

# Chapitre 3: Classes et objets en java

- ☐ Rappel
- ☐ Représentation graphique UML
- ☐ Déclaration de classe
- ☐ Stockage des classes
- ☐ Syntaxe de définition d'une classe
- ☐ Variable de classe et variable d'instance
- ☐ Méthodes
- ☐ passage des paramètres

## □ Un objet

- maintient son **état** dans des variables
- implémente son **comportement** à l'aide de **méthodes**

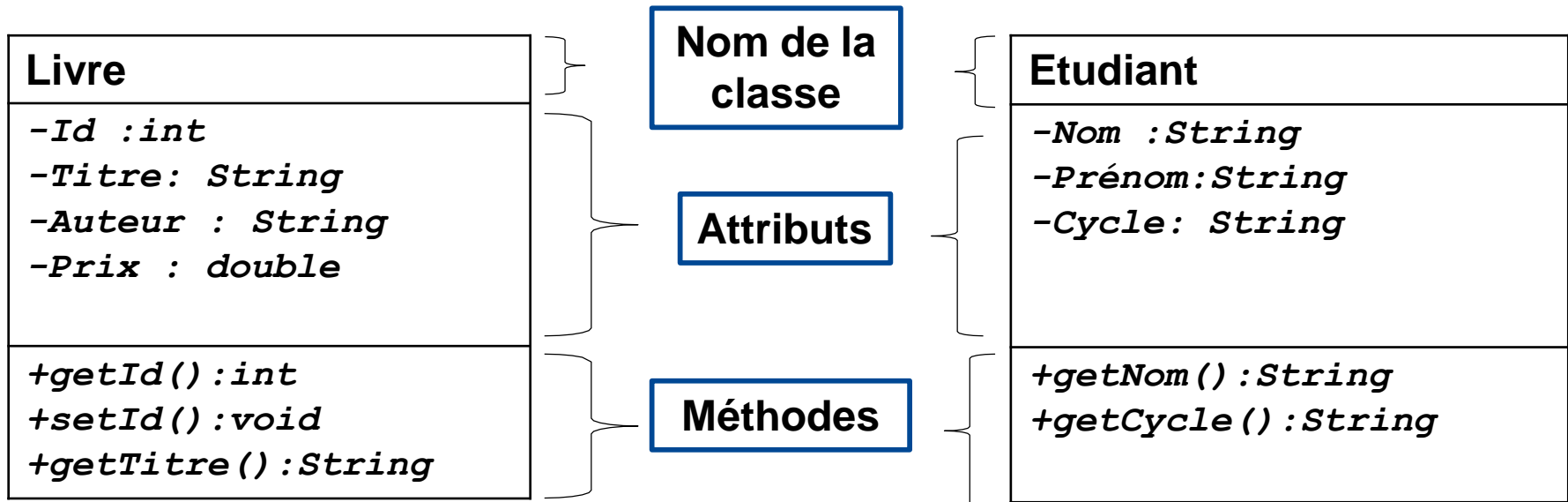
## □ Une Classe est une description d'une famille d'objets ayant une même structure et un même comportement. Elle est caractérisée par :

- Un **nom**
- Des **attributs** nommés ayant une valeur pour chaque objet de la classe. Ils caractérisent l'état des objets pendant l'exécution du programme
- Des **méthodes** représentant le comportement des objets de cette classe. Elles manipulent les champs des objets et caractérisent les actions pouvant être effectuées par les objets.

# Représentation graphique UML(1/2)

□ Une classe se décrit par trois compartiments :

- Nom
- Attributs
- Méthodes



# Représentation graphique UML(2/2)

## Diagramme de classe

### Livre

*-Id :int*  
*-Titre :String*  
*-Auteur :String*  
*-Prix :double*

*+getId() :int*  
*+setId() :void*  
*+getTitre() :String*

### Livre : Livre 1

*Id=120938*  
*Titre=Apprenez à*  
*programmer en Java*  
*Auteur=Cyrille Herby*  
*Prix= 38.00 €*

### Livre : Livre 2

*Id=1788763*  
*Titre=La*  
*programmation*  
*orientée objet en*  
*java*  
*Auteur=Huges Bergini*  
*Prix=25 €*

*Représentation UML*  
*des objets*

# Déclaration d'une classe

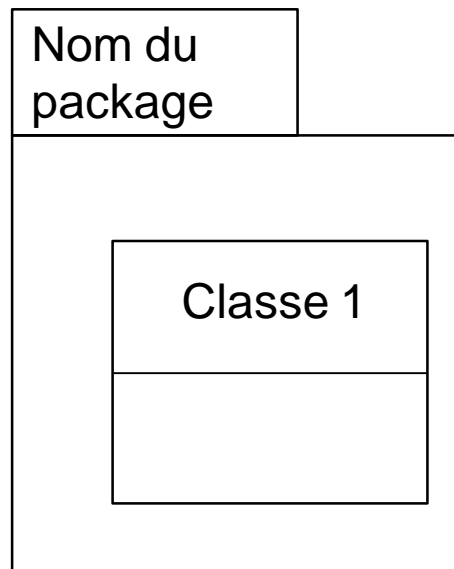
- ☐ Le nom de la classe est spécifié derrière le mot clé « **class** »
- ☐ Le **corps** de la classe est délimité par des accolades **{ }**
- ☐ On définit dans le corps les **attributs** et les **méthodes** qui constituent la classe

```
Modificateurs class Nomclasse{  
  
    < corps de la classe >  
  
}
```

