Université de Tlemcen
Faculté des Sciences
Département d'Informatique

Année universitaire 2019-2020
2ème année licence informatique
Programmation Orientée Objet

TP N° 4

Exercice 1

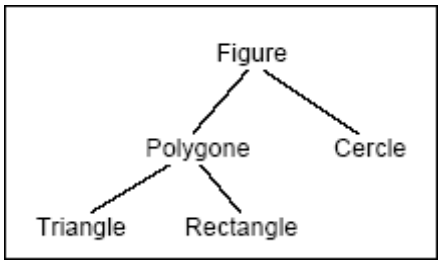
1. Ecrire une classe **abstraite** « **Volaille** ». Une volaille est un oiseau élevé dans une ferme, caractérisée par un « numéro » et un « poids ». Ajouter à cette classe un constructeur et une méthode « ChangePoids » qui permet de modifier son poids. Ecrire deux autres méthodes abstraites, l'une retourne son prix, et l'autre si la volaille est assez grosse pour être abattue. En effet, le prix et l'estimation de la grosseur dépend du type de la volaille (Poulet, canard, dinde, etc.).

2. Ecrire une classe « **Poulet** » qui **spécifie** la classe « Volaille ». Tous les poulets ont le même prix de vente au kilo, et le même poids d'abattage. Cependant, pour des raisons de marché, le prix de vente et le poids d'abattage peuvent changer.

3. Ecrire une classe « **Main** » qui permettra de tester la classe « Poulet »

   NB. Vous pouvez ajouter à votre convenance d'autres **méthodes** dans les classes « **Volaille** » et « **Poulet** »

Exercice 2



Soit l'arbre d'héritage de classes d'objets géométriques.
On vous demande d'implémenter ces différentes classes.

1. Pour cela vous allez commencer par définir une classe **Point** qui servira dans les autres classes. La classe **Point** possède :
   - Deux attributs réels **abscisse** et **ordonnée**.
   - Un **constructeur** possédant **deux paramètres réels** x et y.
   - Un constructeur possédant un paramètre p de type **Point**.
   - Une méthode **statique** qui retourne la distance entre deux points.

   Remarque :
   - Soit deux points $M_1(x1,y1)$ et $M_2(x2,x2)$. La distance entre M_1 et M_2 est : $\sqrt{(x2-x1)^2+(y2-y1)^2}$
   - Math.sqrt(a) est une méthode qui retourne la racine carré de a en java

2. Définissez la classe abstraite **Figure** constituée de :
   - Un attribut **Point** statique, nommé **zéro** qui a pour coordonnées (0,0) ;
   - Un attribut **Point,** nommé **origine** ;
   - Un constructeur **sans paramètre** où **origine** est initialisé par **zéro ;**
   - Un deuxième constructeur avec un paramètre **Point** p ;
   - Deux méthodes abstraites **afficher()** et **périmètre()**.

3. Définissez la classe **Cercle** qui dérive de la classe Figure qui possède :
   - Deux attributs : **centre** (correspond à origine de Figure) et **rayon** ;
   - Un constructeur avec **deux paramètres : le centre et le rayon** ;
   - Complétez cette classe afin qu'elle puisse instancier des objets Cercle.

   Remarque :
   - Périmètre d'un cercle est : 2*Math.PI* rayon

4. La classe **Polygone** dérive de la classe Figure. Elle se caractérise par :
   - Un tableau de Point nommé sommets ;
   - Un entier **nbs** qui représente le nombre de ces sommets ;
   - Un constructeur où **nbs** est initialisé à 0 ;
   - Un constructeur **Polygone(Point[] m, int n),** n désigne le nombre de sommets ;
   - Complétez cette classe afin qu'elle puisse instancier des objets Polygone.

5. La classe **Triangle** dérive de **polygone** et possède 03 sommets.

6. La classe **Rectangle** dérive de la classe Polygone. Elle est définie avec :
   - Ses **caractéristiques** mathématiques classiques, c'est à dire la longueur et la largeur de ses cotés et un de ses sommets, le sommet inférieur gauche.
   - Un constructeur **Rectangle(Point m, double long, double larg).**

7. Complétez la classe principale Geométrie.

```
class Geometrie {
        public static void main(String args[]) {
                Point P1= new Point(3,4);
                Point P2= new Point(4,4);
                Point P3= new Point(0,0);
                Point P4= new Point(1,0);
                Point P5= new Point(0,1);
                Point [] TabP={P3, P4, P5};  // initialisation d'un tableau avec 03 sommets

                Cercle c= …….
                Rectangle r= ……

                Figure f; //autorisé, mais pas new Figure (..) !

                // Initialiser f par un objet Cercle et affichez ses propriétés.
                …….
                // Afficher seulement le rayon de f
                ……

                // Initialiser f par un objet Rectangle et affichez ses propriétés.
                …….

                Figure t = ……..// t référence un objet Triangle
                // Affichez les propriétés de t
                ……

        }
}
```

