Université de Béjaia
Faculté des sciences exactes
Département d'informatique

2ème Année Licence Informatique
Semestre : 04

Module : Programmation Orientée Objet

Cours élaboré par H. EL BOUHISSI

Concepts généraux de l'OO

Cours élaboré par H. EL BOUHISSI

Principe de la POO

- Programmation par Objets
  - Unité logique : l'objet
  - Objet est défini par
    - un état
    - un comportement
    - une identité

maVoiture
- couleur = bleue
- vitesse = 100

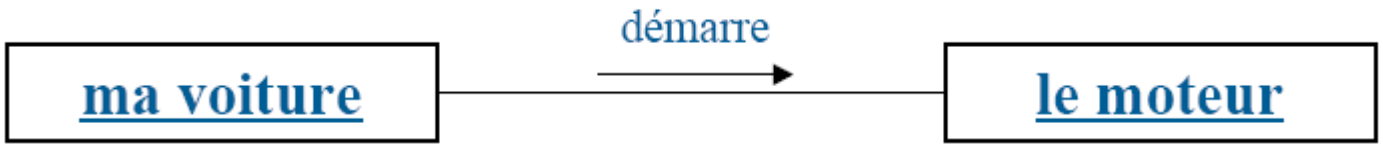
- État : représenté par des attributs (variables) qui stockent des valeurs
- Comportement : défini par des méthodes (procédures) qui modifient des états
- Identité : permet de distinguer un objet d'un autre objet

Principe de la POO

- Les objets communiquent entre eux par des messages
  - Un objet peut recevoir un message qui déclenche :
    - une méthode qui modifie son état

    et / ou

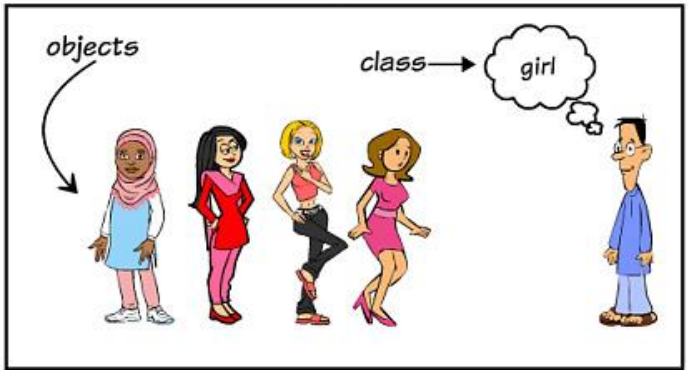
    - une méthode qui envoie un message à un autre objet

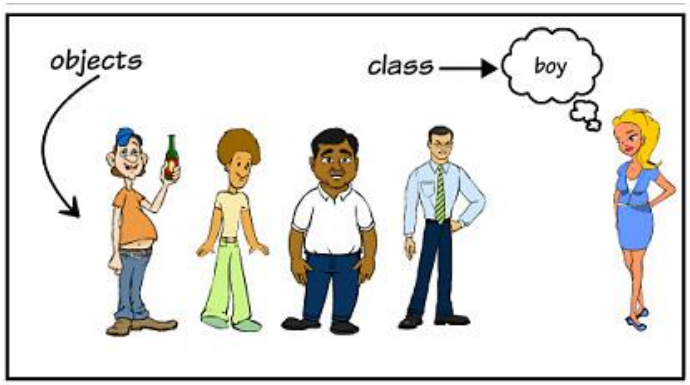


1- Notion de classe

Une classe encapsule, c'est-à-dire regroupe des propriétés et des comportements. Par exemple, la classe Humain *définit* des propriétés (deux bras, deux jambes...) et des comportements (marcher, parler, voir...).

Les comportements sont appelés *méthodes* et les propriétés *variables d'instance*.





Humain
Deux bras Deux jambes
Marcher() Manger()

```
class girl {
}

class boy {
}
```

Un objet est une instance de classe, un exemplaire utilisable crée à partir de cette classe et en valorisant certaines propriétés. Par exemple, *bob* ou *bill* sont des instances de la classe Humain, c'est-à-dire des humains ayant des propriétés spécifiques (*bob* est un humain aux yeux noirs et *bill* un humain aux yeux marrons).

