- Classe, Objet
- Variables locales
- Niveaux de visibilité (Encapsulation)
- Attributs et Méthodes Statiques
- Surcharge des Méthodes
- Transmission des Paramètres
- Paquetage



- Une classe est une sorte de type de données définie par le programmeur
*Variables peuvent être de type classe*

- Objet est la valeur d'une variable type classe; une instanciation de la classe

- Classe définit les types de données qu'un objet peut contenir et les actions qu'ils peuvent entreprendre

**ClasseNom classeVar;**
**classeVar =new ClasseName();**
**ClasseNom classeVar=new ClasseName();**

```
class ClasseNom{
public String  instanceVar1;
public int  instanceVar2;

....

public void maMethode() {
.....}
```

Référencer la variable en dehors de la classe

```
objetNom.instanceVar1
objetNom.instanceVar2
```

Invoquer la méthode en dehors de la classe

```
[returntype =]objetNom.maMethode();
```

*La classe doit être définie dans un fichier dont le nom est le même que la classe suivi de .java*



- A programme java est une classe qui contient la méthode main
- L'exécution du programme java consiste à invoquer la méthode main

```
public static void main(String[] args)
```

- A méthode qui retourne une valeur doit contenir au moins une

```
return Expression
```

- Invoquer une méthode qui retourne une valeur

```
typeReturne tRVariable;
tRVariable = objetNom.methodeName();
```

- Invoquer une méthode `void`

```
objetNom.methodeNom();
```

*Il est possible d'invoquer une méthode qui retourne une valeur pour effectuer une action*

```
objetNom.returnedValueMethode();
```

•Une variable définie dans une méthode est locale
  •toutes variables de main sont locales
  •toutes variables de méthodes sont locales
  •2 méthodes peuvent avoir des variables locales de même nom
  •Il n'y a pas de variables globales dans Java

•Une variable définie dans un bloc ({ }) est locale dans le bloc

•Une variable définie dans une boucle for est locale dans cette boucle

•Les paramètres d'une méthode sont considérés comme des variables locales de cette méthode

•Les variables d'un objet sont initialisés automatiquement
*boolean=false, les types primitives =0, types classe= null*
*Les variables locales ne sont pas automatiquement initialisées*



Modificateurs **public, private**
**public :** pas de restriction d'accès pour des méthodes ou des variables
**private :** seulement accessibles à l'intérieure de la classe

Un bon programmeur doit définir chaque chaque attribut (variable) d'instance comme private

Méthodes sont en générales public
*(possibilité de les invoquer en dehors de la classe)*

Méthodes sont private si elles sont utilisées à l'intérieure des autres méthodes de la classe

Méthodes pour accéder ou modifier les valeurs
des attributs privés des objets : (accesseurs) calss

```
class {MaClasse
private int x;
.....
public int get()
{return x;}
public void put (int a)
{x=a;}
}

class AutreClasse{
.....
MaClasse y=new MaClasse();
int a=y.get();
y.set(5);
```



Certains attributs peuvent être communs à tous les objets
de la classe et exister indépendamment à tout objet de la classe.

- on les déclare **static**
- on les appelle **attributs de classe** ou attributs **statiques**
- une seule copie de ce champ qq soit le nombre d'objets
- tous les objets de classe peuvent lire et changer
- on référence directement dans la classe, mais en dehors il faut préciser la classe

```
class Exemple
{ static int n=0;
public void aff()
{System.out.println(n)};
}
....
.....  Exemple.n  ……
```

Une variable statique doit être **private** sauf si c'est une constante

```
public static final int BIRTH_YEAR = 1954;
```

```
int year=MaClasse.BIRTH_YEAR;
```

Classe Math contient des méthodes mathématiques et constantes standards

Elle est définie dans java.lang package (importé par défaut)
Math.PI, Math.E
public static double sqrt (double arg)  Math.sqrt(4)
public static long round (double arg)
public static int abs(int arg)
public static float abs(float arg)



**Méthodes statiques** sont invoquées sur une classe indépendamment d'une initiation de la classe.

•déclarées à l'intérieur de classe (`static`)

```
•class MaClasse{
public static returnedType maMethode(parameters)
   { . . . }
.....}
```

•invoquées avec le nom de la classe

```
returnedValue = MaClasse.maMethode(arguments);
```

•une méthode statique
  •ne doit pas accéder à un attribut d'instance (variable d'un objet)
  •peut accéder à une variable statique