Exercice 4

```
abstract class Volaille{
        double poids;
        int identite;
        Volaille ( double p , int i ){ poids = p; identite = i ;}
        void changePoids ( double np ){poids = np ;}
        abstract double prix ( ) ;
        abstract boolean assezGrosse ( ) ;
}
--------------------------------
class Poulet extendsVolaille{
        static double prixAuKilo = 1.0 ;
        static double poidsAbattage = 1.2 ;
        Poulet ( double p , int i ){super ( p , i ) ;}
        static void changePrix ( double x ){prixAuKilo = x ;}
        static void changePoidsAbattage ( double x ){poidsAbattage = x ;}
        double prix ( ) {return poids * prixAuKilo ;}
        boolean assezGrosse ( ) {return poids >= poidsAbattage ;}
}
```

```
class Point {
    double abscisse;
    double ordonnee;
    Point(double x, double y) {abscisse=x; ordonnee=y;}
    Point(Point p){abscisse=p.abscisse; ordonnee=p.ordonnee;}
    static double distance(Point p, Point q) {
        double dx=p.abscisse-q.abscisse;
        double dy=p.ordonnee-q.ordonnee;
        return Math.sqrt(dx*dx+dy*dy);
   }
}
```

```
abstract class Figure {
    private static final Point zero=new Point(0,0);
    Point origine;
    Figure(){origine=zero;}
    Figure(Point p){origine=new Point(p);}
    abstract double perimetre();
    abstract void affiche();
}
```

```
class Cercle extends Figure {
    static final double pi=3.141592;
    //  le centre est hérité de Figure (origine)
    double rayon;
    Cercle(Point centre, double r){super(centre); rayon=r;}
    double perimetre() {return 2*pi*rayon;}
    void affiche() {
        System.out.println("Cercle");
        System.out.println("rayon:"+rayon+" et centre:"+"(" +origine.abscisse +","+origine.ordonnee+")");
    }
}
```

```
class Polygone extends Figure {
    Point sommet[]= new Point[100];
    int nbs;
    Polygone(){nbs=0;}
    Polygone(Point[] m, int n) {
        super(m[0]);
        nbs=n;
        for (int i=0; i<n; i++)
                sommet[i]=m[i];
    }
    double lcote(int i) {
         if (i<nbs)
                return Point.distance(sommet[i-1], sommet[i]);
         else
                return Point.distance(sommet[i-1], sommet[0]);
        }
        double perimetre() {
```

```
        double somme=0;
        for (int i=1; i<=nbs; i++)
          somme += lcote(i);
        return somme;
      }
   void affiche() {
        System.out.println("Polygone");
        for (int i=0; i<nbs; i++)
                System.out.print("(" + sommet[i].abscisse +"," + sommet[i].ordonnee + ") ");
        System.out.println();
   }
}
```

```
   class Triangle extends Polygone {
        Triangle(Point[] m) { super(m,3); }
   }
```

```
   class Rectangle extends Polygone {
      double largeur;
      double longueur;
      Rectangle(Point m, double lo, double la) {
        Point P1= new Point(m.abscisse+ lo, m.ordonnee);
        Point P2= new Point(m.abscisse, m.ordonnee+ la);
        Point P3= new Point(m.abscisse+ lo, m.ordonnee+ la);
        Point mr[]={m, P1, P3, P2};
        sommet=mr;
        nbs=4;
        largeur= la;
        longueur= lo;
      }
   }
```

```
   class Geometrie {
      public static void main(String args[]) {
        Point P1= new Point(3,4);
        Point P2= new Point(4,4);
        Point P3= new Point(0,0);
        Point P4= new Point(1,0);
        Point P5= new Point(0,1);
        Point [] TabP={P3, P4, P5};
        Cercle c= new Cercle(P1,2);
        Rectangle r= new Rectangle(P2,5,2);
        Triangle t= new Triangle(TabP);
        Figure f; //autorise, mais pas new Figure !
        System.out.println("perimetre cercle");
        f=c; f.affiche(); // appel de affiche de Figure
        System.out.println( ((Cercle)f).rayon);

        System.out.println("perimetre : " + f.perimetre());
        f=r; f.affiche();
```

```
        System.out.println("perimetre : " + f.perimetre());
        f=t; f.affiche();
        System.out.println("perimetre : " + f.perimetre());
    }
}
```

L'exécution du programme affiche :

```
perimetre cercle
Cercle
rayon:2.0 et centre:(3.0,4.0)
2.0
perimetre : 12.566368
Polygone
(4.0,4.0) (9.0,4.0) (9.0,6.0) (4.0,6.0)
perimetre : 14.0
Polygone
(0.0,0.0) (1.0,0.0) (0.0,1.0)
perimetre : 3.414213562373095
```