P.O.O. (Programmation Orientée Objet)

CHOUITI Sidi Mohammed

Cours pour L2 en Informatique

Département d'Informatique

Université de Tlemcen

2019-2020

Plan du cours

1. Introduction à la Programmation Orientée Objet
2. Classes et Objets
3. Héritage, polymorphisme et Abstraction
4. Interface, implémentation et Paquetage
5. Classes Courantes en Java
6. Gestion des Exceptions
7. Interfaces graphiques

Un objet est une abstraction d'un élément du monde réel.

Il possède un ensemble d'attributs caractérisant son état, et un ensemble d'opérations (les méthodes) qui permettent son comportement.

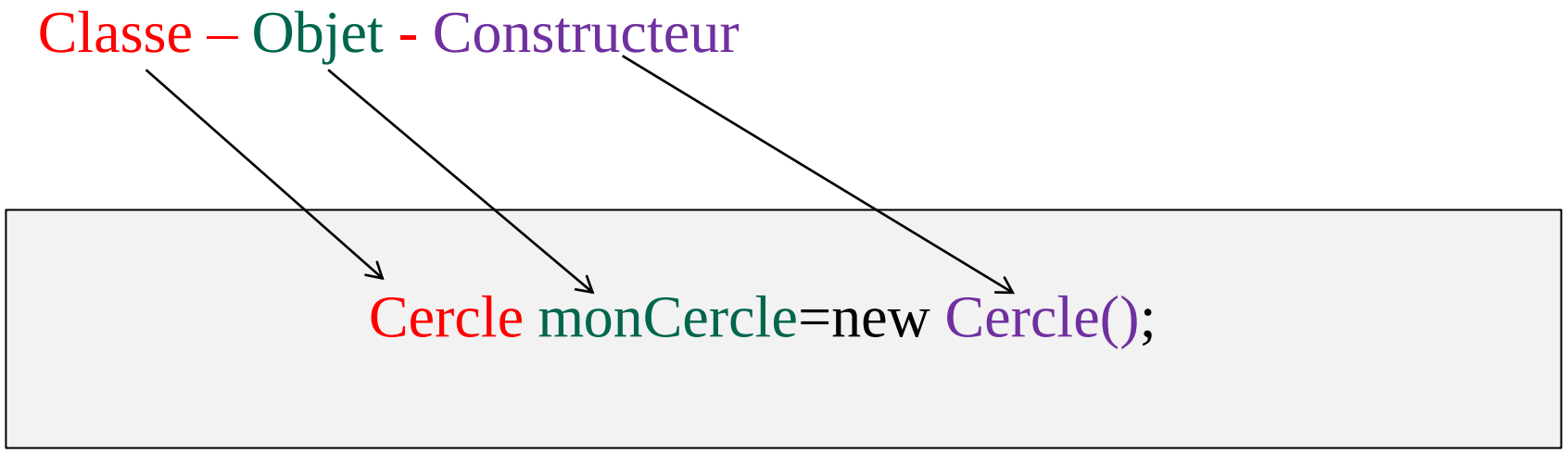
Un objet est l'instance d'une classe.

Une classe, est un type de données abstrait, caractérisé par des propriétés (ses attributs et ses méthodes) communes à des objets.

Elle permet de créer ces objets possédant ces propriétés.

Classe = attributs + opérations

Objet = état (attributs) + comportement (méthodes)


Classe – Objet - Constructeur
Cercle monCercle=new Cercle();

Définition d'une classe en java

```
class NomClasse {

    // définition des Attributs

    // définition des Méthodes

                    }
```

Les attributs de classe

Fichier "Cercle.java"

```
class Cercle {
      Point2D centre;
      double rayon;
                  }
```

Fichier « Point2D.java"

```
class Point2D {
            double x,y;
                        }
```

Les méthodes

Fichier "Cercle.java"

```
class Cercle {
    Point2D centre;
    double rayon;

    void deplacer (Vecteur2D vecteur) {
        centre.x += vecteur.x;
        centre.y += vecteur.y;
                                    }
            }
```

