



# Programmation Orientée Objet

Les notions de base

**Mme Hassna BENSAG**  
Email: [h.bensag@gmail.com](mailto:h.bensag@gmail.com)


# Paradigme orientée objet

---

- La programmation orientée objet permet de concevoir une application sous la forme d'un ensemble d'objets reliés entre eux par des relations.
- Lorsque que l'on programme avec ce paradigme, la première question que l'on se pose plus souvent est :
  - **"qu'est-ce que je manipule ? "**
  - Au lieu de **" qu'est-ce que je fais ? "**
- L'une des caractéristiques de ce paradigme permet de concevoir de nouveaux objets à partir d'objets existants.
  - On peut donc réutiliser les objets dans plusieurs applications.
  - La réutilisation du code fut un argument déterminant pour vanter les avantages des langages à objets.
- Pour faire la programmation orientée objet il faut maîtriser les fondamentaux de l'orienté objet à savoir:
  - Objet et classe
  - Héritage
  - Encapsulation
  - Polymorphisme

# Notion d'objet

---

- Objet=état + comportement
  - L'état regroupe les valeurs instantanées de tous les attributs de l'objet.
  - Le comportement regroupe toutes les actions de l'objet. Autrement dit le comportement est défini par les opérations que l'objet peut effectuer.
  - L'état d'un objet peut changer dans le temps.
  - Généralement, c'est le comportement qui modifie l'état de l'objet.
- En plus de son état, un objet possède une identité qui caractérise son existence propre.
- Cette identité s'appelle également référence ou handle de l'objet.
- En terme informatique de bas niveau, l'identité d'un objet représente son adresse mémoire.
- Deux objets ne peuvent pas avoir la même identité  deux objets ne peuvent pas avoir le même emplacement mémoire.

# Notion de classe

---

- Les objets qui ont des caractéristiques communes sont regroupés dans une entité appelé classe.
- La classe décrit le domaine de définition d'un ensemble d'objets.
- Chaque objet appartient à une classe.
- Les généralités sont contenues dans les classes et les particularités dans les objets.
- Les objets sont construits à partir de leur classe par un processus qui s'appelle l'instanciation.
- Tout objet est une instance d'une classe.

# Caractéristique d'une classe

---

- Une classe est définie par:
  - Les attributs.
  - Les méthodes.
- Les attributs permettent de décrire l'état des objets de cette classe.
- Chaque attribut est défini par son nom, son type et sa valeur initiale.
- Les méthodes permettent de décrire le comportement des objets de cette classe.
- Une méthode représente une procédure ou une fonction qui permet d'exécuter un certain nombre d'instructions.
- Parmi les méthodes d'une classe, existe deux méthodes particulières:
  - Une méthode qui est appelée au moment de la création d'un objet de cette classe. Cette méthode est appelée **CONSTRUCTEUR**.
  - Une méthode qui est appelée au moment de la destruction d'un objet. Cette méthode s'appelle le **DESTRUCTEUR**

# Les 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.
- Les packages divisent et organisent les modèles de la même manière que les dossiers organisent le système de fichier.



# Accessibilité aux membres d'une classe

- En java, il existe 4 niveaux de protection :
  - public : accès à partir de toute entité interne ou externe à la classe.
  - private : Un membre privé d'une classe n'est accessible qu'à l'intérieur de cette classe.
  - protected : un membre protégé d'une classe est accessible à l'intérieur de cette classe, Aux classes dérivées de cette classe, et Aux classes du même package.
  - Autorisation par défaut : dans java, en l'absence des trois autorisations précédentes, l'autorisation par défaut est package. Cette autorisation indique que uniquement les classes du même package ont l'autorisation d'accès.

