# MOOC Intro POO Java
# Tutoriels semaine 3

Les tutoriels sont des exercices qui reprennent des exemples du cours et dont le corrigé est donné progressivement au fur et à mesure de la donnée de l'exercice lui-même.

---

Ils sont conseillés comme un premier exercice sur un sujet que l'étudiant ne pense pas encore assez maîtriser pour aborder par lui-même un exercice «classique».

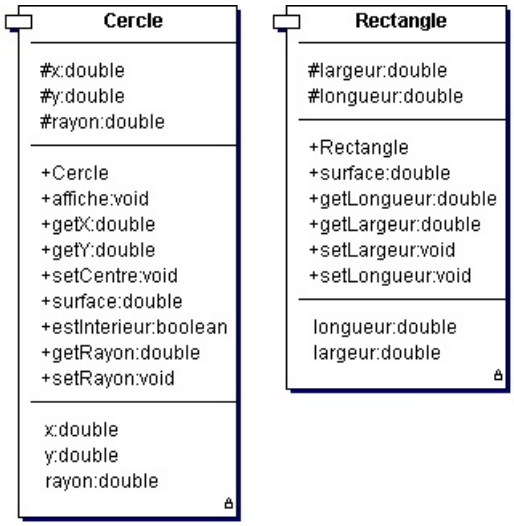
### Semaine 3 : Figures géométriques

Le but de cet exercice est d'illustrer la notion d'héritage en utilisant une hiérarchie de figures géométriques. Pour simplifier, nous laisserons tous les accès de méthodes publics dans cet exercice.

Le programme suivant comporte les définitions des classes `Rectangle` et `Cercle`, ainsi qu'une ébauche de `main`:

```
class FiguresGeometriques {

    public static void main(String[] args) {
        // A COMPLETER
    }
}

class Rectangle {

    private double largeur;
    private double longueur;

    public Rectangle(double largeur, double longueur) {
        this.largeur = largeur;
        this.longueur = longueur;
    }

    public double surface() {
        return largeur * longueur;
    }

    public double getLongueur() {
        return longueur;
    }

    public double getLargeur() {
        return largeur;
    }

    public void setLargeur(double l) {
        largeur = l;
    }

    public void setLongueur(double l) {
        longueur = l;
    }
}

class Cercle {
    // abscisse du centre
    private double x;
    // ordonnée du centre
    private double y;
    private double rayon;

    public Cercle(double x, double y, double r) {
        this.x = x;
        this.y = y;
        rayon = r;
    }

    public void affiche() {
        System.out.println("centre = (" + x + ", " + y + ")");
    }
```

```
    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public void setCentre(double x, double y) {
        this.x = x;
        this.y = y;
    }

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

    public boolean estInterieur(double x, double y) {
        return (((x - this.x) * (x - this.x) + (y - this.y) * (y - this.y))
                    <= rayon * rayon);
    }

    public double getRayon() {
        return rayon;
    }

    public void setRayon(double r) {
        if (r < 0.0) r = 0.0;
        rayon = r;
    }
}
```




Commençons par y ajouter la classe `RectangleColore` qui hérite de `Rectangle`.
Cette classe doit simplement avoir un attribut de plus, `couleur`, disons de type `int`.

**[Essayez de le faire par vous même avant de regarder la solution qui suit]**

---

Ceci se fait très simplement :

1. la classe `RectangleColore` hérite de `Rectangle` :

   ```
   class RectangleColore extends Rectangle {
   }
   ```

2. et a un attribut de plus:

   ```
   class RectangleColore extends Rectangle {
       private int couleur;
   }
   ```

Pour rendre notre nouvelle classe plus réaliste (et utilisable) ajoutons lui un constructeur qui fera appel au constructeur de la classe parente :

```
    public RectangleColore(double larg, double longueur, int c){
        super(larg, longueur);
        couleur = c;
    }
```

Testez avec ce `main` simple :

```
public static void  main(String[] args) {
    RectangleColore r = new RectangleColore(4.3, 12.5, 4);
    System.out.println(r.getLargeur());
}
```

(que vous aurez complété dans le fichier fourni)

Continuez en définissant la classe `Figure`. Celle-ci contient:

- Deux attributs `x` et `y` qui représentent le centre de la figure
- Une méthode `affiche()` qui affiche simplement les coordonnées du centre
- Un constructeur prenant les coordonnées du centre en paramètre

Faites ensuite hériter les classes `Cercle` et `Rectangle` de la classe `Figure`.

