

- ☐ Méthodes de classe et méthodes d'instance
- ☐ Surchage des méthodes
- ☐ L'instanciation
- ☐ Les constructeurs
- ☐ La création d'objet

- ❑ Une méthode de classe ou une méthode statique est une méthode qui n'agit pas sur des variables d'instance mais uniquement sur des variables de classe et ses propres variables.
- ❑ Elle peut être appelée même sans avoir instancié la classe avec la notation `Nomdeclasse.methode()` au lieu de `objet.methode()`
- ❑ Dans la déclaration, tout comme les variables de classe le nom de la méthode est précédé du mot clé **static**
- ❑ Exemple :
 - la méthode `main`
 - Toutes les méthodes mathématiques définies dans la classe `Math` en Java s'appellent de la manière suivante : `Math.sin(45)` ou `Math.pow(2,1)`

- ❑ Une méthode d'instance est une méthode qui agit sur des variables d'instance.
- ❑ Elle peut être appelée après avoir instancié un objet de la classe
- ❑ Dans la déclaration, le nom de la méthode n'est précédé par aucun mot clé
- ❑ Elle peut être appelée avec la notation uniquement objet.methode()
- Exemple :

Méthode de classe et méthode d'instance(1/2)

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

```
    static int id;
```

```
    String titre;  
    String auteur;  
    double prix;
```

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

```
    public String getTitre() {  
        return titre;  
    }
```

```
}
```

Variables de classes

Variables d'instance

Méthodes de classe

Méthodes d'instance

Surcharge de méthodes (1/2)

- ❑ Dans une même classe, plusieurs méthodes peuvent posséder le même nom, pourvu qu'elles diffèrent en nombre et/ou type de paramètres.
- ❑ On parle de **surdéfinition** ou **surcharge**
- ❑ Le choix de la méthode à utiliser est en fonction des paramètres passés à l'appel.
 - Ce choix est réalisé de façon statique (c'est-à-dire à la compilation).
 - Très souvent les constructeurs sont surchargés (plusieurs constructeurs prenant des paramètres différents et initialisant de manières différentes les objets)

Surcharge de méthodes (2/2)

```
Class addition {  
    public double add (double number1, double number2)  
    {  
        return (number1 +number2);  
    }  
    public double<add (double number1, double number2,  
        double number3 )  
    {  
        return (number1 +number2+ number3);  
    }  
}
```

