b, d);
}
void dessineToi () {
// c'est une primitive
}
}

class Ligne extends DessinElementaire {
Ligne(int xp1, int yp1, int xp2, int yp2) {
Dessin(xp1, yp1, xp2, yp2);
}
void dessineToi () {
// c'est une primitive
}
}
```

Question A) Donnez le code de la méthode **dessineToi()** dans DessinComplex

Question B) On veut dessiner le dessin complexe **dc1** suivant comprenant 2 rectangles et 1 ligne, les rectangles et la ligne étant définis ainsi:
Rectangle r1 = new Rectangle(10,10,20,40);
Rectangle r2 = new Rectangle(5,25,50,60);
Ligne l1 = new Ligne(5,35,45,10);
Donnez le code de l'instanciation permettant de construire **dc1**.

Question C) On désire réaliser la classe DoubleLigne qui comprend 2 lignes reliées entre elles, comme le montre la figure suivante :



Une instance de cette ligne est définie ainsi:
DoubleLigne dl1=DoubleLigne(5,10,30,35,20,60);
Question: Définissez la classe DoubleLigne avec ses champs et ses méthodes (on donnera le code des méthodes).

ANNEXE 1 (la classe *java.util.ArrayList*)

Méthodes utiles de la classe *java.util.ArrayList* pour gérer des tableaux dynamiques :

- ArrayList() // constructeur non paramétré
  Exemple : ArrayList T = new ArrayList(); // Création de T initialement vide
- ArrayList(int N) // constructeur paramétré
  Exemple : ArrayList T2 = new ArrayList(10);
- void add(object o) // Ajout d'un objet
  Exemple : T.add(A); // Ajout de l'objet A dans T
- void add(int i, object o) // Ajout d'un objet à la position i
  Exemple : T.add(0,B);
- int size() //retourne la taille du tableau
  Exemple : s.o.p(" Taille de T :"+ T.size());
- boolean isEmpty() // retourne vrai si le tableau est vide
- void clear() //vide le tableau
  Exemple : T.clear();
- void remove(int i) //suppression de l'objet se trouvant à la position i
- void remove(object o) //suppression de l'objet o
- object get(int i) // retourne l'objet se trouvant à la position i
- void set(int i, object o) // remplace l'objet se trouvant à la position i par o

ANNEXE2 : La classe Vector

La classe Vector du package java.util permet de stocker des objets dans un tableau dont la taille évolue avec les besoins.

➢ **Constructeurs**

- Vector vect=new Vector();
- Vector vect=new Vector(n); //pour prévoir une capacité initiale de n.

➢ **Ajouter un objet**

On ajoute un objet à la fin du Vector en utilisant la méthode addElement. Par exemple, pour ajouter l'objet o, on écrira :

- vect.addElement(o);

Le premier élément ajouté a l'indice 0, le suivant l'indice 1, etc... La méthode **size()** renvoie le nombre d'éléments contenus dans le vecteur qui sera aussi l'indice du prochain objet ajouté.

➢ **Lire un objet**

On retrouve un objet à partir de son indice en utilisant la méthode elementAt. Par exemple on obtient l'objet d'indice n en écrivant :

- o=vect.elementAt(n);