

# INFO-H-200

## Programmation orientée objet

Séance d'exercices 2  
Classes et objets

Université libre de Bruxelles  
École polytechnique de Bruxelles

Professeur : Hugues Bersini

2015-2016

# Classes et objets

# Classes et objets

- Une classe peut être vue comme un nouveau type de donnée, un modèle, qui possède
  - ses propres variables (attributs)
  - son propre comportement (méthodes ou fonctions de la classe)
- Un objet est une instance d'une classe qui a une existence propre

## Exemples :

Classe → Personne

Objet → Jean Dupont

Classe → Voiture

Objet → Peugeot 308

# Définition de classes et attributs

En Java, on va créer un fichier par classe (MaClasse.java)

- **Attributs** : variables des instances de la classe
  - Variables de type simple, des tableaux ou des objets
  - Niveau de protection :
    - *public* : pour les attributs visibles à l'extérieur de la classe
    - *private* : pour les attributs visibles uniquement à l'intérieur.

**Exemple de définition d'une classe avec des attributs :**

```
public class Person {  
    private String firstName;  
    private String lastName;  
    private double salary = 1000;  
}
```

# Définition de classes et méthodes

- **Les fonctions (méthodes) :**
  - Les méthodes définissent le comportement des objets de la classe.
  - Elles peuvent prendre ou non des paramètres et renvoyer ou non une valeur de retour.
  - Elles ont aussi un niveau de protection (public / private)

## Exemple de définition d'une classe avec des attributs et une méthode

```
public class Person {  
    private String firstName;  
    private String lastName;  
    private double salary = 1000;  
  
    public String toString(){  
        return firstName + " " + lastName;  
    }  
}
```

# Constructeurs

Un constructeur est une méthode spéciale appelée lors de la création d'un objet.

- Cette méthode n'a pas de type et ne renvoie rien.
- Elle doit avoir le même nom que la classe.

## Exemple précédent + constructeur :

```
public class Person {
    private String firstName;
    private String lastName;
    private double salary = 1000;

    public String toString(){
        return firstName + " " + lastName;
    }

    public Person(String fn, String ln){
        firstName = fn;
        lastName = ln;
    }
}
```

# Constructeurs (suite)

Il peut y avoir plusieurs constructeurs par classe avec différents paramètres.

**this()** permet de réutiliser le code d'un constructeur dans un autre.

## Exemple précédent (suite) :

```
public class Person {
    private String firstName;
    private String lastName;
    private double salary = 1000;

    public String toString(){
        return firstName + " " + lastName;
    }

    public Person(String ln){
        firstName = "";
        lastName = ln;
    }

    public Person(String ln, String fn){
        this(ln);
        firstName = fn;
    }
}
```

# Mot clé 'this'

Permet de se référer aux attributs ou méthodes de la classe courante quand il y a une ambiguïté.

## Exemple précédent (suite) :

```
public class Person {
    private String firstName;
    private String lastName;
    private double salary = 1000;

    public String toString(){
        return firstName + " " + lastName;
    }

    public Person(String firstName, String lastName){
        this.firstName = firstName;
        this.lastName = lastName;
    }
}
```

# Accesseurs (set / get)

Un accesseur est une méthode pour accéder, en lecture ou en écriture aux attributs d'une classe.

\* **GET** : Un accesseur en lecture renverra l'attribut concerné.

```
public class Person {  
    private String firstName;  
    private String lastName;  
  
    public String getLastName() {  
        return lastName;  
    }  
}
```

\* **SET** : Un accesseur en écriture est généralement 'void'. Il prend un paramètre et modifié l'attribut concerné après avoir vérifié que cette modification ne portait pas atteinte à l'intégrité de l'objet.

```
public class Person {  
    private String firstName;  
    private String lastName;  
  
    public void setLastName(String ln) {  
        lastName = ln;  
    }  
}
```

# Créer et utiliser des objets

Pour créer un objet, il faut déclarer une variable du type de l'objet et instancier l'objet à l'aide du mot clé 'new' et d'un de ses constructeurs.

```
| Person jean = new Person("Dupond", "Jean");
```

Une fois l'objet créé, on peut accéder à ses attributs ou méthodes publiques à l'aide d'un point.

```
| jean.lastName = "Durant";  
| System.out.println(jean.toString());
```

# Exemple récapitulatif

Fichier 'Person.java'

```
package Example;

public class Person {
    private String firstName;
    private String lastName;
    private double salary = 1000;

    public String getFirstName(){                return this.firstName;}
    public String getLastName(){                return this.lastName;}
    private void setFirstName(String firstName){ this.firstName = firstName;}
    private void setLastName(String lastName){  this.lastName = lastName;}

    public String toString(){                   return firstName + " " + lastName;}

    public Person(String lastName){            setLastName(lastName);}

    public Person(String firstName, String lastName){
        this(lastName);
        this.firstName = firstName;
    }
}
```

Fichier 'Main.java'

```
package Example;

public class Main {
    public static void main(String args[]){
        Person me = new Person("Michael", "Waumans");
        System.out.println(me.toString());
        System.out.println(me);    // Appel implicite à la méthode 'toString'
    }
}
```