Les méthodes

Fichier "Cercle.java"

```
class Cercle {
        Point2D centre;
        double rayon;

        void deplacer (Vecteur2D vecteur) {
                centre.x += vecteur.x;
                centre.y += vecteur.y;
                                                }
                        }
```

Fichier "Vecteur2D.java"

```
class Vecteur2D {
                double x,y;
        }
```

La surcharge de méthodes

Fichier "Cercle.java"

```
class Cercle {
        Point2D centre;
        double rayon;

        void deplacer (Vecteur2D vecteur) {
                centre.x += vecteur.x;
                centre.y += vecteur.y;
                                                }
        void deplacer (double x, double y) {
                centre.x += x;
                centre.y += y;
                                                }
                }
```

Les méthodes

Fichier « Point2D.java"

```
class Point2D {

        double x,y;

        void afficher(){
            System.out.print("[" + x + ", " + y + "]");
                        }

            }
```

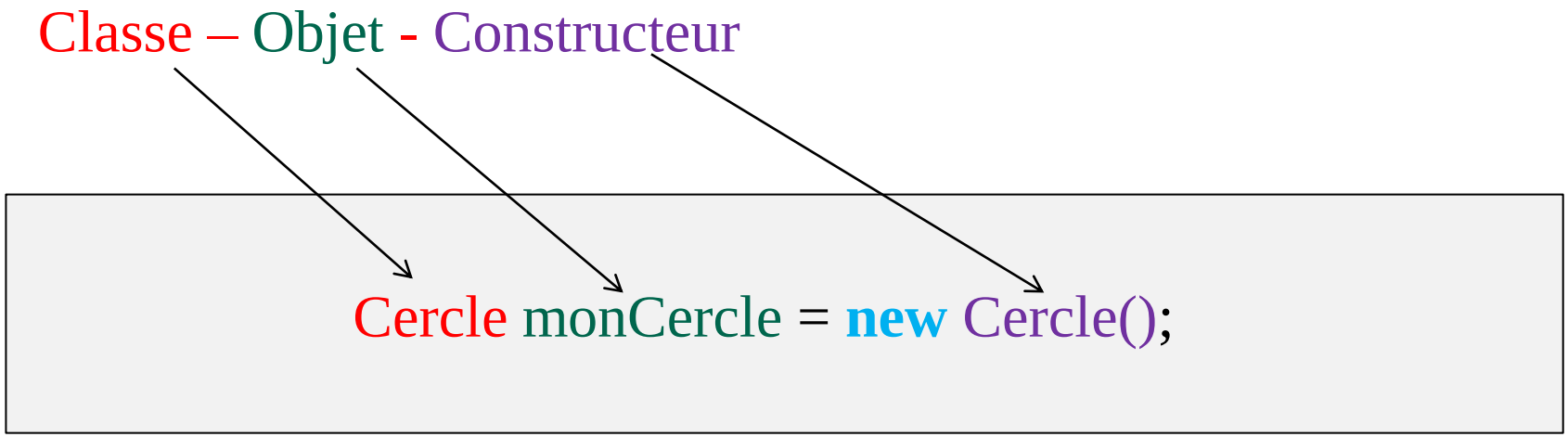
Appel de méthodes

Fichier "Cercle.java"

```
class Cercle {
    Point2D centre;     double rayon;
    void deplacer (Vecteur2D vecteur) {   centre.x += vecteur.x;     centre.y += vecteur.y;       }
    void deplacer (double x, double y) {   centre.x += x;  centre.y += y;                       }

    void deplacerH(double x) { centre.x += x; }
    void deplacerV(double y) { centre.y += y; }

    void afficher(){
        System.out.print("Objet Cercle :\n\tcentre : ");
        centre.afficher();
        System.out.println("\n\trayon : " + rayon);
                    }
            }
```

Instanciation d'objets

On instancie un objet en appliquant l'opérateur **new** sur un **constructeur** de classe.



