Programmation Orientée Objet (POO)

EX 6 : Interfaces

Objectifs

- Comprendre l'utilité d'une interface
- Programmer des interfaces
- Liens entre interfaces et classes abstraites

1. AbstractAction

Voici un extrait de la javadoc:

```
public abstract class AbstractAction
extends Object
implements Action, Cloneable, Serializable

This class provides default implementations for the JFC Action
interface. Standard behaviors like the get and set methods for Action
object properties (icon, text, and enabled) are defined here. The
developer need only subclass this abstract class and define the
actionPerformed method.
```

Vous n'avez pas à écrire de programme qui utilise cette classe.
Vous devez seulement répondre aux questions ci-dessous :

1. Est-ce que AbstractAction est une classe concrète ? NON
2. Est-ce que AbstractAction est une classe abstraite ? OUI
3. Est-ce que AbstractAction est une interface ? NON

Qu'est-ce que cela signifie pour le développeur qui veut utiliser AbstractAction :

4. Qu'est-ce qu'il peut faire : définir une sous classe de AbstractAction qui sera concrète en proposant une implémentation de méthode abstraite `actionPerformed`

ne peut pas faire : créer des instances de la classe AbstractAction

5. Qu'est-ce qu'il doit faire ? cf ce qu'il peut faire

2. Héritage, classes abstraites, interfaces : Salariés

Le directeur d'une entreprise souhaite gérer les salaires et primes de ses employés au moyen d'un programme Java.

QUESTION 1). Vous devez écrire un programme Java permettant de gérer ce qui est décrit ci-dessous.

Vous déciderez vous même si vous devez définir des classes concrètes, des classes abstraites et/ou des interfaces. Vous devrez aussi choisir les relations d'héritage et d'implémentation appropriées.

Un employé est caractérisé par
son nom, son prénom, son age, son année d'entrée en dans l'entreprise ;
une méthode calculerSalaire (ce calcul dépendra du type de l'employé) ;
une méthode getNom retournant une chaine de caractères obtenue en concaténant la chaine de caractères "L'employé " avec le prénom et le nom ;
si cela est faisable, un constructeur permettant d'initialiser les attributs.

Le calcul du salaire mensuel dépend du type de l'employé. On distingue les types d'employés suivants :
Ceux affectés à la Vente directe. Leur salaire mensuel est 20 % du chiffre d'affaire qu'ils réalisent mensuellement, plus 200 Euros.
Ceux affectés à la Représentation. Leur salaire mensuel est 30 % du chiffre d'affaire qu'ils réalisent mensuellement, plus 500 Euros.
Ceux affectés à la Production. Leur salaire vaut le nombre d'unités produites par mois par cet employé multipliées par 10.
Ceux affectés à la Manutention. Leur salaire vaut leur nombre d'heures de travail mensuel de cet employé multipliées par 15 Euros.

Chaque sous classe est dotée de constructeur prenant en argument l'ensemble des attributs nécessaires.

N'hésitez pas à introduire des classes intermédiaires pour éviter au maximum les redondances d'attributs et de méthodes dans les sous-classes.

Ecrire un main() dans une classe Salaires.

Certains employés des secteurs production et manutention sont appelés à fabriquer et manipuler des produits dangereux et ont une prime mensuelle. Ajouter également à votre programme une spécification pour les employés à risque permettant de leur associer une prime mensuelle fixe (donc une constante définie) de 200 Euros. On veut pouvoir bénéficier du polymorphisme, par exemple en gérant de manière homogène des salariés à risque. Donner un exemple dans le main().

