

Programmation Objet – Initiation à Java

Letícia SEIXAS PEREIRA

Cours 06 – Révision

18/11/2019



```
public class Rectangle {  
  
    private int longueur ;  
    private int largeur ;  
  
    // Mutateurs ...  
    // Accesseurs ...  
    // toString() ...  
  
    int surface () {  
        return this.longueur * this.largeur ;  
    }  
}
```

```
public class Rectangle {  
  
    private int longueur ;  
    private int largeur ;  
  
    // Mutateurs ...  
    // Accesseurs ...  
    // toString() ...  
  
    int surface () {  
        return this.longueur * this.largeur ;  
    }  
}
```

```
public class TestRectangle {  
    public static void main (String[] args) {  
        Rectangle rectangle1;  
        rectangle1 = new Rectangle();  
    }  
}
```

```
public class Rectangle {  
  
    private int longueur ;  
    private int largeur ;  
  
    // Mutateurs ...  
    // Accesseurs ...  
    // toString() ...  
  
    int surface () {  
        return this.longueur * this.largeur ;  
    }  
}
```

```
public class TestRectangle {  
    public static void main (String[] args) {  
        Rectangle rectangle1;  
        rectangle1 = new Rectangle(3, 4);  
        System.out.println(rectangle1.getLongueur());  
    }  
}
```

```
public class Rectangle {  
  
    private int longueur ;  
    private int largeur ;  
  
    // Mutateurs ...  
    // Accesseurs ...  
    // toString() ...  
  
    int surface () {  
        return this.longueur * this.largeur;  
    }  
}
```

```
public class TestRectangle {  
    public static void main (String[] args) {  
        Rectangle rectangle1;  
        rectangle1 = new Rectangle(3, 4);  
        System.out.println(rectangle1.getLongueur());  
    }  
}
```

```
javac TestRectangle.java
```

```
TestRectangle.java:4: error: constructor Rectangle in class Rectangle cannot  
be applied to given types;
```

```
rectangle1 = new Rectangle(3, 4);
```

^

required: no arguments

found: int,int

reason: actual and formal argument lists differ in length

Les Constructeurs en Java

1. Classe **sans** constructeur explicite;
2. Classe **avec** constructeur explicite;
3. Classe x Héritage.

Les Constructeurs en Java

1. Classe **sans** constructeur explicite:

```
public class Animal {  
  
    private double poids;  
    private String groupe;  
  
    // accesseurs et mutateurs  
    // toString()  
  
}
```

Les Constructeurs en Java

1. Classe sans constructeur explicite:

```
public class Animal {  
  
    private double poids;  
    private String groupe;  
  
    // accesseurs et mutateurs  
    // toString()  
  
}
```

```
public class Zoo {  
    public static void main(String[] args)  
    {  
        Animal a1 = new Animal();  
        a1.setPoids(5);  
        a1.setGroupe("Mammifere");  
    }  
}
```


Les Constructeurs en Java

1. Classe sans constructeur explicite:

```
public class Animal {  
  
    private double poids;  
    private String groupe;  
  
    // accesseurs et mutateurs  
    // toString()  
  
}
```

```
public class Zoo {  
    public static void main(String[] args)  
    {  
        Animal a1 = new Animal();  
        a1.setPoids(5);  
        a1.setGroupe("Mammifere");  
    }  
}
```

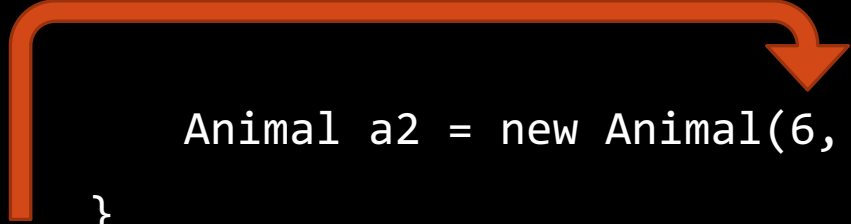
- Pas besoin de déclarer un constructeur;
- Si vous ne fournissez aucun constructeur, JAVA en créera un pour vous.

Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(double poid, String groupe){  
        this.poid = poid;  
        this.groupe = groupe;  
    }  
    // accesseurs et mutateurs  
    // toString()  
}
```

```
public class Zoo {  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```



- Créer un objet et insérer dans cet objet des valeurs au moment de sa création, sans avoir besoin d'utiliser la méthode set() à cet effet

Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(double poid, String groupe){  
        this.poid = poid;  
        this.groupe = groupe;  
  
    }  
    // accesseurs et mutateurs  
    // toString()  
}
```

```
public class Zoo {  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```

- Chaque fois qu'un constructeur avec des arguments est déclaré, le compilateur ne créera pas le constructeur par défaut.
- Nous devons donc le créer explicitement si nous devons l'utiliser.

Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
    }  
  
    public Animal(double poid, String groupe){  
        this.poid = poid;  
        this.groupe = groupe;  
    }  
  
    // accesseurs et mutateurs  
    // toString()
```

```
public class Zoo {  
  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```

- Chaque fois qu'un constructeur avec des arguments est déclaré, le compilateur ne créera pas le constructeur par défaut.
- Nous devons donc le créer explicitement si nous devons l'utiliser.

Les Constructeurs en Java

2. Classe **avec** constructeur explicite:

- Chaque fois qu'un constructeur avec des arguments est déclaré, le compilateur ne créera pas le constructeur par défaut;
- Nous devons donc le créer explicitement si nous devons l'utiliser;
- **Il est possible de créer plusieurs constructeurs dans une même classe;**
- **Il est possible d'appeler le constructeur par défaut à partir d'un autre constructeur.**

Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        System.out.println("Constructeur default");  
    }  
    public Animal(double poid, String groupe){  
        this();  
        this.poid = poid;  
        this.groupe = groupe;  
    }  
    // accesseurs et mutateurs
```

```
public class Zoo {  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```

```
java Zoo  
Constructeur default  
Constructeur default
```

Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        System.out.println("Constructeur default");  
    }  
    public Animal(double poid, String groupe){  
        this();  
        this.poid = poid;  
        this.groupe = groupe;  
    }  
    // accesseurs et mutateurs
```

```
public class Zoo {  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```

```
java Zoo  
Constructeur default  
Constructeur default
```

Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        System.out.println("Constructeur default");  
    }  
    public Animal(double poid, String groupe){  
        this();  
        this.poid = poid;  
        this.groupe = groupe;  
    }  
    // accesseurs et mutateurs
```

```
public class Zoo {  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```

```
java Zoo  
Constructeur default  
Constructeur default
```


Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        System.out.println("Constructeur default");  
    }  
    public Animal(double poid, String groupe){  
        this();  
        this.poid = poid;  
        this.groupe = groupe;  
    }  
    // accesseurs et mutateurs
```

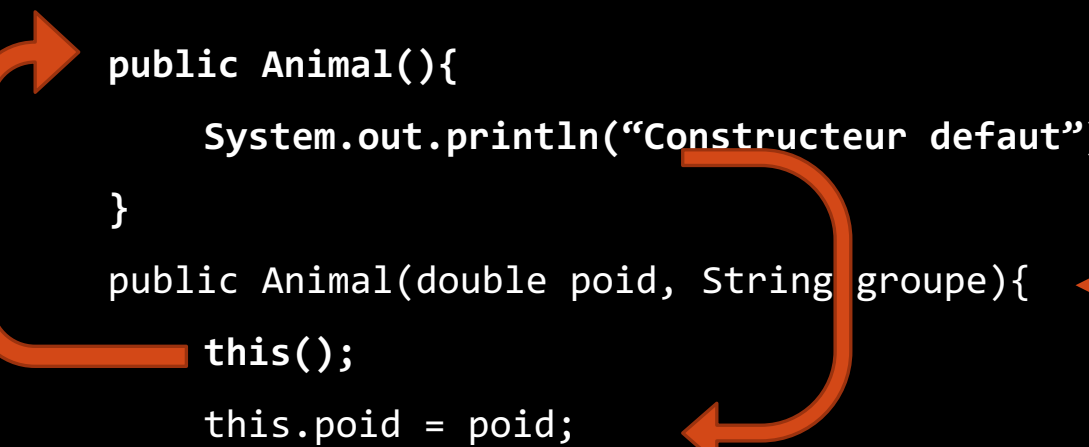
```
public class Zoo {  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```

```
java Zoo  
Constructeur default  
Constructeur default
```

