Travaux pratiques
séance n°8 - corrigé

Héritage et Polymorphisme

Exercice 1

Question 1 : construire une classe représentant des cercles. Par défaut chaque instance de cercle a un rayon égal à 1. . Implanter les accesseurs de la variable d'instance (`getRayon` et `setRayon`) et une méthode de calcul de surface (`rayon*rayon*Math.PI`).

Solution

```
public class Cercle{
    protected double rayon;

    public Cercle(){
        rayon = 1.0;
    }

    public Cercle(double rayon){
        this.rayon = rayon;
    }

    public double getRayon(){
        return rayon;
    }

    public void setRayon(double rayon){
        this.rayon = rayon;
    }

    public double surface(){
        return rayon*rayon*Math.PI;
    }
}
```

Question 2 : modifier la méthode `setRayon` de manière à gérer par une exception le cas où le rayon passé en paramètre est négatif.

Solution

```
public class RayonNegatifException extends Exception{}
public void setRayon(double rayon)throws RayonNegatifException {
      if( rayon>=0.)
        this.rayon = rayon;
      else throw new RayonNegatifException();
}
```

Question 3 : on peut considérer qu'un cylindre est un cercle dont l'épaisseur est quelconque. Construire une classe `Cylindre` qui dérive de la classe `Cercle`. Par défaut, sa hauteur vaut 1. Ajouter un constructeur qui initialise les variables d'instance, puis une méthode pour calculer son volume (2*surface du cercle+2*Math.PI*hauteur). Quelle nouvelle visibilité pour la variable d'instance de `Cercle` ?

Solution

```
public class Cylindre extends Cercle{
    private double hauteur;

    public Cylindre(){
        super();
        hauteur = 1.0;
    }

    public Cylindre(double rayon, double hauteur){
        super(rayon);
        this.hauteur = hauteur;
    }

    public double getHauteur(){
        return hauteur;
    }

    public void setHauteur(double hauteur){
        this.hauteur = hauteur;
    }

    public double surface(){
        return 2*super.surface()+2*rayon*Math.PI*hauteur;
    }

    public double volume(){
        return super.surface()*hauteur;
    }
}
```

Question 4 : écrire un programme de test qui créé une instance de cylindre, affiche sa hauteur, son rayon, son volume et la surface de son cercle.

Solution

```
public class Main{
    public static void main(String[] args){
        Cylindre c = new Cylindre();
        System.out.println("hauteur : "+c.getHauteur());
        System.out.println("rayon   : "+c.getRayon());
        System.out.println("volume  : "+c.volume());
        System.out.println("surface : "+c.surface());
    }
}
```

Question 5 : l'héritage permet la factorisation des membres d'une hiérarchie de classes. Vérifiez que vous avez utilisé toutes les possibilités de l'héritage (partage de variables, redéfinition de méthodes).

Solution

```
Voir solutions précédentes
```

Question 6 : la pseudo variable **this** est souvent implicite. Modifier la classe `Cylindre` de manière à la rendre explicite partout où c'est possible. On peut ainsi économiser les identificateurs.

Solution

```
Voir solution question 3
```

Question 7 : la pseudo variable **`super`** est souvent implicite. Modifier la classe `Cylindre` de manière à la rendre explicite.

Solution

```
Voir solution question 3
```

Question 8 : ajouter une méthode de calcul de la surface d'un cylindre (surface du cercle +2*rayon*Math.PI*hauteur du cylindre).

Solution

```
Voir solution question 3
```

Question 9 : compléter le programme de test suivant (utilisation de l'opérateur **`instanceof`**) :

```
Cercle c = new Cylindre();
c.setRayon(4.5);
……..
……..
System.out.printtln(" le volume de mon cylindre est : " +……);
```

Solution

```
Cercle c = new Cylindre();
c.setRayon(4.5);
if(c instanceof Cylindre){
     Cylindre cyl = (Cylindre)x;
     System.out.println(
             "le volume de mon cylindre est : "+cyl.volume());
}
```

Question 10 : compléter le programme de test avec une méthode qui affiche la surface d'un cercle et d'un cylindre ainsi que le volume du cylindre.
La signature de la méthode est : **`static`** `void afficher( Cercle cercle)`

Solution

```
public static void afficher(Cercle c){
        if(c instanceof Cylindre){
            Cylindre cyl = (Cylindre)c;
            System.out.println("surface du cylindre : "+cyl.surface());
            System.out.println("volume du cylindre : "+cyl.volume());
        }
        else
            System.out.println("surface du cercle : "+c.surface());
    }
```

Question 11 : le programme de test créé 2 instances, l'une de `Cercle` et l'autre de `Cylindre`. Il affiche ensuite ces 2 objets en exécutant la méthode précédente.

Solution

```
Cylindre x = new Cylindre(4.5,3.);
Cercle y = new Cercle(4.5);
afficher(x);
```

```
afficher(y);
```

Question 12 : construire un tableau constitué d'instances de `Cercle` et `Cylindre`. Parcourir ce tableau afin d'afficher les surfaces respectives de chaque objets contenus dans ce tableau.

Solution

```
Cercle[] tab = new Cercle[3];
tab[0]=x;
tab[1]=y;
tab[2]=c;
for(int i=0;i<tab.length;i++)
     afficher(tab[i]);
```