- ☐ Les modificateurs pour les classes peuvent être: public, final, abstract
- ☐ public : classe visible à tout le monde
- ☐ final : classe qui ne pas être héritée
- ☐ abstract : classe qui ne peut pas être instanciée
- ☐ **(les modificateurs vont être détaillés dans le chapitre 4)**

# Stockage des classes

- ❑ les classes sont stockées dans des packages
- ❑ les packages offrent un mécanisme général pour la partition des modèles et le regroupement des éléments de la modélisation
- ❑ Chaque package est représenté graphiquement par un dossier



# Syntaxe de définition d'une classe

- Exemple : Une classe définissant un **Livre**

```
package bibliothèque;  
public Class Livre  
{
```

Nom du package

Nom de la Classe

```
    int id;  
    String titre;  
    String auteur;  
    double prix;
```

Attributs de Classe

```
    public int getId() {  
        return id;  
    }  
    public void setId(int id) {  
        this.id = id;  
    }  
    public String getTitre() {  
        return titre;  
    }  
}
```

Méthodes de Classe



# Variable de classe et variable d'instance(1/2)

- ❑ Variable de classe (variable statique)
- ❑ Le nom de la variable est précédé du mot clé **static**
  - Exemple :
  - **static int i;**
- ❑ N'appartiennent pas à une instance particulière, elles appartiennent à la classe.
- ❑ Elle peut être appelée même sans avoir instancié la classe
  - Exemple :
  - `Nomdeclasse.i ; // Correct`
  - `objet.i ; // Correct`
- ❑ Existe dès que sa classe est chargée, indépendamment de toute instantiation.
- ❑ chaque instance de la classe partage la même variable (la même copie)

# Variable de classe et variable d'instance(2/2)

- ❑ Variable d'instance :
- ❑ Le nom de la variable n'est pas précédé par un mot clé spécial
  - Exemple :
    - ❑ `int i;`
- ❑ Variable d'instance sont associées à des objets (instance)
- ❑ Elle peut être appelée avec la notation objet.variable
  - Exemple :
    - `Nomdeclasse.i` // impossible Incorrect
    - `objet.i` correct
- ❑ Chaque instance possède sa propre copie de la variable

# Exemple

```
Package module ;
public class MaClasse{
// Variable d'instance
    int value;
    // Variable de classe partagée entre toutes
    les instances
    static int nbr = 5;
    public static void main(String args[]){
        //Variable d'instance
        MaClasse obj = new MaClasse();
        obj.value = 6;
        System.out.println("Variable d'instance=
"+obj.value);
        System.out.println("Variable de classe=
"+MaClasse.nbr);
    }
}
```

**Résultat :**  
**Variable d'instance= 6**  
**Variable de classe= 5**

# Méthodes et paramètres (1/2)

- La notion de méthodes dans les langages objets :
  - Proches de la notion de procédure ou de fonction dans les langages procéduraux.
  - La méthode c'est avant tout le regroupement d'un ***ensemble d'instructions***
  - Comme pour les procédures ou les fonctions (au sens mathématiques) on peut passer des paramètres aux méthodes et ces dernières peuvent renvoyer des valeurs (grâce au mot clé **return**).

# Méthodes et de paramètres (2/2)

exemple : public,  
Static, private

type de la valeur  
renvoyée ou void

couples d'un type et d'un  
identificateur séparés par des « , »

**<modificateur>** **<type-retour>** **<nomMéthode>** (**<liste-param>**)  
{**<bloc>**}

**public double** add (**double** number1, **double** number2)  
{  
    return (number1 + number2);  
}

Notre méthode  
retourne ici une  
valeur

# Passage des paramètres(1/3)

- ❑ Lors des appels de méthode, les arguments sont toujours passés par valeur
- ❑ Dans le cas des types primitifs:
  - C'est la valeur de l'argument qui est recopiée dans le paramètre de la méthode
- ❑ Dans le cas des type référence (objet) :
  - C'est la référence qui est passée par valeur **le contenu de l'objet peut être modifié, mais pas la référence** elle-même.
  - En revanche, la modification de la référence elle-même est sans effet sur l'argument (c'en est une copie)

# Passage des paramètres(2/3)

## □ Types primitifs

- Exemple :

```
public class MaClasse{  
    public static void m1(int i) {  
        i++;  
    }  
    public static void main(String[] args) {  
        int entier = 5;  
        m1(entier);  
        System.out.println(entier); }  
    }  
}
```

**Résultat :**  
**5**

# Passage des paramètres(3/3)

## □ Types référencés

- Exemple :

```
package exemple
public class Maclasse {
    int x;
    public static void      test(Maclasse a)
    {
        a.x = 30; //on pointe vers l'emplacement en
        mémoire    puis on modifie l'attribut
    }
    public static void main(String[] args) {
        Maclasse a = new Maclasse();
        a.x = 10;
        test(a);
        System.out.pr
        intln(a.x) ;
    }
}
```

**Résultat :**  
**30**





**Fin partie 1**