2- Attributs/Propriétés

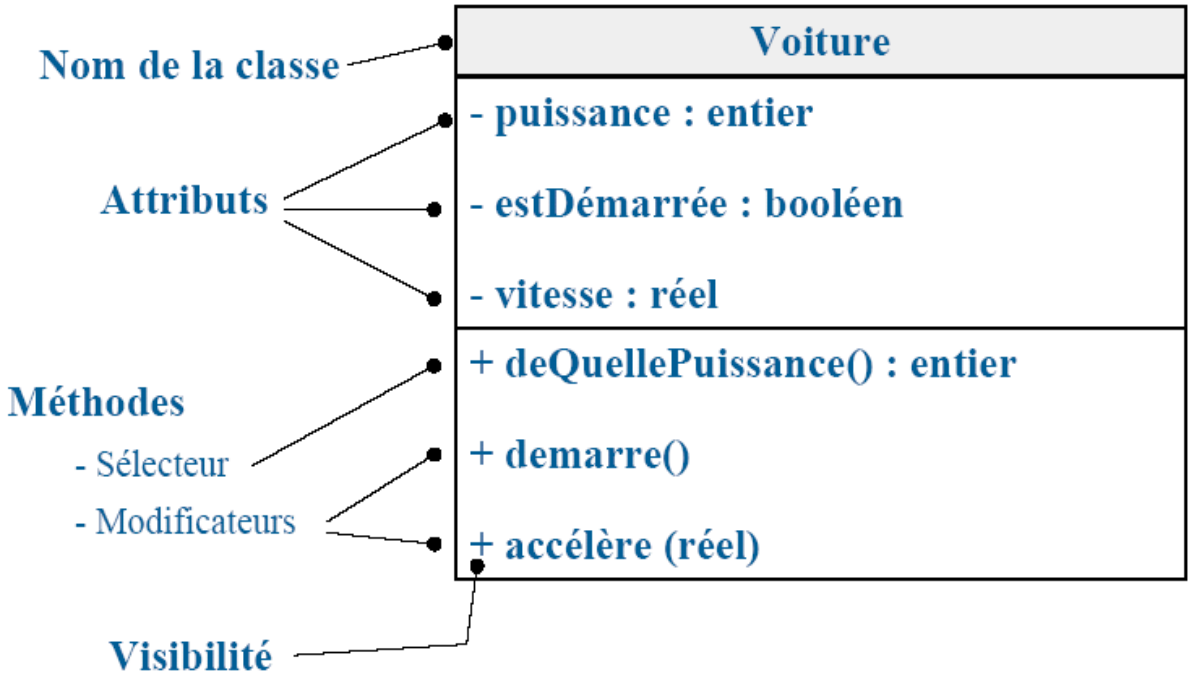
Chaque fille et chaque garçon ont des attributs. Ces attributs s'appellent souvent propriétés. Le sens précis de ces termes dépend souvent de quel système/langue/univers nous parlons.

Une propriété est utilisée pour décrire le modèle. Dans la classe exemple, nous utilisons les propriétés eyecolor et name.

Les attributs sont des variables de la classe et sont accessibles dans toutes les méthodes de la classe.

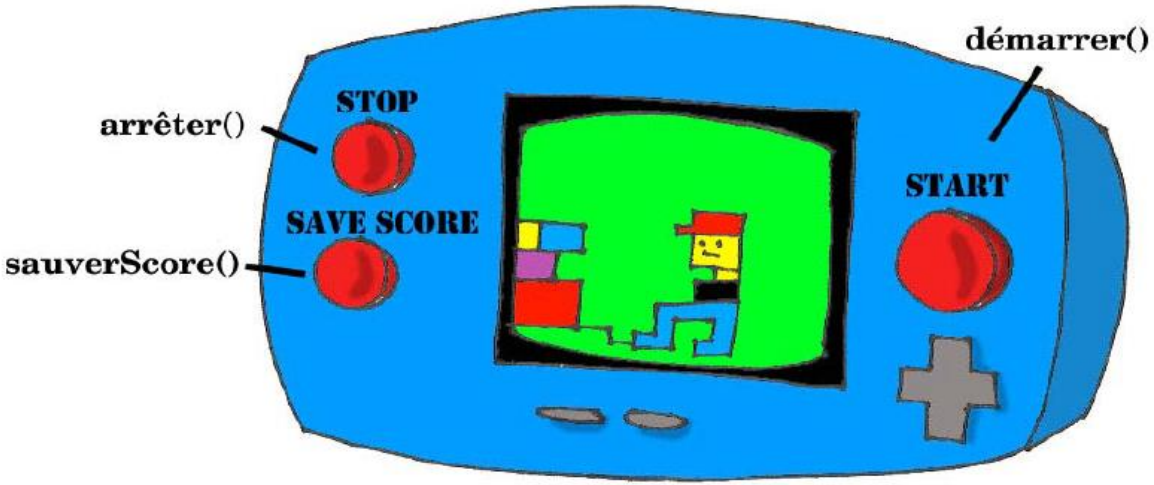
La « naissance » de notre objet fille et garçon est appelée instanciation. L'objet lui-même peut être aussi appelé instance.

