Série 4 : Héritage, Polymorphisme Classes abstraites et interfaces

Exercice01

Etant donné le programme suivant :

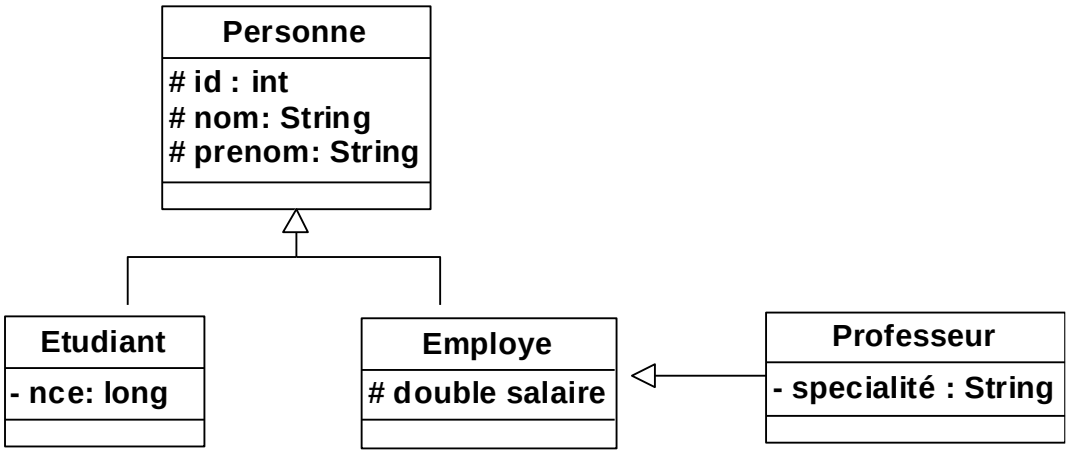
```
package com.java.exercice4;
public class Test {
	public static void main(String[] args) {
		C1 o1 = new C1();
		C1 o2 = new C11();
		C111 o3 = new C111();
		C11 o4 = new C111();
		C1 o5 = new C111();
	}
}
```

Déterminez si les instructions suivantes sont valides ou non. En cas d'erreur justifier rapidement pourquoi et préciser s'il y a erreur à la compilation ou à l'exécution. Les instructions suivantes sont indépendantes les unes les autres.

```
o1=o2;
o1=o3;
o3=o1;
o4=o5;
o3=(C111)o1;
o4=(C11)o5;
o4=(C111)o2;
o3=(C11)o5;
class c1{}
class c11 extends c1{}
class c111 extends c11{}
```

Exercice02

Soit le diagramme de classe suivant :



1- Développer les classes sachant que :
   - Le champ identifiant est incrémental.
   - Chaque classe doit contenir un constructeur d'initialisation.
   - Chaque classe doit redéfinir la méthode **toString ().**
2- Développer une classe **Application** dont on demande de créer :
   - Deux étudiants.
   - Deux employés.
   - Deux professeurs.
   - Afficher les informations de chaque personne.

Exercice03

Pour la gestion d'une bibliothèque on nous demande d'écrire une application manipulant des documents de nature diverse : des livres, des dictionnaires, etc. Tous les documents ont un numéro d'enregistrement et un titre. Les livres ont, en plus, un auteur et un nombre de pages, les dictionnaires ont une langue et un nombre d'articles. Ces divers objets doivent pouvoir être manipulés de façon homogène en tant que documents.

1- Définissez les classes **Document**, **Livre** et **Dictionnaire**. Donnez à chacune le constructeur qui prend autant arguments qu'il y a de variables d'instance à initialiser, et qui se limite à recopier les valeurs des arguments dans les variables correspondantes. Définissez également les accesseurs (getNumero (), getTitre (), etc.) nécessaires, ainsi que la méthode toString () habituelle.
2- Définissez la classe **ListeDeDocuments** permettant de créer une liste (vide) et d'y ajouter des documents. La liste peut être représentée par un objet java.util.Vector. Dans cette classe, définissez les méthodes
   - void add (Document d) qui ajoute un document à la liste,
   - void afficherTousLesDocuments() qui affiche tous les documents de la liste,
   - void afficherTousLesLivres () qui affiche tous les livres (uniquement) de la liste.
3- Définissez une classe **Bibliothèque** réduite à une méthode main permettant de tester les classes et méthodes précédentes, ainsi que celles que vous écrivez pour la question suivante.

Exercice04

Un compte bancaire possède à tout moment une donnée : son **solde**. Ce solde peut être positif (compte créditeur) ou négatif (compte débiteur).

Chaque compte est caractérisé par un **code** incrémenté automatiquement. A sa création, un compte bancaire a un solde nul et un code incrémenté. Il est aussi possible de créer un compte en précisant son solde initial.

Utiliser son compte consiste à pouvoir y faire des dépôts et des retraits. Pour ces deux opérations, il faut connaître le montant de l'opération.