Les Constructeurs en Java

2. Classe **avec** constructeur explicite;

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        System.out.println("Constructeur default");  
    }  
    public Animal(double poid, String groupe){  
        this();  
        this.poid = poid;  
        this.groupe = groupe;  
    }  
    // accesseurs et mutateurs
```



```
public class Zoo {  
    public static void main(String[] args) {  
        Animal a1 = new Animal();  
        a1.setPoid(5);  
        a1.setGroupe("Mammifere");  
  
        Animal a2 = new Animal(6, "Oiseaux");  
    }  
}
```

```
java Zoo  
Constructeur default  
Constructeur default
```

Les Constructeurs en Java

2. Classe **avec** constructeur explicite:

- Chaque fois qu'un constructeur avec des arguments est déclaré, le compilateur ne créera pas le constructeur par défaut;
- Nous devons donc le créer explicitement si nous devons l'utiliser;
- **Il est possible de créer plusieurs constructeurs dans une même classe;**
- **Il est possible d'appeler le constructeur par défaut à partir d'un autre constructeur;**
- **La méthode `this()` appellera toujours le constructeur défaut de classe elle-même.**

Les Constructeurs en Java

3. Classe x Héritage

- Lorsque nous travaillons avec l'héritage en Java, nous utilisons le mot-clé **extends** pour faire référence à la classe héritée;

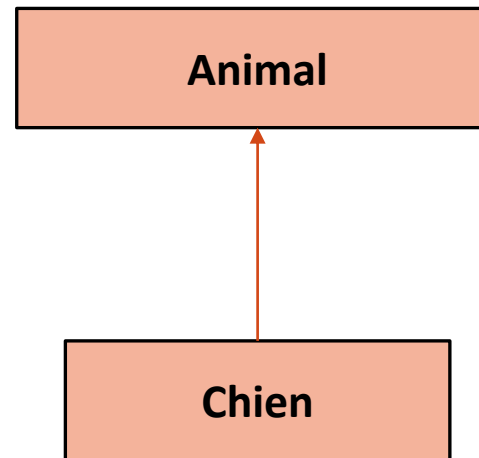
Les Constructeurs en Java

3. Classe x Héritage

- Lorsque nous travaillons avec l'héritage en Java, nous utilisons le mot-clé **extend** pour faire référence à la classe héritée;
- Chaque fois qu'une classe est instanciée, son constructeur est appelé et la première exécution à l'intérieur du constructeur est un appel à `super()`, qui appelle le constructeur de sa classe mère.

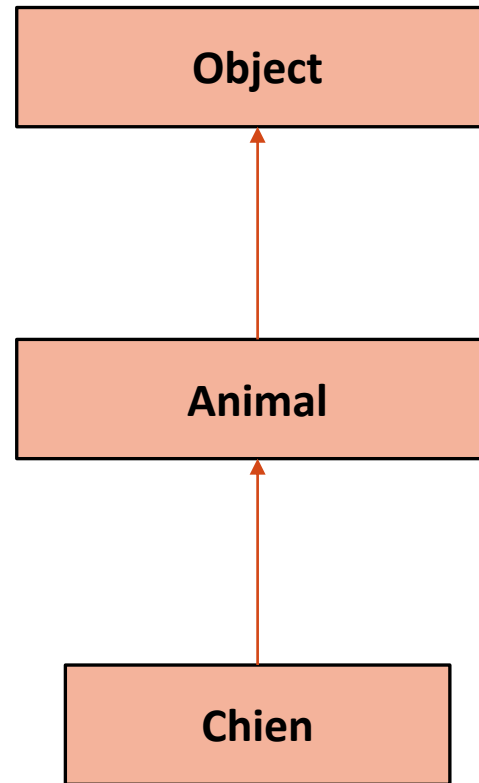
Les Constructeurs en Java

3. Classe x Héritage



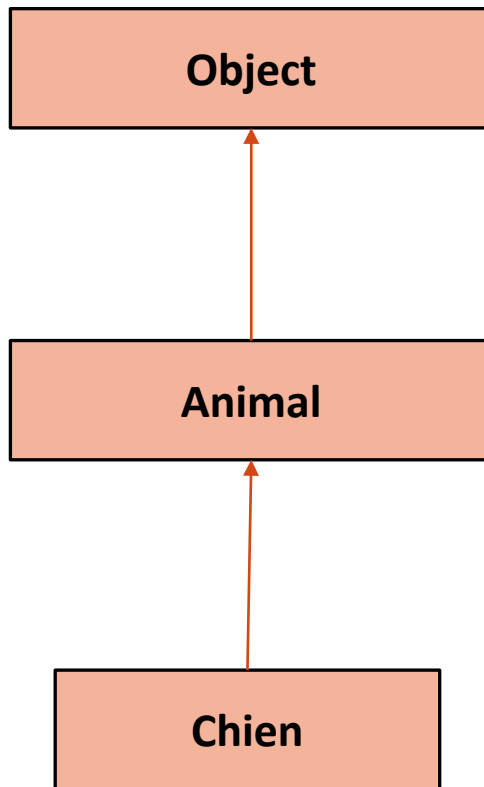