*Il est possible d'ajouter la méthode main dans une classe quelconque*

```
class Point
{ private static int origine;  //abscisse absolu de
l'origine
   private int x;           //abscisse absolu du point
   public Point (int xx) { x=xx;}
   public void affiche ()
   { System.out.print("abscisse ="+(x-origine));
     System.out.println(" relative a l'origine="+origine);
   }
   public static void setorigine(int o) {origine=o;}
   public static int getorigine(int o) {return origine;}
}
public class TstOrig
{ public static void main(String[] args)
      {Point a=new Point(10);   a.affiche();
        Point.setorigine(3); a.affiche();
      }
}
```



La méthode Max retourne le point le plus éloigné de l'origine

avec une méthode static:

```
public static Point max (Point a, Point b)
{if (Math.abs ( a.x - origine) > Math.abs ( b.x -
origine)  return a;
      else return b;
}
```

***appel***

```
Point p1=new Point(4); Point p2= new Point(9);
Point. setorigine(5);
Point p= Point.max(p1,p2);
p.affiche();
```

avec une méthode usuelle:

```
public  Point max (Point a)
{if (Math.abs ( this.x - origine) > Math.abs ( a.x -
origine)  return this;
      else return a;
}
```

***appel***

```
Point p1=new Point(4); Point p2= new Point(9);
Point. setorigine(5);
Point p= p1.max(p2);
p.affiche();
```



Des méthodes dites **surchargées,** si elles portent le même nom
et que leurs paramètres sont de type différents,
et/ou que le nombre de paramètres est différent.

Il n'est pas possible d'avoir différents types pour la valeur retournée

```
public Personne(String n, int a)
{       nom=n;
        annee_n=a;
        salaire=1200;
}  //méthode  constructeur

public Personne(Personne p)
{       nom=p.nom;
        annee_n=p.annee_n;
        salaire=p.salaire;
}  //méthode  constructeur
```

```
public Personne(String n, int a, int s)
{       nom=n;
        annee_n=a;
        salaire=s;
}  //méthode  constructeur

public void affiche()
{
System.out.println(nom+"      "  + annee_n+"  "+salaire);
}

public void affiche(String tt)
{
System.out.println(tt +"     " + nom+"    "+  annee_n+"
"+salaire);
}
```



```
public class Person1
{
public static void main(String args[])
        { Personne p1= new Personne("dupont ",1961,1700);
        Personne p2= new Personne("bernard",1981);
        Personne p3= new Personne(p1);

        p1.affiche();
        p2.affiche();
        p3.affiche(" duplicata   " );
        p1.calcul_age();
        p2.calcul_age();
        p1.calcul_age();
        }
}
```

## Transmission des paramètres

•Les paramètres des méthodes sont passés par valeur

•Lorsque le paramètre est un objet c'est la référence d'objet qui est passé mais pas l'objet lui-même.

•On peut modifier l'objet dans la méthode
càd modifier les valeurs de ses attributs(variables)

•On ne peut pas modifier la référence d'objet



```
import java.io.*;
class Maclasse
{ private int x;
     public Maclasse(int xx)    {x=xx;}
     public int getx (){    return x; }
     public void putx (int a){x=a;}
}
class Operations
{public static void ajout(int n, int p) {n=n+p;}
 public static void ajout(Maclasse c, int p) {c.putx(c.getx()+p);}
 public static void ajout(Maclasse c){c.putx(c.getx()+5); c=null;}
}
public class TstTrans
{ public static void main(String[] args)
  { Maclasse a=new Maclasse(2);  int n=2;
  System.out.println("valeur d'objet avant "+a.getx());
  Operations.ajout(a,5);
  System.out.println("valeur d'objet apres "+a.getx());
  Operations.ajout(a);
  System.out.println("valeur d'objet  apres "+a.getx());
  System.out.println("valeur de n avant "+n);
  Operations.ajout(n,5);
  System.out.println("valeur d'objet  apres "+n);
}}
```

- Un **paquetage** est une collection de classes et d'interfaces

- Préciser en première ligne du fichier
  package nom_du_paquetage;
- Le nom du paquetage est séparé par des points
  -si la classe Maclasse est dans le paquetage p1.p2
  *dans le fichier source:*
  package p1.p2;
  *le chemin d'accès:* p1/p2/Maclasse.class
  -si on veut importer dans un autre fichier source
  import p1.p2.Maclasse
  import p1.p2.* (toutes les classes du paquetage)

*javac -d dir_dest program.java (*.class sont dans dir_dest/p1/p2)*
*java -classpath dir_path p1.p2.program (peu importe le catalogue courant)*



- En l'absence d'instruction package
  les classes appartiennent au paquetage par défaut (catalogue courant)

**NIVEAUX de VISIBILITE (Encapsulation)**

3 types d'accès à un attribut ou à une méthode

- **public** : (public) par tous les objets
- **private** : (privé) depuis sa propre classe
- **protected** (protégé) par sa classe et celles du paquetage

- Par défaut: protected