3- Classe : Notations

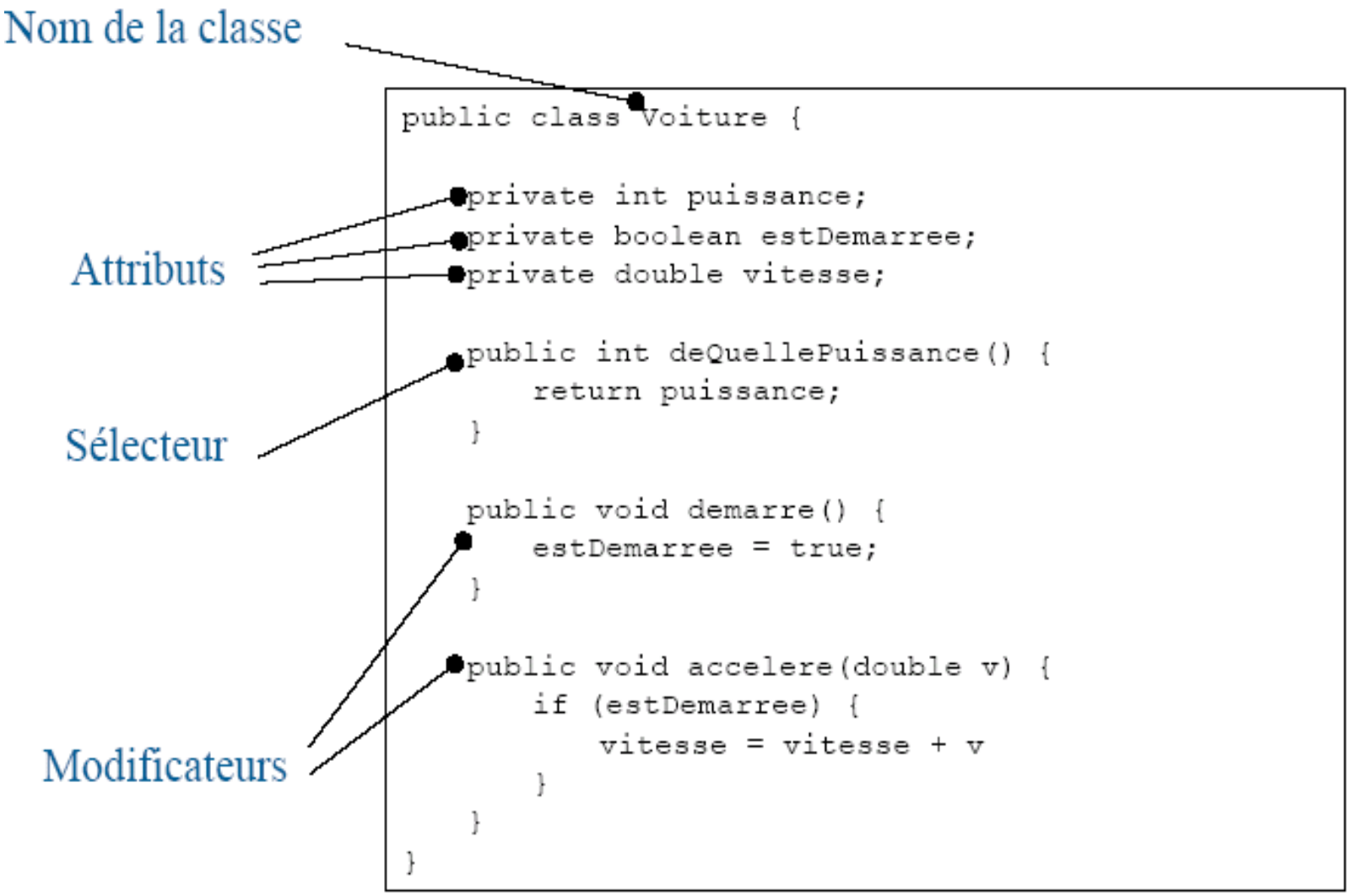



```
class girl {
  //properties
   public eyecolor;
   public name;
}

class boy {
   //properties
   public eyecolor;
   public name;
}
```