**[Essayez de le faire par vous même avant de regarder la solution qui suit]**

---

Pour la classe `Figure`, rien de particulier, et il suffit de **déplacer** les définitions correspondantes de la classe `Cercle` à la classe `Figure` (donc les supprimer de `Cercle`).

Puis on ajoute la méthode `affiche` et le constructeur :

```
class FiguresGeometriques {

    public static void main(String[] args) {
        RectangleColore r = new RectangleColore(1.2,3.4,12.3,43.2,4);
        r.affiche();

        Cercle c = new Cercle (2.3, 4.5, 12.2);
        c.affiche();
    }
}

class Figure {
    // abscisse du centre
    private double x;
    // ordonnee du centre
    private double y;

    public Figure(double x , double y){
        this.x = x;
        this.y = y;
    }

    public void affiche() {
        System.out.println("centre = (" +  x + ", " + y + ")");
    }

    public double getX() {
```

Venons-en à l'héritage proprement dit. Il faut spécifier les liens d'héritage aux deux classes concernées.

```
class Rectangle extends Figure { ...
class Cercle extends Figure { ...
```

puis modifier les constructeurs de manière appropriée :

```
    //constructeur de la classe Cercle
    public Cercle(double x, double y, double r) {
        super(x,y);
        rayon=r;
    }
...
    //constructeur de la classe Rectangle
    public Rectangle(double x, double y,double larg, double longueur){
        super(x,y);
        this.largeur = larg;
        this.longueur = longueur;
    }
...
    //constructeur de la classe RectangleColore
    public RectangleColore(double x, double y,double larg, double longueur, int c){
        super(x,y,larg, longueur);
        couleur = c;
    }
```

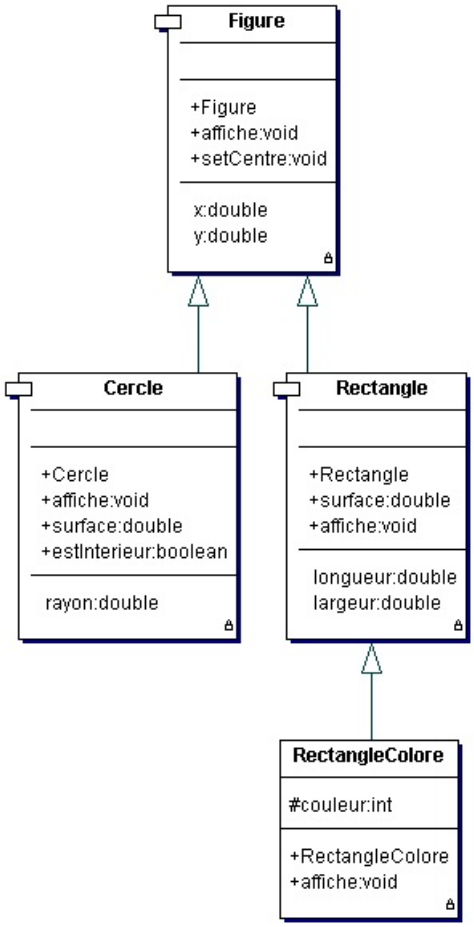
Pour la méthode d'affichage, supposons que l'on souhaite que, pour la classe `Cercle`, la méthode affiche également le rayon (mais continue d'afficher le centre).

Cela se fait en définissant dans la classe `Cercle` une méthode `affiche` utilisant la méthode masquée `affiche` de la super classe `Figure` :

```
class Cercle extends Figure {
 ...
    public void affiche(){
        super.affiche();
        System.out.println("rayon = " + rayon);
    }
..
}
```

On peut bien sûr faire de même avec les classes `Rectangle` et `RectangleColore` :

```
..
//dans la classe Rectangle
    public void affiche(){
        super.affiche();
        System.out.println("Largeur = " + largeur);
        System.out.println("Longeur = " + longueur);
    }
..
//dans la classe RectangleColore
    public void affiche(){
        super.affiche();
        System.out.println("Couleur = " + couleur);
    }
```




La méthode `estInterieur` de la classe `Cercle` doit être modifiée car l'accès direct aux attributs `x` et `y`, désormais hérités de `Figure`, n'est plus possible. Il faut passer par des getters.

Testez le programme avec la méthode `main` suivante :

```
    public static void main(String [] args) {
        RectangleColore r = new RectangleColore(1.2,3.4,12.3,43.2,4);
        r.affiche();

        Cercle c = new Cercle (2.3, 4.5, 12.2);
        c.affiche();
    }
```

[Le code source final est disponible sur la page des exercices sous le nom `FiguresGeometriques.java`]

---