On vous donne la classe suivante :

```
class Personnel {
        private Employe[] staff;
        private int nbreEmploye;
        private final static int MAXEMPLOYE = 200;
        public Personnel() {
```

```
                staff = new Employe[MAXEMPLOYE];
                nbreEmploye = 0;
        }
    public void ajouterEmploye(Employe e) {
        if (nbreEmploye <= MAXEMPLOYE) {
                ++nbreEmploye;
                staff[nbreEmploye - 1] = e;
        } else {
                System.out.println("Pas plus de " + MAXEMPLOYE + " employés");
        }
    }

    public double salaireMoyen() {
        double somme = 0.0;
        for (int i = 0; i < nbreEmploye; i++) {
                somme += staff[i].calculerSalaire();
        }
        return somme / nbreEmploye;
    }
    public void afficherSalaires() {
        for (int i = 0; i < nbreEmploye; i++) {
                System.out.println(staff[i].getNom() + " gagne "+ staff[i].calculerSalaire() + "
        francs.");
        }
    }
```

QUESTION 2) parmi les méthodes qui sont définies et celles qui sont appelées dans cette classe Personnel, quelle est (ou quelles sont) la (ou les) méthodes polymorphes ?

---

3. Gestion d'albums photos

Vous devez écrire un programme qui permet de gérer des photos de vacances.
Une photo est caractérisée par le nom du pays dans lequel cette photo a été prise, une année de prise de vue, un commentaire, un nom de fichier et la taille du fichier.

1) Photo
Définissez la classe Photo dans le package photos.modeles.
On voudra pouvoir comparer deux photos sur leur année.
Vous définirez un constructeur qui permettra d'initialiser les attributs. L'attribut taille du fichier sera initialisé à l'aide de la classe File dont voici un exemple d'utilisation dont vous pouvez vous inspirer :

```
import java.io.File;
public class FileSizeExample {
   public static void main(String[] args)     {
      File file =new File("image.jpg");
       if(file.exists()){
         // Récupérer la taille du fichier
```

double bytes = file.length();
  }
2) Ecrire une classe AlbumPhoto dans le package photos.modeles. Celle classe devra permettre de représenter un ensemble de photos.
a) Définissez tout d'abord une interface qui permette de définir les spécifications : on voudra pouvoir :
ajouter une nouvelle photo dans cet ensemble,
trier les photos de l'ensemble selon l'année,
rechercher dans l'ensemble de photos un mot-clé dans les commentaires et retourner un arrayList de références des photos qui ont ce mot-clé dans leur commentaire.

b) Proposez ensuite une implémentation de cette spécification. Les photos seront stockées dans un ArrayList.
Pour information, l'instruction Collections.sort(arrayList) permet de trier l'arrayList passé en paramètre. Il faut bien sûr que les éléments stockés dans l'array list soient comparables...

---

4. Banque : interface Compte

Dupliquer votre projet Banque.
Définissez Compte comme une interface avec des spécifications de méthodes : débiter, crediter, getSolde, …
Implémentez ensuite cette interface pour gérer les comptes courants et les comptes d'épargne.

---

5. Mediane (paramètre d'une méthode de type interface)

Ecrire l'interface MinMax et les classes concrètes nécessaires pour pouvoir exécuter le programme ci-dessous.

```
class Mediane {
	// Polymorphisme au niveau du paramètre qui est de type "interface"
	int mediane(MinMax mm) {
		return((mm.minimum()+mm.maximum())/2) ;
	}
	public static void main (String args []) {
		Mediane med = new Mediane ();
		MinMaxPaire mmPaire = new MinMaxPaire ();
		mmPaire.setV1 (10);
		mmPaire.setV2 (20);

		System.out.println ("Valeur mediane pour une paire : " + med.mediane
(mmPaire));

		MinMaxTriplet mmTriplet = new MinMaxTriplet();
		mmTriplet.setV1 (10);
		mmTriplet.setV2 (20);
```

```
            mmTriplet.setV3 (30);
            System.out.println ("Valeur mediane pour un triplet : " +
med.mediane (mmTriplet));
      }
}
```

Résultat de l'exécution :

```
Valeur mediane pour une paire : 15
Valeur mediane pour un triplet : 20
```