4- Classe : Exemple



5- Instantiation

On ne peut pas déclarer un objet comme on déclare une variable simple. Un objet en Java doit toujours être créé dynamiquement avec l'opérateur new. Il faut d'abord déclarer une référence à un objet du type voulu puis créer (construire) l'objet grâce à l'opérateur **new**

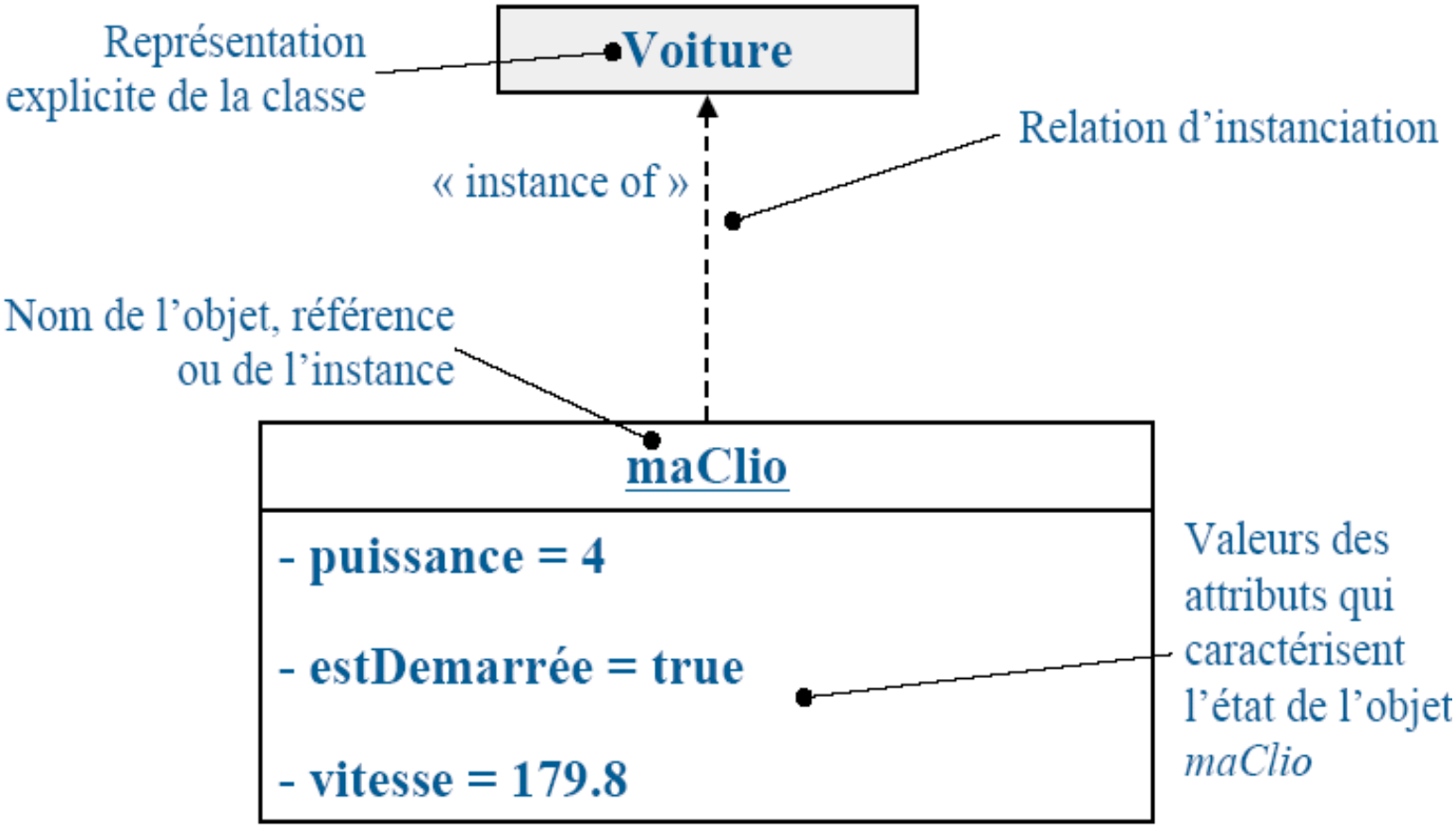
Soit une classe Personne :
Voilà comment créer un objet toto de type Personne.

