esprit
Se former autrement

# Conception Orientée Objet & Programmation JAVA

## Chapitre 2 : Classe et Objet**(Partie 1)**

# PLAN

- Introduction
- **Classe et objet**
- Encapsulation
- Héritage
- Polymorphisme
- Exceptions
- Interfaces
- Collection
- Interface Fonctionnelle
- Expression Lambda
- Stream

# Objectifs du chapitre

- ✔ Notion de classe et d'objet
- ✔ Déclaration de classe
- ✔ Déclarations des attributs et des méthodes
- ✔ Les types des variables (primitives et objets)
- ✔ Notion de référence
- ✔ Les constructeurs

# Approche procédurale vs OO

- **Etude de cas: Gestion des Elections en Tunisie**

## Raisonnement par Approche Procédurale

- ✔ Que fait le système ?
- ✔ Comment ajouter un candidat aux élections ?
- ✔ Comment gérer les votes de citoyens ?

## Raisonnement par Approche OO

- ✔ De quoi le système est composé?
- ✔ On doit décortiquer le Système en classes :
  - Candidat
  - Vote
  - Citoyen

# POO: Identification des classes

- les classes sont les composants fondamentaux de la POO
- C'est une unité de base de la programmation orientée objet et représente les entités de la vie réelle.

Candidat

Citoyen

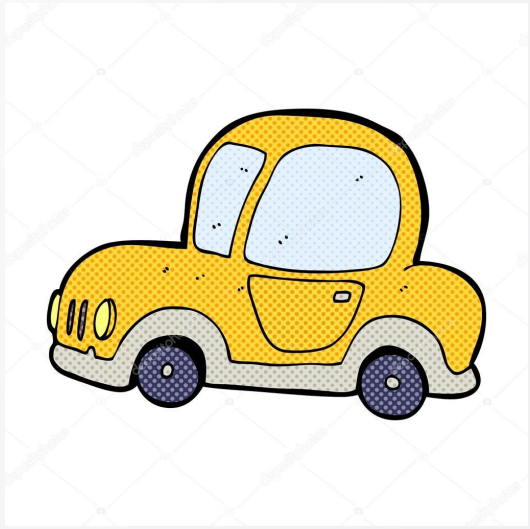
Vote

**Systeme de Gestion des Elections en Tunisie**

# Notion classe et d'objet

- Le concept d'utilisation de classes et d'objets consiste à encapsuler l'état et le comportement dans une seule unité de programmation.
- Les objets Java sont similaires aux objets du monde réel.
- Par exemple, nous pouvons créer une classe voiture en Java, qui aura des propriétés telles que la vitesse actuelle et l'immatriculation ; et un comportement comme: rouler et changerPneu….