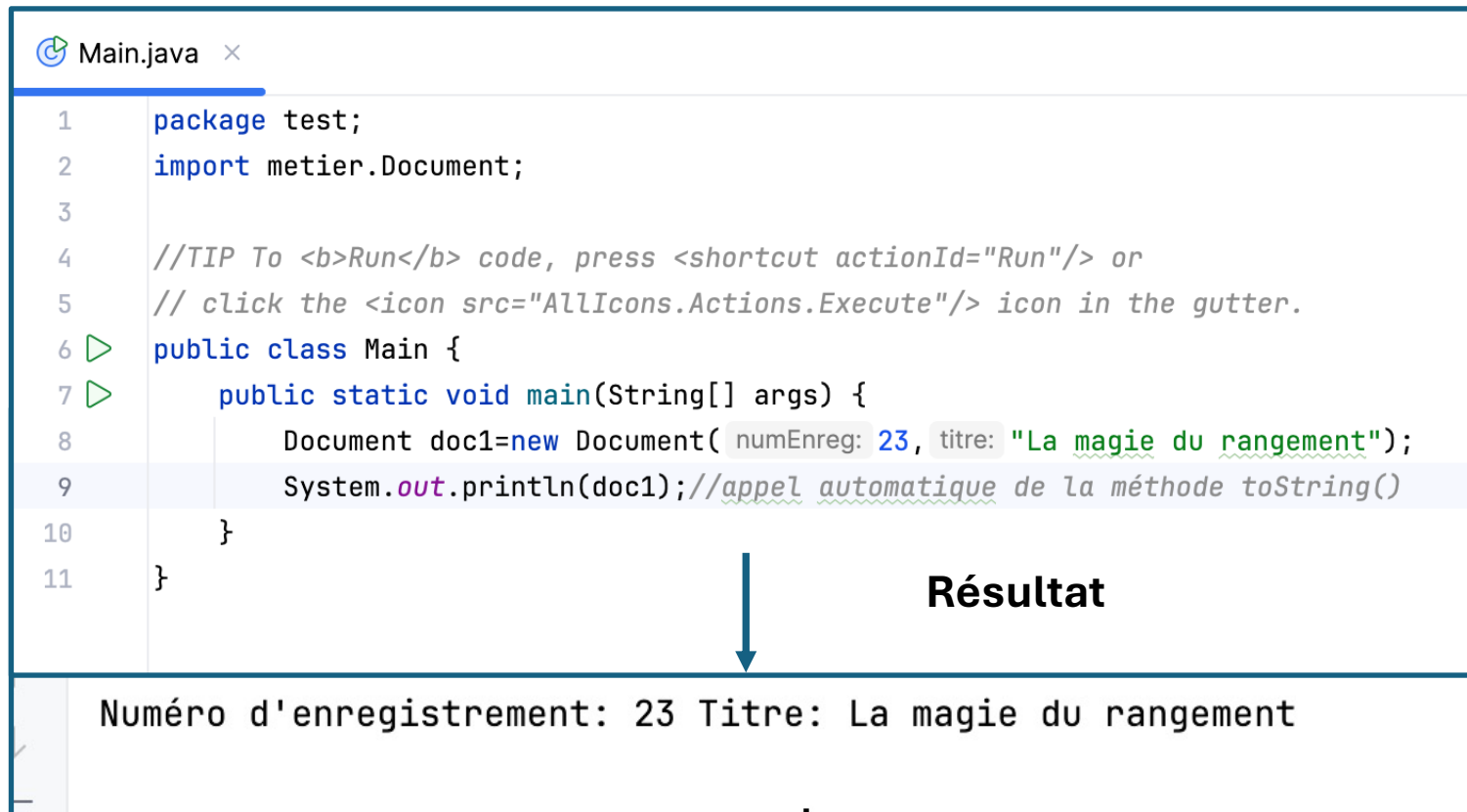
Accessibilité du membre	private	Par défaut	protected	public
Accès depuis la classe	Oui	Oui	Oui	Oui
Accès depuis une classe du même package	Non	Oui	Oui	Oui
Accès depuis une classe dérivée	Non	Non	Oui	Oui
Accès depuis toute autre classe	Non	Non	Non	Oui