L'utilisateur peut aussi consulter le solde de son compte par la méthode toString().

Un compte Epargne est un compte bancaire qui possède en plus un champ « TauxInterêt =6» et une méthode calculIntérêt() qui permet de mettre à jour le solde en tenant compte des intérêts.

Un ComptePayant est un compte bancaire pour lequel chaque opération de retrait et de versement est payante et vaut 5 dt.

- Définir la classe CompteBancaire.
- Définir la classe CompteEpargne.
- Définir la classe ComptePayant.
- Définir une classe contenant la fonction main() permettant de tester les classes CompteBancaire et CompteEpargne avec les actions suivantes:
  - Créer une instance de la classe CompteBancaire , une autre de la classe CompteEpargne et une instance de la classe ComptePayant
  - Faire appel à la méthode deposer() de chaque instance pour déposer une somme quelconque dans ces comptes.
  - Faire appel à la méthode retirer() de chaque instance pour retirer une somme quelconque de ces comptes.
  - Faire appel à la méthode calculInterêt() du compte Epargne.
  - Afficher le solde des 3 comptes.

Exercice05

Ecrire les classes nécessaires au fonctionnement du programme suivant, en ne fournissant que les méthodes nécessaires à ce fonctionnement :

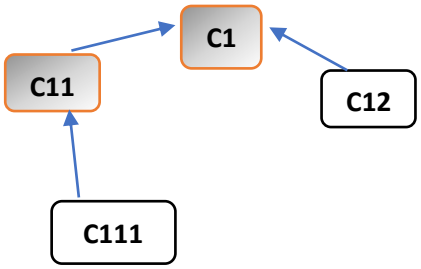
```
public class TestMetiers {
public static void main(String[] args) {
      Personne[] personnes = new Personne[4];
      personnes[0] = new Personne("Moez");
      personnes[1] = new Forgeron("Ali");
      personnes[2] = new Menuisier("Mohammed");
      personnes[3] = new Agriculteur("Amor");
      for(Personne e: personnes)
      e.affiche();
 }
}
```

Sortie de programme

```
Je suis Moez
Je suis Ali le forgeron.
Je suis Mohammed le menuisier.
Je suis Amor l'agriculteur.
```

Exercice06

Soit la hiérarchie de classes suivantes :



Les deux classes C1 et C11 son abstraites

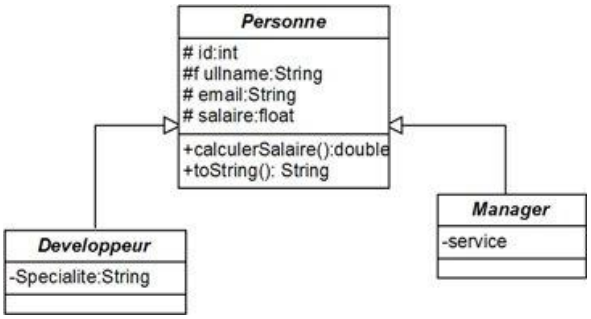
Indiquer pour chacun des cas suivants si les instructions sont acceptées par le compilateur ou non.

```
C1 x = new C1() ;
C1 y = new C11() ;
C1 z = new C111() ;
C1 t = new C12() ;
z=t ;
```

Exercice07

Le directeur des systèmes d'information de la société TOUBKAL-IT souhaite développer un module pour la gestion des utilisateurs de son service, pour cela il vous a fait appel pour réaliser cette tâche.

Le diagramme de classe a été établi par un analyste afin de mettre en place une base de données sous ORACLE ou MySQL :



1. Créer la classe abstraite « Personne ».
2. Créer les classes « Developpeur » et « Manager ». Qu'est-ce que vous remarquez ?

3. Redéfinir la méthode calculerSalaire (), sachant que :
   - Le développeur aura une augmentation de 20% par rapport à son salaire normal.
   - Le manager aura une augmentation de 35% par rapport à son salaire normal.
4. Créer deux développeurs et deux managers.
5. Afficher les informations des objets construit, sous la forme :

   Le salaire du manager LOGHMARI Mohamed est : 30 000 dt, son service : Informatique

   Le salaire du développeur SALIM karim est : 10 000 dt, sa spécialité : PHP

6. Créer un objet de type Personne. Qu'est-ce que vous remarquez ?

Exercice08

Compléter les classes nécessaires à l'exécution de ce programme tout en sachant que la classe Cylindre dérive de la classe Cercle qui est une sous classe de Point.