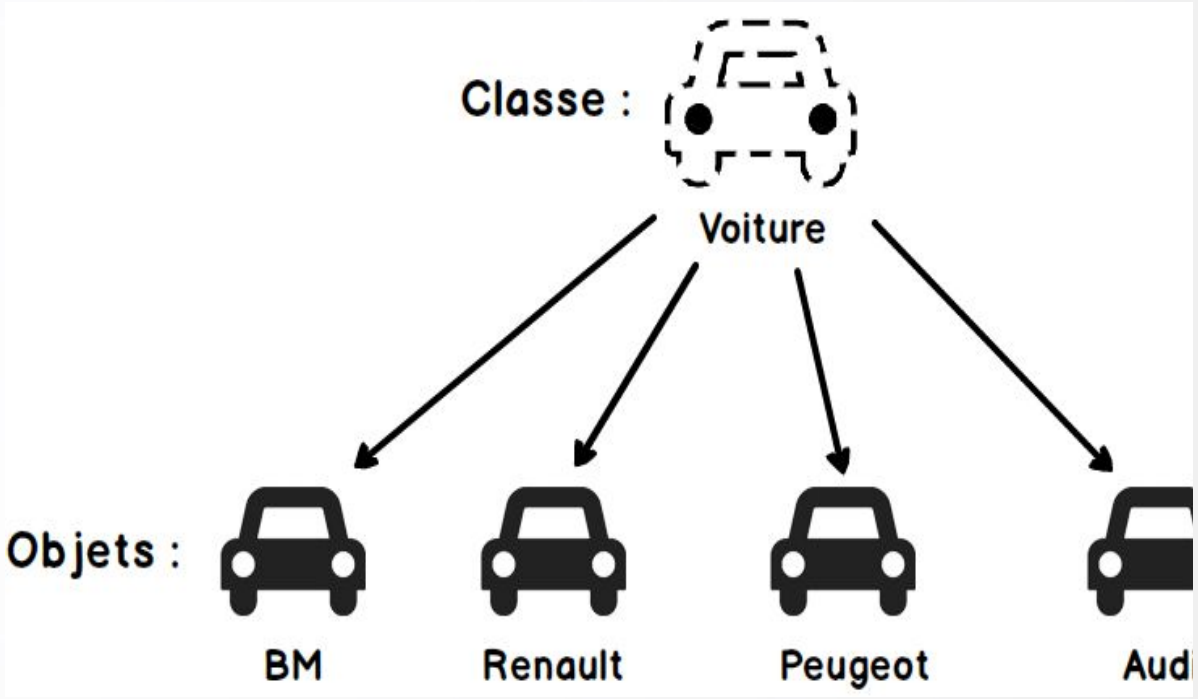
Voiture



# Notion classes et d'objet

- Qu'est ce qu'une classe?
  Une classe est un plan ou un prototype défini par l'utilisateur à partir duquel des objets sont instanciés.

  Il représente l'ensemble des propriétés ou méthodes communes à tous les objets d'un même type.



# Notion classe : Nom de la classe

Lorsque vous créez une classe java, vous devez suivre cette règle: le nom du fichier et le nom de la classe doivent être les mêmes.

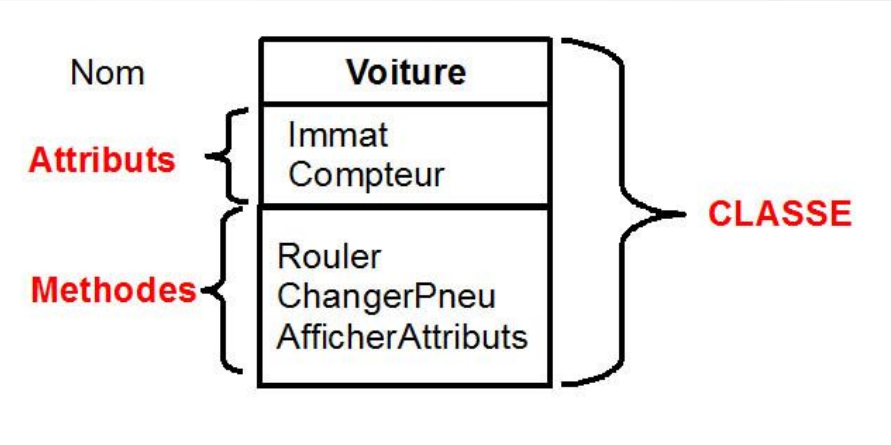
Voiture écrit avec une majuscule V n'est pas la même chose que voiture, écrit avec une minuscule v.

```java
public class Voiture {

    int vitesse;
    String model;

    public Voiture(String model) {
        this.model = model;
    }

    public void accelerer() {
        // ajoute 10 miles par heure à la vitesse actuelle
        vitesse = vitesse + 10;
    }

    public void freiner() {
        // déduire 10 miles par heure à la vitesse actuelle
        vitesse = vitesse - 10;
    }
}
```

# Notion classe : Noms de la classe

Un objet est donc « issu » d'une classe, c'est le produit qui sort d'un moule.
En réalité on dit qu'un objet est une instanciation d'une classe

objet = instance




Une classe est composée de deux parties :

- Les attributs (parfois appelés données membres) :
- il s'agit des données représentant l'état de l'objet

- Les méthodes (parfois appelées fonctions membres): il s'agit des opérations applicables aux objets

# Manipulation de variables

# Déclaration des variables

Quel nom choisir pour notre variable?

Les noms des variables sont case-sensitive

Les espaces ne sont pas permis

Le nom de la variable doit commencer par une lettre miniscule.

**Syntaxe:**

type nom_variable =valeur;

**int id = 0;**

# Mots clés JAVA



Évitez les mots réservés de java tel que :

| | | | |
|---|---|---|---|
| abstract | double | int | strictfp ** |
| boolean | else | interface | super |
| break | extends | long | switch |
| byte | final | native | synchronized |
| case | finally | new | this |
| catch | float | package | throw |
| char | for | private | throws |
| class | goto * | protected | transient |
| const * | if | public | try |
| continue | implements | return | void |
| default | import | short | volatile |
| do | instanceof | static | while |

* Indique un mot clé qui est peu utilisé

** A partir de la plate-forme Java2

# Déclaration des variables

- **Constante**

  - On déclare une constante avec le mot final

    Exemple:

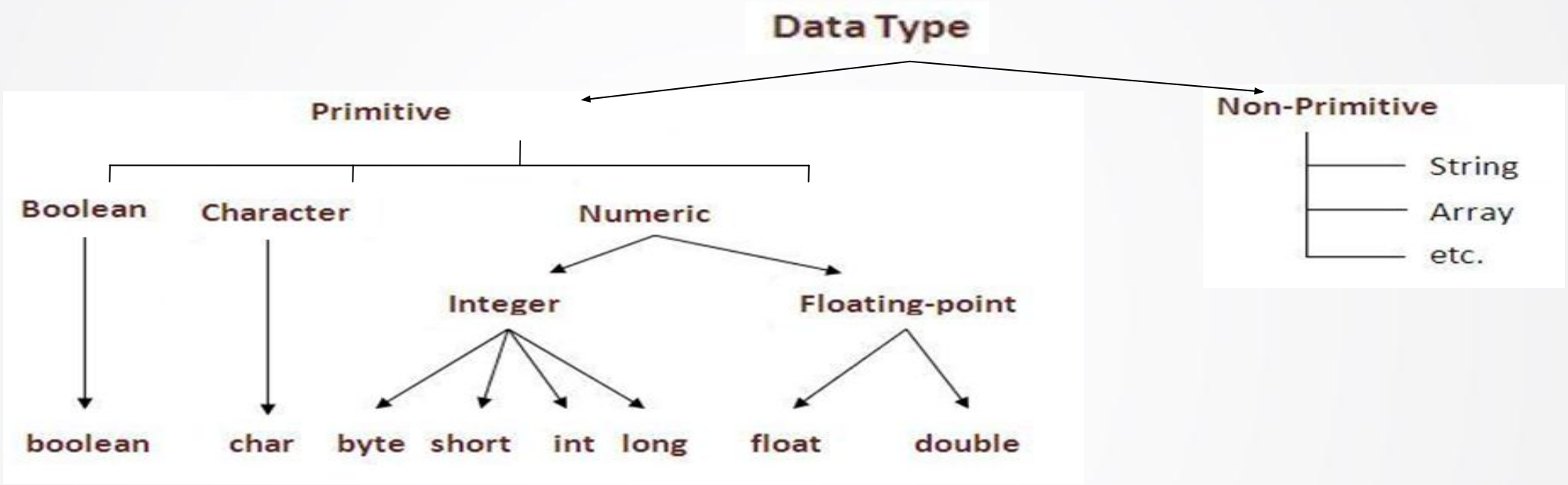
    final int **CAPACITE**=2000;

  - Le nom de la constante doit être en majuscule.
  - Si le nom est composé de plusieurs mots, on utilise _ pour la séparation des mots

Exemple:

final int TAILLE;
final int MAX_STOCK;

# Déclaration des variables




Il existe deux sortes de variables :

- Une variable primitive contient des bits qui représentent sa valeur.
  Exemple : int, foat , boolean.

- Une variable référence contient des bits qui indiquent comment accéder à l'objet.

# Déclaration des variables

## Les variables primitives

- Déclaration d'une variable primitive

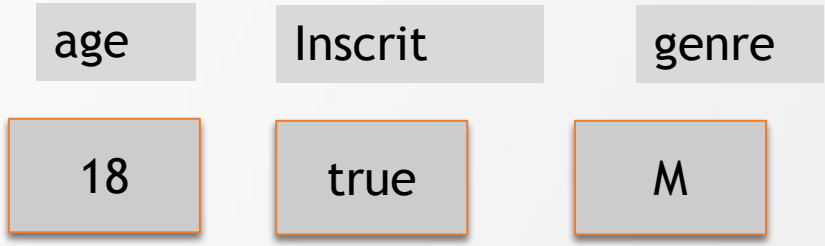
```
int age;
boolean inscrit;
char genre;
```

Réservation de l'espace mémoire.


- Affectation de valeur

```
age=18;
inscrit=true;
genre='M';
```

- Variable sera stockée dans l'espace mémoire réservé.



# Déclaration des variables

## Les types entiers

| | Valeur minimale | Valeur maximale | Codé sur |
|---|---|---|---|
| **byte** | - 128 | 127 | 8 bits |
| **short** | - 32 768 | 32767 | 16 bits |
| **int** | - 2147 483 648 | 2 147 483 647 | 32 bits |
| **long** | -9223372036854775808 | 9223372036854775807 | 64 bits |

## Les types réels

| | Valeur minimale | Valeur maximale | Codé sur |
|---|---|---|---|
| **float** | 1.4E45 | 3.4028235E38 | 4 octets |
| **double** | 4.9E324 | 1.79769313486231570E308 | 8 octets |