# Exemple d'implémentation d'une classe

```
Document.java ×
1 package metier;
2
3 public class Document { 3 usages
4     private int numEnreg; 2 usages
5     private String titre; 2 usages
6     public Document(int numEnreg,String titre){ 1 usage
7         //ici this est obligatoire: nom du paramètre est identique au nom de l'attribut
8         this.numEnreg=numEnreg;
9         this.titre=titre;
10    }
11
12    public String toString() {
13        return ("Numéro d'enregistrement: "+numEnreg+" Titre: "+titre);
14    }
15 }
16
```



# Instanciation et accès au méthodes



The screenshot shows a Java IDE window titled 'Main.java'. The code defines a package 'test', imports 'metier.Document', and contains a 'Main' class with a 'main' method. The 'main' method creates a 'Document' object with 'numEnreg: 23' and 'titre: "La magie du rangement"', and prints it. A blue arrow points from the 'main' method to the output window below. The output window displays the text: 'Numéro d'enregistrement: 23 Titre: La magie du rangement'.

```
1 package test;
2 import metier.Document;
3
4 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
5 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
6 public class Main {
7     public static void main(String[] args) {
8         Document doc1=new Document( numEnreg: 23, titre: "La magie du rangement");
9         System.out.println(doc1);//appel automatique de la méthode toString()
10    }
11 }
```

**Résultat**

Numéro d'enregistrement: 23 Titre: La magie du rangement

# Constructeur par défaut

- Quand on ne définit aucun constructeur pour une classe, le compilateur crée le constructeur par défaut.
- Ce constructeur par défaut n'a aucun paramètre et il initialise les attributs de l'objet par des par défaut.
- Exemple:

```
Document.java x
1 package metier;
2
3 public class Document { 3 usages
4     private int numEnreg; 1 usage
5     private String titre; 1 usage
6
7
8     public String toString() {
9         return ("Numéro d'enregistrement: "+numEnreg+" Titre: "+titre);
10    }
11 }
12
```

```
Main.java x
1 package test;
2 import metier.Document;
3
4 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
5 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
6 public class Main {
7     public static void main(String[] args) {
8         Document doc1=new Document();
9         System.out.println(doc1);//appel automatique de la méthode toString()
10    }
11 }
```

Numéro d'enregistrement: 0 Titre: null

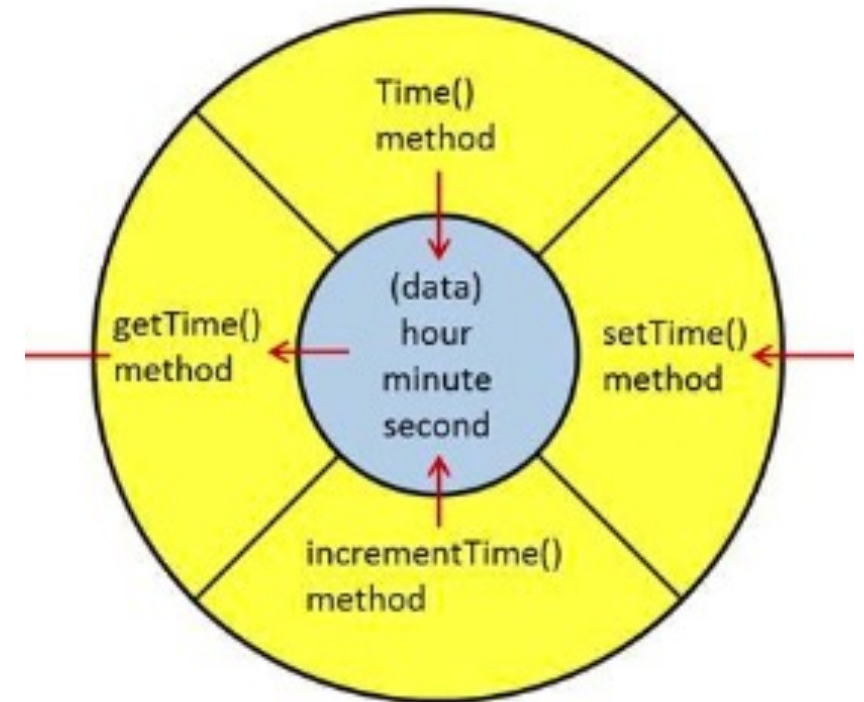
# Les accesseurs et mutateurs

```
Document.java x
2
3 public class Document { 3 usages
4     private int numEnreg; 4 usages
5     private String titre; 4 usages
6     public Document(int numEnreg,String titre){ 1 usage
7         //ici this est obligatoire: nom du paramètre est identique au nom de l'
8         this.numEnreg=numEnreg;
9         this.titre=titre;
10    }
11
12    public int getNumEnreg() { no usages
13        return numEnreg;
14    }
15
16    public void setNumEnreg(int numEnreg) { no usages
17        this.numEnreg = numEnreg;
18    }
19
20    public String getTitre() { no usages
21        return titre;
22    }
23
24    public void setTitre(String titre) { no usages
25        this.titre = titre;
26    }
27
28    public String toString() {
29        return ("Numéro d'enregistrement: "+numEnreg+" Titre: "+titre);
30    }
```