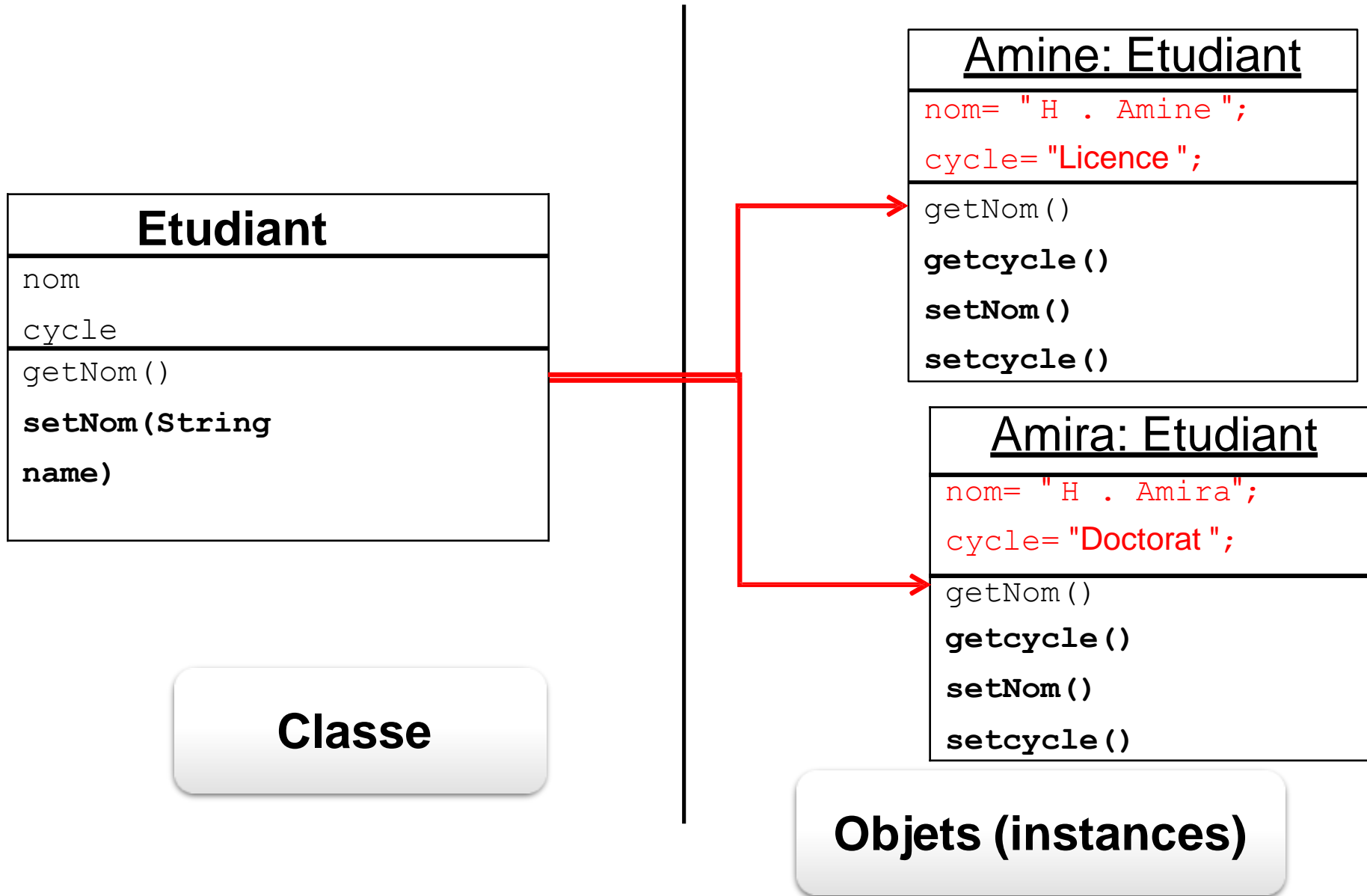
La méthode add est
surchargée

L'instanciation (1/2)

- ❑ **Instanciation** : concrétisation d'une classe en un objet « *concret* ».
- ❑ Dans nos programmes Java nous allons définir des classes et **instancier** ces classes en des objets qui vont interagir.
- ❑ Le fonctionnement du programme résultera de **l'interaction** entre ces **objets** « **instanciés** ».
- ❑ En Programmation Orientée Objet, on décrit des classes et l'application en elle-même va être constituée des objets instanciés, à partir de ces classes, qui vont communiquer et agir les uns sur les autres.

Instance = Objet

L'instanciation (2/2)



- ❑ Le constructeur est une méthode particulière :
 - Portant le même nom que la classe
 - Définie **sans aucun type de retour.**
- ❑ **Son but est :**
 - Initialiser les attributs d'un objet dès sa création.
- ❑ Est appelé uniquement lors de la construction de l'objet
- ❑ Toute classe possède **au moins** un constructeur. Si le programmeur ne l'écrit pas, il en existe un **par défaut, sans paramètres, de code vide.**
- ❑ Le constructeur est souvent une des méthodes les plus surchargées

Constructeurs(2/5)

```
public class Livre
{
    int id ; // identifiant du livre
    String auteur ; // auteur du livre
    String titre; // titre du livre;
    double prix;// prix du livre
    // constructeur par défaut (sans
    // paramètres)
    public livre(){

    }

    public int getId() {
        return id;
    }
}
```

Constructeur par défaut (code vide)

En l'absence de constructeur explicitement défini, le compilateur ajoute un constructeur public sans paramètre

Les constructeurs (3/5)

```
public class Livre
{
    int id ; // identifiant du
    livre
    String auteur ;// auteur du
    livre
    String titre; // titre du
    livre;
    double prix;// prix du livre
    // constructeur paramétré
    Livre (int idd,String
    aut,String tr, double p){
        id=idd;
        auteur=aut;
        titre=tr;
        prix=p;
    }
}
```

Va donner une erreur à la compilation

Va être exécuté sans erreur

Le constructeur par défaut (Livre()) n'existe plus.

```
public class TestLivre
{
    public static void main(String
    args[])
    {
        //appel au constructeur par
        défaut
        Livre L= new Livre();
        //Appel au constructeur
        paramétré
        Livre L= new
        Livre (12334,"auteur1","java",40.
        80)
    }
}
```

Les constructeurs (4/5)

- Pour une même classe, il peut y avoir **plusieurs** constructeurs, de paramètres différents (**surcharge**).

comment appeler un constructeur ?

- Est appelé avec le **new** auquel on fait passer les paramètres.

Livre L1 = new Livre(123,"auteur1","java",34.90);

Déclenchement du "bon" constructeur ?

- Il se fait en fonction des paramètres passés lors de l'appel (nombre et types).
- Si le programmeur crée un constructeur (même si c'est un constructeur avec paramètres), le constructeur par défaut n'est plus disponible. Attention aux erreurs de compilation !