Personne toto; //toto est une référence sur un objet de classe Personne
toto = new Personne( ); //un nouvel objet de type Personne est créé
//et l'adresse du nouvel objet est affectée à toto

On pourrait simplifier l'écriture en une seule ligne :
Personne toto = new Personne( );

5- Instantiation : Exemple

- maClio est une instance de la classe *Voiture*



6- Objet

- Un objet est **instance** d'une seule classe :
  - Se conforme à la description que celle-ci fournit
  - Admet une valeur propre à l'objet pour chaque attribut déclaré dans la classe
  - Les valeurs des attributs caractérisent l'**état** de l'objet
  - Possibilité de lui appliquer toute opération (**méthode**) définie dans la classe

- Tout objet est manipulé et identifié par sa référence
  - Utilisation de pointeur caché (plus accessible que le C++)
  - On parle indifféremment d'**instance**, de **référence** ou d'**objet**

6- Objet : Cycle de vie

- Création
  - Usage d'un Constructeur
  - L'objet est créé en mémoire et les attributs de l'objet sont initialisés
- Utilisation
  - Usage des Méthodes et des Attributs (non recommandé)
  - Les attributs de l'objet peuvent être modifiés
  - Les attributs (ou leurs dérivés) peuvent être consultés

L'utilisation d'un objet non construit provoque une exception de type *NullPointerException*

- Destruction et libération de la mémoire lorsque :

7- Constructeur

Chaque classe doit définir une ou plusieurs méthodes particulières appelées des constructeurs. Un constructeur est une méthode invoquée lors de la création d'un objet. Cette méthode, qui peut être vide, effectue les opérations nécessaires à l'initialisation d'un objet. Chaque constructeur doit avoir le même nom que la classe où il est défini et n'a aucune valeur de retour (c'est l'objet créé qui est renvoyé).

```
class Rectangle {
    ...
    Rectangle(int lon, int lar) {
        this.longueur = lon;
        this.largeur = lar;
        this.origine_x = 0;
        this.origine_y = 0;
    }
    ...
}
```

Plusieurs constructeurs peuvent être définis s'ils acceptent des paramètres d'entrée différents.

7- Constructeur (Suite)

Pour instancier une classe, c'est-à-dire créer un objet à partir d'une classe, il s'agit d'utiliser l'opérateur *new*.

En réalité l'opérateur *new*, lorsqu'il est utilisé, fait appel à une méthode spéciale de la classe: le **constructeur**.

Le rôle du constructeur est de déclarer et de permettre d'initialiser les données membres de la classe, ainsi que de permettre différentes actions (définies par le concepteur de la classe) lors de l'instanciation.

Un constructeur se définit comme une méthode standard, mais ne renvoie aucune valeur.

7- Constructeur (Suite)

Ainsi, le constructeur d'un objet porte le même nom que la classe et ne possède aucune valeur de retour (même pas *void*).

- Un constructeur porte le même nom que la classe dans laquelle il est défini
- Un constructeur n'a pas de type de retour (même pas *void*)
- Un constructeur peut avoir des arguments
- La définition d'un constructeur n'est pas obligatoire lorsqu'il n'est pas nécessaire.

La définition de cette fonction membre spéciale n'est pas obligatoire (si vous ne souhaitez pas initialiser les données membres par exemple) dans la mesure où un *constructeur par défaut* (appelé parfois *constructeur sans argument*) est défini par le compilateur Java si la classe n'en possède pas.

8- Méthodes - Comportements

Les méthodes (aussi appelées fonctions par abus de langage) vont vous permettre d'effectuer des traitements généralement que vous effectuerez au moins deux fois dans le code (sinon la création d'une méthode ne vous permettra juste que d'éclaircir un peu votre code avec en contrepartie une petite perte de performances).

Une méthode se délimite comme une classe, c'est à dire par deux accolades. Vous êtes cependant obligé d'y ajouter des parenthèses, même si elles ne contiennent rien. Ce qu'on peut placer à l'intérieur des parenthèses s'appellent les paramètres. Il existe deux types de méthodes :

1. **Une méthode qui renvoie un résultat** : Le nom doit être précédé par le type de résultat et le corps de la méthode doit contenir le mot réservé « **return** ».
2. **Une méthode qui ne renvoie pas de résultat** : Le nom doit être précédé par le mot réservé « **void** ».