# Encapsulation

---

- Généralement, l'état d'un objet est privé ou protégé et son comportement est public.
- Quand l'état de l'objet est privé Seules ses méthodes ont le droit d'y accéder.
- Quand l'état de l'objet est protégé, les méthodes des classes dérivées et les classes appartenant au même package peuvent également y accéder.
- Le sens de l'encapsulation est de s'assurer que les données "sensibles" sont cachées aux utilisateurs. Pour y parvenir, il faut :
  - Déclarer les variables/attributs de classe comme privés.
  - fournir des méthodes publiques get et set pour accéder et mettre à jour la valeur d'une variable privée.



# Membres statiques

---

- Le mot-clé `static` en Java indique que le membre (attribut ou méthode) appartient à la classe plutôt qu'à une instance spécifique de la classe.
- Un attribut statique est partagé entre toutes les instances de la classe.
- Une méthode statique ne peut accéder qu'aux attributs statiques. Elles peuvent être appelées sans instancier la classe.
- Dans la notation UML, les membres statiques d'une classe sont soulignés.

# Membres statiques

Document.java

```
1 package metier;
2
3 public class Document { 4 usages
4     private int numEnreg; 4 usages
5     private String titre; 4 usages
6     private static int nbrDocument=0; 2 usages
7     public Document(int numEnreg,String titre){ 1 usage
8         //ici this est obligatoire: nom du paramètre est identique au nom de l'attribut
9         this.numEnreg=numEnreg;
10        this.titre=titre;
11        nbrDocument++;
12    }
13
14    public static int getNbrDocument() { no usages
15        return nbrDocument;
16    }
17
18    public int getNumEnreg() { no usages
19        return numEnreg;
20    }
21
22    public void setNumEnreg(int numEnreg) { no usages
23        this.numEnreg = numEnreg;
24    }
25
26    public String getTitre() { no usages
27        return titre;
28    }
```

Main.java

```
1 package test;
2 import metier.Document;
3
4 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
5 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
6 public class Main {
7     public static void main(String[] args) {
8         Document doc1=new Document( numEnreg: 23, titre: "La magie du rangement");
9         Document doc2=new Document( numEnreg: 54, titre: "La fille de papier");
10        System.out.println(doc1);//appel automatique de la méthode toString()
11        System.out.println("Le nombre de document est : "+Document.getNbrDocument());
12    }
13 }
```

# héritage

---

- Une classe peut hériter d'une autre classe en utilisant le mot `extends`.
- Une classe Hérite d'une autre classe tous ses membres sauf le constructeur.
- Il faut toujours définir le constructeur de la nouvelle classe dérivée.
- Le constructeur de la classe dérivée peut appeler le constructeur de la classe parente en utilisant le mot `super()`, avec la liste des paramètres.