# Déclaration des variables

**Le type caractère char**

```
char sexe='M';
```

**Le type boolean**

```
boolean Inscrit=true;
```

**Le type chaine de caractère String**

```
String message = ''Hello Word'';
```

# Déclaration des variables

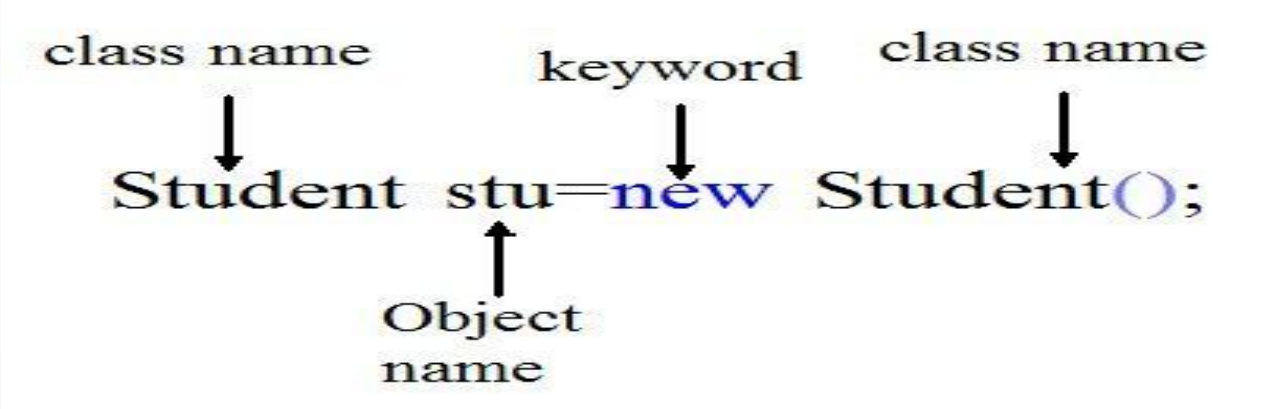
Les valeurs par défaut des variables

| Type | Valeur par défaut |
|---|---|
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| | |
| boolean | false |

# Déclaration d’une classe et Manipulation des constructeurs

# Instanciation de Classe

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 constructeur

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

# Le constructeur

- 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

# Le constructeur

- Constructeur par défaut

```
Candidat(){}
```

```
Candidat(){
      id=0;
      nom='BA'
      NBVote=10.2f;
          }
```

- Constructeur surchargé

```
    Candidat(int id, String couleur, float vote){
          this.id=id;
          this.nom=nom;
          this.vote=vote;
    }
```

- Le constructeur par défaut initialise les variables de la classe aux valeurs par défaut.

• Si vous ne créez pas un constructeur dans votre classe, le compilateur va automatiquement vous créer un constructeur par défaut implicite

• Si le constructeur surchargé est créé, le constructeur par défaut implicite ne sera plus créer par le compilateur

• La plateforme java différencie entre les différents constructeurs déclarés au sein d'une même classe en se basant sur le nombre des paramètres et leurs types.

On ne peut pas créer deux constructeurs ayant le même nombre et types des paramètres.

# Le constructeur

Quel constructeur va être déterminé lorsque vous allez créer votre objet ?

```
class Chemise{
        int id;
        char couleur;
        float prix;
        String description;
        int quantite;

        Chemise () {}

        Chemise(int id) {
                this.id=id;
        }

        Chemise(int id, char couleur) {
                this.couleur=couleur;
        }
}
```

**Utilisation:**

Chemise ch1=new Chemise();

Chemise ch2=new Chemise(122);

Chemise ch3=new Chemise(122, 'B');

# Utilisation de This

- Le mot-clé *this* permet de désigner l'objet courant,
- Pour manipuler un attribut de l'objet courant:

this.couleur

- Pour manipuler une méthode de la classe courante :

this.ajouterChemise (100)

- Pour faire appel au constructeur de l'objet courant: this()

```
class Chemise{
        int id;
        char couleur;

        Chemise(int id) {
                this.id=id;
        }

        Chemise(int id, char couleur) {
                this.couleur=couleur;
        }
}
```

# Manipulation de Méthodes

# Déclaration des méthodes

- Le nom de la méthode doit commencer par un verbe

  - Une méthode est une fonction faisant partie d'une classe.

  - Elle permet d'effectuer des traitements sur (ou avec) les données membres des objets.

**Syntaxe:**

```
Niveau d’accès Type_retour nom_method([arguments])
{

}
```

# Appel des méthodes

Pour exécuter une méthode, il suffit de faire appel à elle en écrivant l'objet auquel elle s'applique (celui qui contient les données), le nom de la méthode ( en respectant la casse), suivie de ses arguments entre parenthèse :

```
objet.nomDeLaMethode(argument1,argument2);
```

Le passage d'arguments à une méthode se fait au moyen d'une liste d'arguments (séparés par des virgules) entre parenthèses suivant immédiatement le nom de la méthode.

Le nombre et le type d'arguments dans la déclaration, le prototype et dans l'appel doit correspondre, sinon, on risque de générer une erreur lors de la compilation...

# Merci pour votre attention