9- Package

La programmation Java consiste à créer des classes. Or, étant donné qu'un programme est généralement développé par une équipe de programmeurs, c'est-à-dire plusieurs personnes, le fait de concaténer (mettre bout à bout) des classes dans un fichier est loin d'être satisfaisant. C'est pour cette raison que Java propose l'utilisation de **packages** (comparables aux bibliothèques du langage C++/C)

Ainsi, un **package** est une unité (un fichier) regroupant des classes. Pour créer un tel package, il suffit de commencer le fichier source contenant les classes à regrouper par l'instruction *package* suivi du nom que l'on désire donner au package. dès lors, toutes les classes contenues dans le fichier feront partie du package

10- Droits d’accès

En Java, la déclaration d'une classe, d'une méthode ou d'un membre peut être précédée par un modificateur d'accès.
Un modificateur indique si les autres classes de l'application pourront accéder ou non à la classe/méthode.

Les modificateurs d'accès permettent aux programmeurs de modifier la visibilité des éléments selon leurs besoins. Ils se placent, par convention, avant le type de l'objet et peuvent s'appliquer aux classes, méthodes et attributs.

Le tableau ci-dessous illustre les différents modificateurs qui existent en Java et où on peut les utiliser.

10- Droits d'accès (Suite)

Modifier	Signification
final	•Classe: ne peut pas être dérivée •Méthode: ne peut pas la redéfinir. Toutes les méthodes d'une classe *final* sont *final.* •Variable: une fois initialisé, on ne peut pas la modifier.
static	•Méthode: peut être utilisée sans instancier la classe •Attribut: attribut de classe, sa valeur est partagée entre toutes les instances de la classe.

Modificateur static / méthode : Exemple

Une méthode statique est une méthode qui peut être appelée même sans avoir instancié la classe. Une méthode statique ne peut accéder qu'à des attributs et méthodes statiques.

```
public class Static_exemple {
        static int mymethod(int a, int b)
    {
    int c=a+b;
    return c;
    }
        public static void main(String [] args)

    {
  System.out.println(Static_exemple.mymethod(2,5));
    }
}
```

Static_exemple.java

Modificateur final (1)

1) Devant une méthode :

On indique que cette méthode ne pourra plus être redéfinie dans une classe fille. Ce qui entraîne une certaine optimisation dans les appels à cette méthode.

```
public class MaClass
{
    public final void uneMethode() { ... }
}
```

On peut s'en servir pour forcer le comportement d'une méthode dans les sous-classes.

2) Devant une classe :

```
public final class ClasseFinal
{ ... }
```

On ne peut pas créer de classe dérivée de celle-ci.

Modificateur final (2)

3) Devant une variable :

Si le mot clé final est utilisé dans la déclaration d'une variable, on ne peut affecter de valeur à cette variable qu'une fois, cette valeur ne pouvant plus être modifiée par la suite.

public final double PI = 3.14159; // Impossible de modifier la valeur

```
public class Final_exemple {

    public static final double i=2.5;
    public static void main(String args[])
    {

      i=3.5;  // interdit
    }
}
```

10- Droits d'accès (Suite)

Ces trois mots clefs du langage java définissent la **portée d'une variable, d'une méthode ou d'une classe**. Il existe en fait **quatre modificateurs** d'accessibilité. Le quatrième est le modificateur vide (rien, pas de modificateur). Il ne faut pas confondre ce dernier avec public.

Mot-clé	Portée	Remarques
public	Les variables, méthodes ou classes publiques sont accessibles par tout objet.	Il ne peut y avoir qu'une seule classe publique par .java et celle-ci doit obligatoirement porter le nom du fichier .java
"rien"	Les variables, méthodes ou classes définies sans modificateur sont accessibles par toute classe appartenant au même package.	Attention : les variables sans modificateur ne sont pas accessibles aux classes fille définies dans un autre package.
protected	Les variables, méthodes ou classes définies comme protégées ne sont accessibles que par les classes filles et classes du même package..	
private	Les variables, méthodes ou classes définies comme privées ne sont accessibles que par la classe dans laquelle elles sont définies.	Il est fortement conseillé de déclarer comme privés tous les attributs d'une classe, et de créer des méthodes de type get/set pour y accéder.

Le mot clé this

THIS en JAVA indique l'instance courante de l'objet de la classe. On peut l'utiliser comme on ne peut pas.

Elle est surtout utilisée dans les méthodes lorsque le nom du paramètre de la méthode est le même que le nom de la variable globale de la classe. Donc pour faire la différence on utilise **THIS**.

```
class MaClasse{
String nom;
public void setName(String nom){
    this.nom = nom;
/* le nom lié avec this, est celle qu'on a
déclaré au début de la classe
tandis que nom, le paramètre de la méthode
setName() est une variable locale.*/
} // fin méthode
/*....*/
} // fin classe
```