Les Constructeurs en Java

3. Classe x Héritage



Les Constructeurs en Java

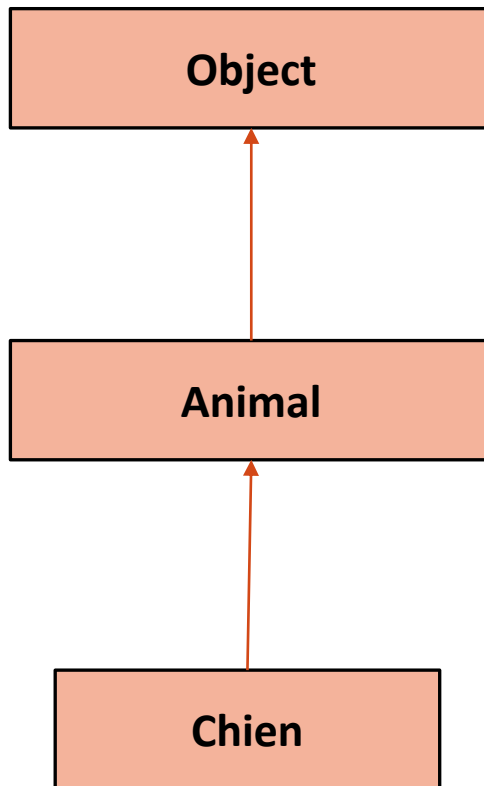
3. Classe x Héritage



1. Chien chien = new Chien ()	Appelle	public Chien () { super(); }
2. public Chien() { super(); }	Appelle	public Animal() { super(); }
3. public Animal() { super(); }	Appelle	public Object() { }
4. public Object() { }		

Les Constructeurs en Java

3. Classe x Héritage



1. Chien chien = new Chien ()	Appelle	public Chien () { super(); }
2. public Chien() { super();}	Appelle	public Animal() { super(); }
3. public Animal() { super(); }	Appelle	public Object() { }
4. public Object() { }		

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        super();  
        System.out.println("Constructeur default");  
    }  
}
```

Les Constructeurs en Java

3. Classe x Héritage

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        // super();  
        System.out.println("Constructeur default: Animal");  
    }  
    // ...  
}
```

```
public class Chien extends Animal{  
  
    public Chien(){  
        System.out.println("Constructer default: Chien");  
    }  
}
```

```
public class Zoo {  
    public static void main(String[] args) {  
  
        Chien c1 = new Chien();  
    }  
}
```

Les Constructeurs en Java

3. Classe x Héritage

```
public class Animal {  
  
    private double poid;  
    private String groupe;  
  
    public Animal(){  
        // super();  
        System.out.println("Constructeur default: Animal");  
    }  
    // ...  
}
```

```
public class Chien extends Animal{  
  
    public Chien(){  
        // super();  
        System.out.println("Constructer default: Chien");  
    }  
}
```

```
public class Zoo {  
    public static void main(String[] args) {  
  
        Chien c1 = new Chien();  
    }  
}
```

```
java Zoo  
Constructeur default: Animal  
Constructer default: Chien
```