Les constructeurs (5/5)

```
public class Livre
{
    int id; // identifiant du livre
    String auteur y; // auteur du livre
    String titre; // titre du livre
    double prix; // prix du livre
    // plusieurs constructeurs
    Livre () {
        id=0123;
        titre="abc "
        prix=87.00;
    }
    Livre (int ident){
        id=ident;
        prix=20.00;
    }
    Livre (int ident, String aut, String
    tr, double p){
        id=ident;
        auteur=aut;
        titre="tr";
        prix=p;
    }
}
```

Redéfinition d'un
Constructeur sans
paramètres

```
public class TestLivre
{
    public static void main(String args[])
    {
        Livre L1= new Livre();
        Livre L2=new Livre(3);
        Livre L3=new Livre(3,4);
    }
}
```

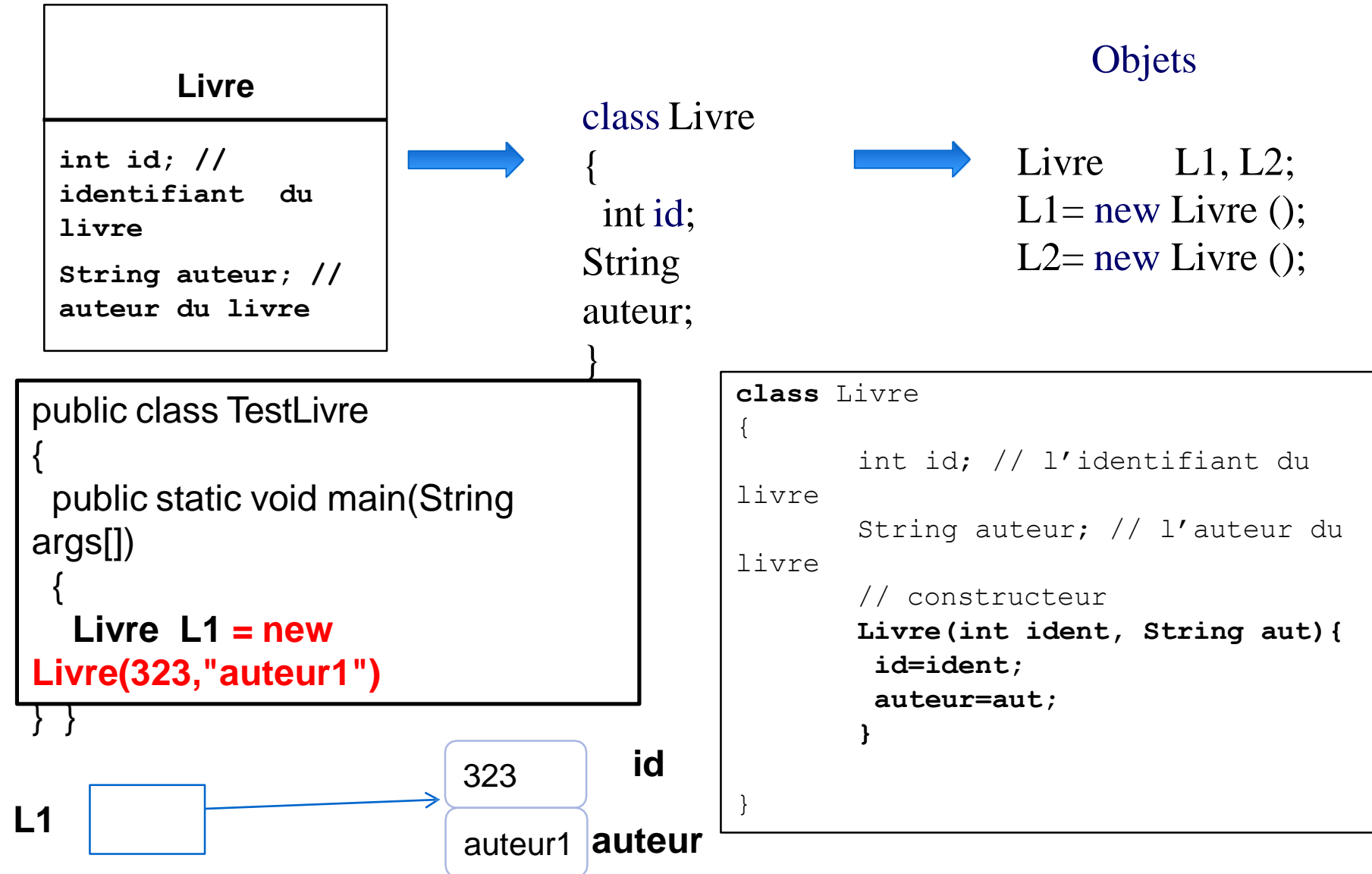
Erreur de compilation

On définit plusieurs constructeurs
qui se différencient uniquement
par leurs paramètres (surcharge)

La création d'objet (1/2)

- L'opérateur d'instanciation en Java est **new** :
MaClasse monObjet = new MaClasse();
- L'appel de **new** pour créer un nouvel objet déclenche, dans l'ordre :
 - L'allocation mémoire nécessaire au stockage de ce nouvel objet et l'initialisation par défaut de ces attributs,
 - L'initialisation explicite des attributs, s'il y a lieu,
 - L'exécution d'un **constructeur**.
- Pour accéder à une méthode on utilise Nomobjet.Méthode

La création d'objet (2/2)





Fin partie 2