Atelier 3 : Héritage et Polymorphisme

Exercice 1 :

On dispose de la classe suivante:

```
class Point
{
public Point (int x, int y) { this.x = x ; this.y = y ; }
public void affiche() {
    System.out.println ("Coordonnees : " + x + " " + y) ;}
private int x, y ;
}
```

Réaliser une classe PointNom, dérivée de Point permettant de manipuler des points définis par leurs coordonnées (entières) et un nom (caractère). On y prévoira:

• un constructeur pour définir les coordonnées et le nom d'un objet de type PointNom,

• la méthode affiche pour afficher les coordonnées et le nom d'un objet de type PointNom.

Écrire un petit programme utilisant la classe PointNom.

Exercice 2 :

Soit les classes Point et PointNom ainsi définies :

```
class Point
{   public Point (int x, int y) { this.x = x ; this.y = y ;
}
    public static boolean identiques (Point a, Point b)
    { return ( (a.x==b.x) && (a.y==b.y) ) ; }
    public boolean identique (Point a)
    { return ( (a.x==x) && (a.y==y) ) ; }
    private int x, y ;
}
class PointNom extends Point
{   PointNom (int x, int y, char nom)
    { super (x, y) ; this.nom = nom ; }
    private char nom ;
}
```

1. Quels résultats fournit ce programme ? Expliciter les conversions mises en jeu et les règles utilisées pour traiter les différents appels de méthodes :

```
public class LimPoly
{ public static void main (String args[])
    { Point p = new Point (2, 4) ;
    PointNom pn1 = new PointNom (2, 4, 'A') ;
    PointNom pn2 = new PointNom (2, 4, 'B') ;
    System.out.println (pn1.identique(pn2)) ;
    System.out.println (p.identique(pn1)) ;
    System.out.println (pn1.identique(p)) ;
    System.out.println (Point.identiques(pn1, pn2)) ;
    }
}
```

2. Doter la classe PointNom d'une méthode statique identiques et d'une méthode identique fournissant toutes les deux la valeur **true** lorsque les deux points concernés ont à la fois mêmes coordonnées et même nom. Quels résultats fournira alors le programme précédent ? Quelles seront les conversions mises en jeu et les règles utilisées ?

Exercice 3 :

A: Déclarer une classe Personne qui contient:

1- quatre attributs privés: nom, prenom et cin de type String et age de type int;

2- les méthodes getters et setters pour renvoyer et modifier la valeur de chaque attribut;

3- une méthode afficherInformation qui affiche à la console le nom, le prenom, le cin et l'age;

4- un constructeur qui initialise les attributs par les valeurs données en paramettre;

B: Déclarer une classe Etudiant qui étend la classe Personne et qui contient;

5- un attribut privé cne de type String;

6- deux méthodes ; getter et setter pour renvoyer et modifier la valeur du cne;

7- une méthode afficherInformation qui affiche au console le nom, le prenom, le cin, le l'age et le cne;

8- deux constructeurs :

a-un constructeur qui initialise les attributs : nom, prenom, cin et cne à null, et age à 0;

b-un constructeur qui initialise tous attributs par les valeurs données en parametre;

C: dans la méthode main:

9- Créer deux objets : etudiant de type Etudiant et personne de type Personne;

10-Invoquer la méthode afficherInformation pour les deux objets : etudiant et personne.

Exercice 4 :

Il s'agit de permettre la création de points, cercles et cylindres. Pour cela, on considère qu'un cercle étend la notion de point pour caractériser son rayon. De même, un cylindre étend la notion de cercle.

1. Vous devez implémenter une classe Point qui sera la super classe de la classe Cercle et donc transitivement de la classe Cylindre. La classe Point comporte 3 méthodes : calcul de l'aire (calculerAir revoie 0), calcul du volume (calculerVolume renvoie 0) et affichage du nom de la classe, respectivement “point”, “cercle” et “cylindre” pour nos 3 classes. La classe Point est caractérisée par 2 coordonnées réelles : x,y. La classe doit permettre d'accéder et de modifier ces coordonnées. Elle implémente également une méthode toString() affichant les coordonnées du point.
2. La classe Cercle étend la classe Point, elle comporte également une valeur réelle pour son rayon. Vous devez écrire 2 constructeurs pour cette classe :

• le premier va créer un cercle dont le centre est le point (0,0) et de rayon 0;

• le second comporte des paramètres pour les coordonnées du centre et le rayon.

De même que pour le point, vous devez proposer une méthode toString() affichant toutes les informations sur le cercle (coordonnées du centre, valeur du rayon).

Enfin le cercle propose une méthode pour le calcul de l'aire.

3. La classe Cylindre étend la classe Cercle et comporte une valeur réelle hauteur.

• Implémenter un get et un set pour la hauteur.

• Implémenter les méthodes calculerAire() et calculerVolume() pour le cylindre. Aire vaut le périmètre du cercle*hauteur. Le volume vaut l’aire du cercle* hauteur et.

• La méthode toString() permet d'obtenir toutes les informations sur le cylindre : centre du cercle, rayon du cercle et hauteur.

4. Vous devez tester l'implémentation en écrivant une classe TesterForme.

La classe TesterForme comporte un tableau de formes.

• ajouter un point avec x=5 et y=5 au tableau.

• ajouter un cercle avec un rayon de 8.5 et des coordonnées x=10,y=15 pour centre, au tableau.

• ajouter un cylindre avec une hauteur de 20, un rayon de 12,5 et des coordonnées x=30,y=30, au tableau.

• et afficher les coordonnées, les aires, les volumes et le nom des 3 formes.