Indication :

```
public class TestForme {
	public static void main(String[] args) {
		int n = Integer.parseInt(args[0]);
		Forme f;
		switch(n){
		case 1:
			// un point est défini par les coordonnées x et y
			f=new Point(1,2);
			break;
		case 2:
			/* un cercle est défini par les coordonnées x et y de son
centre et par son rayon*/
			f=new Cercle(5,2,3);
			break;
		case 3:
			/* un cylindre est défini par les coordonnées x et y de
son centre et par son rayon et par sa hauteur*/
			f=new Cylindre(5,2,3,10);
			break;
		default:
			f = new Point(0,0);
		}
		f.affiche();
	}
}
```

Sortie du programme

```
Si n= 1, Je suis un point de coordonnée 1 ,2
Si n= 2, Je suis un cercle de surface 28,26
Si n= 3, Je suis un cylindre de surface 244,92
```

```
Autrement on obtient à l'écran, Je suis un point de coordonnée 0 ,0
```

Indication

- Surface d'un cercle = 3.14*rayon*rayon
- Surface d'un cylindre = 2*surface d'un cercle+ 2*3.14*rayon*hauteur

Exercice09

Cet exercice vous permettra de concevoir une hiérarchie de classes utilisant la notion d'interface. Il vous servira également de révision pour les notions d'héritage, de classes abstraites et de polymorphisme.

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

Un **employé** est caractérisé par son nom, son prénom, son âge et sa date d'entrée en service dans l'entreprise.

Codez une classe abstraite Employe dotée des attributs nécessaires, d'une méthode abstraite calculerSalaire (ce calcul dépendra en effet du type de l'employé) et d'une méthode getNom retournant une chaine de caractère obtenue **en concaténant la chaine de caractères "L'employé " avec le prénom et le nom**.

Dotez également votre classe d'un constructeur prenant en paramètre l'ensemble des attributs nécessaires.

Calcul du salaire

Le calcul du salaire mensuel dépend du type de l'employé. On distingue les types d'employés suivants :

- Ceux affectés à la *Vente*. Leur salaire mensuel est le 20 % du *chiffre d'affaire* qu'ils réalisent mensuellement, plus 400 dinars.
- Ceux affectés à la *Représentation*. Leur salaire mensuel est également le 20 % du *chiffre d'affaire* qu'ils réalisent mensuellement, plus 800 dinars.
- Ceux affectés à la *Production*. Leur salaire vaut le *nombre d'unités* produites mensuellement multipliées par 5.

- Ceux affectés à la *Manutention*[1]. Leur salaire vaut leur *nombre d'heures* de travail mensuel multipliées par 65 dinars.

Codez une hiérarchie de classes pour les employés en respectant les conditions suivantes :

- La super-classe de la hiérarchie doit être la classe Employe.
- Les nouvelles classes doivent contenir les attributs qui leur sont spécifiques ainsi que le codage approprié des méthodes calculerSalaire et getNom, en changeant le mot "employé" par la catégorie correspondante.
- 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

Employés à risques

Certains employés des secteurs production et manutention sont appelés à fabriquer et manipuler des produits dangereux.
Après plusieurs négociations syndicales, ces derniers parviennent à obtenir une prime de risque mensuelle. Complétez votre programme en introduisant deux nouvelles sous-classes d'employés. Ces sous-classes désigneront les employés des secteurs production et manutention travaillant avec des produits dangereux.
Ajouter également à votre programme une interface pour les employés à risque permettant de leur associer une prime mensuelle fixe de 200.

Collection d'employés

Satisfait de la hiérarchie proposée, notre directeur souhaite maintenant l'exploiter pour afficher le salaire de tous ses employés ainsi que le salaire moyen.

Ajoutez une classe Personnel contenant une "collection" d'employés. Il s'agira d'une collection polymorphique d'Employe.

Définissez ensuite les méthodes suivantes à la classe Personnel :

- void ajouterEmploye(Employe) : qui ajoute un employé à la collection.
- void afficherSalaires() : qui affiche le salaire de chacun des employés de la collection.

[1] La manutention désigne l'action de manipuler, de déplacer des marchandises, des colis ou des documents dans un lieu de production ou de stockage tel un entrepôt, une usine, un magasin, un bureau, etc.

- double salaireMoyen() : qui affiche le salaire moyen des employés de la collection.

Testez votre programme avec le code suivant :

```
class Salaires {
    public static void main(String[] args) {
        Personnel p = new Personnel();
        p.ajouterEmploye(new Vendeur("Pierre", "Business", 45, "1995", 30000));
        p.ajouterEmploye(new Representant("Léon", "Vendtout", 25, "2001", 20000));
        p.ajouterEmploye(new Technicien("Yves", "Bosseur", 28, "1998", 1000));
        p.ajouterEmploye(new Manutentionnaire("Jeanne", "Stocketout", 32, "1998",
45));
        p.ajouterEmploye(new TechnARisque("Jean", "Flippe", 28, "2000", 1000));
        p.ajouterEmploye(new ManutARisque("Al", "Abordage", 30, "2001", 45));

        p.afficherSalaires();
        System.out.println("Le salaire moyen dans l'entreprise est de " +
p.salaireMoyen() + " francs.");
    }
}
```