Les constructeurs

Fichier "Cercle.java"

```
class Cercle {
    Point2D centre;     double rayon;
    Cercle(){
            centre=new Point2D(); rayon=1;
            }
    Cercle(Point2D c,double r){
            centre=c; rayon=r;
                                }
    void deplacer (Vecteur2D vecteur) {   centre.x += vecteur.x;     centre.y += vecteur.y;       }
    void deplacer (double x, double y) {   centre.x += x;  centre.y += y;                        }

    void deplacerH(double x) { centre.x += x; }
    void deplacerV(double y) { centre.y += y; }

    void afficher(){  System.out.print("Objet Cercle :\n\tcentre : ");     centre.afficher();
                      System.out.println("\n\trayon : " + rayon);                                 }
                }
```

Les constructeurs

Fichier « Point2D.java"

```
class Point2D {
        double x,y;

        Point2D(){
                x=y=0;
                }

        Point2D(double i, double j){
                x=i; y=j;
                                }

        void afficher(){    System.out.print("[" + x + ", " + y + "]");    }

                }
```

Les constructeurs

Fichier « Vecterus2D.java"

```
class Vecteur2D {
        double x,y;

        Vecteur2D(){
                x=y=0;
                    }

        Vecteur2D(double i, double j){
                x=i; y=j;
                                        }

                }
```

Classe principale

Fichier « Demarrer.java"

```
class Demarrer {

    public static void main(String args[]){

        Cercle    c1=new Cercle();
        Point2D   c=new Point2D(5,4);
        Cercle    c2=new Cercle(c,3);

        c1.afficher();
        c2.afficher();
                                        }
            }
```

Quelques règles sur les constructeurs

•Si aucun constructeur n'est spécifié, dans la définition de la classe, un constructeur par défaut vous est obligatoirement fourni, celui-ci n'admettant aucun paramètre.

•Si vous en définissez au moins un, le constructeur par défaut (qui n'admet pas de paramètres) n'est plus fourni. Si vous en avez l'utilité il vous faudra alors le définir explicitement.

Destructeur

```
class Point2D {
      //…
       public void finalize() {
            System.out.println(" Objet Point2D détruit");
       }
}
```

Le ramasse-miettes

Le ramasse-miettes (ou GC [Garbage Collector]) se charge de repérer les objets inutiles et de libérer leurs espaces mémoires
Il fonctionne en permanence dans un thread de faible priorité.

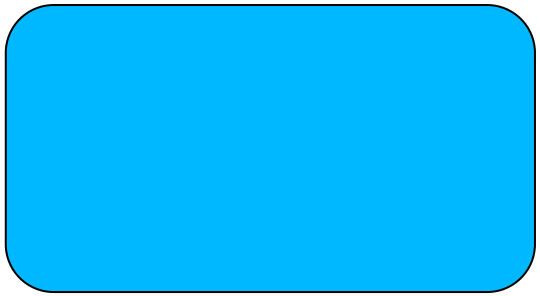
Méthodes et attributs statiques

Toute propriété (attribut ou méthode) **statique** existe **indépendamment** de toute instanciation d'objet.

```
class Demarrer {

      static int a = 3;

      static public void main(String args[]){
          a += 5;
          System.out.println("a^2 = " + carre(a));
      }

       static int carre(int v){        return v*v;     }
}
```

Méthodes et attributs statiques

```
class Demarrer {
    static int a = 6;

    static public void main(String args[]){
        Demarrer d1=new Demarrer(), d2=new Demarrer();
        d1.a++;   d2.a++;
        System.out.println("a = " + Demarrer.a);
    }
}
```



Méthodes et attributs statiques

- Les **propriétés statiques** d'une classe sont **partagées** par **toutes les instances** de cette classe.

- Si une **méthode est statique**, et si elle doit utiliser des attributs ou des méthodes **de sa classe**, il faut alors que ces propriétés soient elles aussi déclarées **static.**

A télécharger

Cours et TP en POO pour L2 en Informatique

https://gl2site.wordpress.com