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
4. Abstraction et déclaration finale
5. Interface, implémentation et Paquetage
6. Classes Courantes en Java
7. Gestion des Exceptions
8. Interfaces graphiques

Classe et méthode abstraites

Fichier « Forme.java"

```
class Forme{
        Point2D centre;

        Forme(){ centre=new Point2D(); }
        Forme(Point2D c){ centre=c;  }
        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; }

        double surface() { return 0; }

        void afficher(String nature){
                System.out.print("Objet " + nature+" :\n\tcentre : ");
        centre.afficher();     }
}
```

Classe et méthode abstraites

Fichier « Carre.java »

```
class Carre extends Forme {
    double longueur;

    Carre(){ longueur=1; }
    Carre(Point2D c,double l){ super(c);   longueur=l; }

    double surface() { return longueur * longueur; }

    void afficher(){
        super.afficher("Carre");
        System.out.println("\n\tlongueur : " + longueur);
                    }
                                }
```

Classe et méthode abstraites

Fichier « Cercle.java »

```
class Cercle extends Forme {
    double rayon;

    Cercle(){ rayon=1; }
    Cercle(Point2D c,double l){ super(c);   rayon=l; }

    double surface() { return rayon * rayon; }

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

Méthode abstraite

Il peut donc, dans certains cas, être utile de définir une méthode sans en donner le code. Utilisation du mot clé abstract

```
abstract double surface();
```

au lieu de

```
double surface(){ return 0; }
```

Classe et méthode abstraites

Fichier « Forme.java"

```
abstract class Forme{
    Point2D centre;

    Forme(){ centre=new Point2D(); }
    Forme(Point2D c){ centre=c;  }
    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; }

    abstract double surface() ;

    void afficher(String nature){
        System.out.print("Objet " + nature+" :\n\tcentre : ");
    centre.afficher();    }
}
```

Faire usage de l'abstraction

On ne peut plus créer d'objet de la classe *Forme*

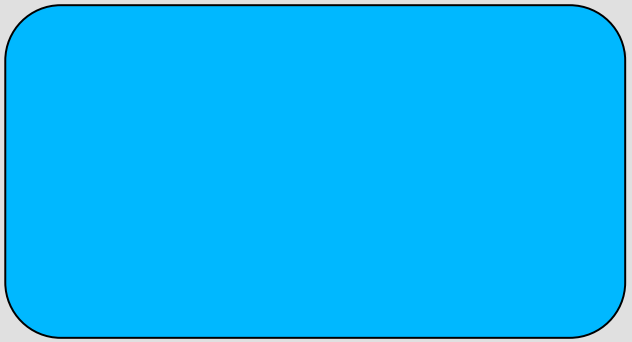
Attributs et méthodes constants

Attributs constants

```
class Math {
        final double PI = 3.141592654; // Attribut de classe
        void exemple(){
                final int value = 2; // Variable
        }
}
```

Ou encore

```
class Math {
        final double PI ;
        Math(double delta) {
                PI = 3.141592654 + delta;
        }
}
```



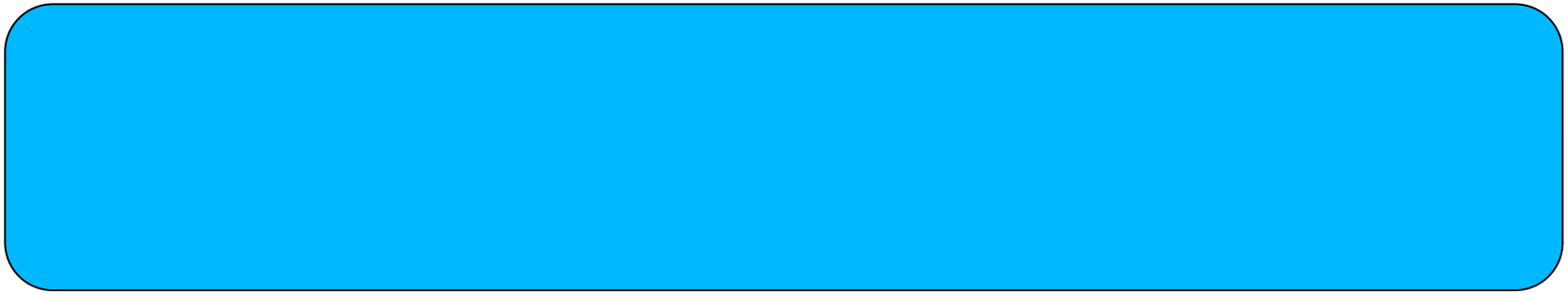
Objet Constant

```
class Math {
        double PI = 3.141592654;

        static public void main(String args[]){
                final Math m=new Math();
                m=new Math(); // n'est plus autorisé
                m.PI = 6.28; // est autorisé
        }
}
```

Méthodes finales

lorsque vous déclarez, dans une classe donnée, une méthode finale, il vous sera alors impossible de la redéfinir dans les sous-classes.



Paramètres constants

```
class ClasseQueconque {

        void methodeQuelconque(final int x){

                // On affiche la valeur du paramètre
                System.out.print(x);

                // Ceci marche très bien
                int y = 2 * x;

                // Ceci ne marche pas du tout !!!
                x += 2;
        }
}
```

Classes finales

Se sont des classes dont on **ne peut pas hériter**, qui **ne peuvent pas avoir de sous-classes**.

De nombreuses classes de la bibliothèque standard Java sont déclarées comme `final` telles que :

```
java.lang.System
java.lang.String,
java.lang.Integer,
etc.
```