# Redéfinition des méthodes

---

- Quand une classe hérite d'une autre classe, elle peut redéfinir les méthodes héritées.
- La redéfinition d'une méthode dans une classe dérivée permet de fournir une implémentation spécifique d'une méthode qui a déjà été définie dans la superclasse.
- Dans la méthode redéfinie de la nouvelle classe dérivée, on peut faire appel à la méthode de la classe parente en utilisant le mot `super` suivi d'un point et du nom de la méthode.
- Les méthodes « `private`, `final`, `static` » ne peuvent pas être redéfinis.
- La méthode redéfinis doit avoir le même type de retour.




# Héritage – Exemple

Enoncé: on veut implémenter en java une application devant servir à l'inventaire d'une bibliothèque. Elle devra traiter des documents de nature diverse: des livres et des dictionnaires.

- 1) Tous les documents possèdent un titre.
  - ☐ Quand un document est créé:
    - son titre est donné à la création
    - on veut attribuer un numéro d'enregistrement unique dès que l'on crée un objet Document. On veut que le premier document créé ait le numéro 0, et le numéro doit s'incrémenter de 1 à chaque création
  - ☐ Définir un constructeur et une méthode toString()
- 2) A chaque livre est associé, en plus un auteur et un nombre de pages
  - ☐ définir au sein de la classe livre une méthode volumineux(), qui vérifie si le livre est volumineux ou non. Pour qu'un livre soit volumineux son nombre de page dépasse 400.
  - ☐ Définir un constructeur et une méthode toString().
- 3) Les dictionnaires ont, eux pour attributs supplémentaires une langue et un nombre de tomes:
  - ☐ définir une méthode multipleTome(), vérifie si le dictionnaire est divisé en plusieurs tomes
  - ☐ Définir un constructeur et une méthode toString().

# La classe mère - Dictionnaire



```
Document.java x
1 package metier;
2
3 public class Document { 2 usages 2 inheritors
4     private int numEnreg; 2 usages
5     private String titre; 2 usages
6     private static int count=0; 2 usages
7
8     public Document(String titre){ 2 usages
9         //ici this est obligatoire: nom du paramètre est identique au
10        this.numEnreg=count;
11        this.titre=titre;
12        count++;
13    }
14
15    public String toString() { 2 overrides
16        return ("Numéro d'enregistrement: "+numEnreg+" Titre: "+titre)
17    }
18 }
19
```

# La classe dérivée - Livre

© Livre.java x

```
1 package metier;
2
3 public class Livre extends Document{ 3 usages
4     private String auteur; 2 usages
5     private int nbrPage; 3 usages
6
7     public Livre(String titre,String auteur,int nbrPage){ 1 usage
8         super(titre);
9         this.auteur=auteur;
10        this.nbrPage=nbrPage;
11    }
12    public String volumineux(){ 1 usage
13        if (this.nbrPage>400)
14            return " Le livre est volumineux";
15    else
16        return " Le livre n'est pas volumineux";
17    }
18    public String toString() {
19        return super.toString()+" Auteur: "+auteur+volumineux()+"il contient: "+nbrPage+" pages";
20    }
21 }
22
```

# La classe dérivée - Dictionnaire

```
© Dictionnaire.java x
1 package metier;
2
3 public class Dictionnaire extends Document{ 3 usages
4     private String langue; 2 usages
5     private int nbrTome; 3 usages
6
7     public Dictionnaire(String titre,String langue,int nbrTome){ 1 usage
8         super(titre);
9         this.langue=langue;
10        this.nbrTome=nbrTome;
11    }
12    public String multipleTome(){ 1 usage
13        if (this.nbrTome>1)
14            return " Le dictionnaire est divisé en plusieurs tomes";
15        else
16            return " Le livre ne contient qu'un seul Tome";
17    }
18    public String toString() {
19        return super.toString()+" langue: "+langue+multipleTome()+"il contient: "+nbrTome+" tomes";
20    }
21 }
```