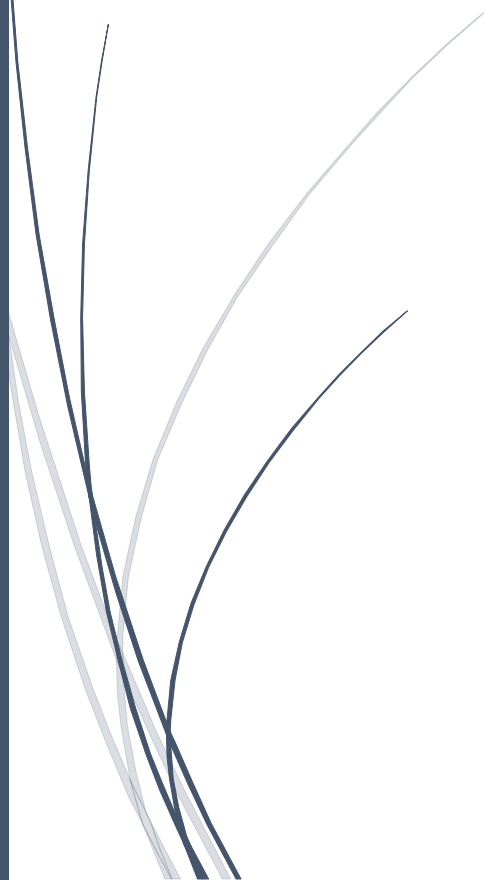


15/02/2024

# 420-210-MV

## Rencontre 07

Programmation orientée objet



---

## *Programmation orientée objet*

---

La programmation orientée objet (POO) est un paradigme de programmation qui fait le lien avec la manière dont nous concevons le monde.

La POO repose sur l'assemblage des objets, définis par un état et un comportement.

Le terme « orienté objet » signifie que l'on organise notre code en 1 ou plusieurs objets, qui peuvent communiquer entre eux.

---

## *Les concepts de la programmation orientée objet*

---

- Classes et objets
- Encapsulation
- Héritage
- Polymorphisme
- Abstraction

---

## *Classes et objets*

---

### Notion de Classe

Une classe est un ensemble de données et de fonctions regroupées dans une même entité.

Une classe est une description abstraite d'un objet. Les fonctions qui opèrent sur les données sont appelées des méthodes.

Une classe est une sorte de moule qui permet ensuite de créer autant d'instances qu'on veut.

Une classe est la description d'un objet.

---

## *Package*

---

### Notion de Package

Les classes Java sont regroupées par finalités dans des ensembles appelés packages eux-mêmes organisés sous forme hiérarchique. Quand vous rédigez le code source et que vous souhaitez

utiliser une classe d'un package donné il vous faut soit préciser son "chemin complet" à chaque appel ou, de façon plus concise, déclarer son importation dans l'en-tête du fichier source. Les classes "standard" du package *java.lang* ne suivent pas cette règle ; elles sont directement accessibles !

Les classes que vous allez développer devront obligatoirement appartenir à des packages que vous allez donc devoir créer. L'organisation de ces ensembles de classes et de leurs hiérarchies vous permet de contrôler la "portée" des packages, des classes et des méthodes.

Il existe des conventions de nommage pour les packages (comme d'ailleurs pour les classes, les méthodes, etc.). Celles-ci peuvent varier d'une entreprise à l'autre. Généralement le package commence par le nom d'un type de domaine (com, edu, gov, mil, net, org) suivi par un point (.) suivi par le nom de la société lui-même suivi par un point (.). Ensuite, il peut y avoir un nom de projet ou de module utilisable dans plusieurs projets, etc.

*Exemple de nom de package : ca.cegepmv.rencontre07*

### Format d'écriture d'une classe


En général la syntaxe de déclaration d'une classe se divise en 03 parties :

**1ère Partie:** déclaration de classe

```
public class NomDeLaClasse {  
  
}
```

**2ème partie:** déclaration des attributs (ou propriétés de la classe)

Je prends l'exemple d'un cas, ou j'ai N attributs



```
int attribut_Un ;  
  
String attribut_Deux ;  
  
...  
  
boolean attribut_N ;
```

**3ème partie:** déclaration des constructeurs

Un constructeur est une méthode qui permet de créer des instances d'une classe.

➤ Exemple de constructeur sans paramètres :

```
public NomDeLaClasse () {
```

```
}
```

- Exemple de constructeur avec un seul paramètre

```
public NomDeLaClasse ( int unAttribut_un) {  
    attribut_Un = unAttribut_un ;  
}
```

- Exemple de constructeur avec N paramètres

```
public NomDeLaClasse ( int unAttribut_un, . . . . . , boolean unAttribut_N) {  
  
    attribut_Un = unAttribut_un ;  
  
    .....  
  
    .....  
  
    attribut_N = unAttribut_N ;  
  
}
```

---

### *Constructeur d'une classe*

---

Un constructeur est le nom donné à la méthode qui permet de créer une instance de classe (une copie de l'objet).

La méthode de type constructeur ne retourne rien et doit être publique.

Par défaut, toute classe possède un constructeur sans paramètre.

Une classe peut avoir plusieurs constructeurs.

#### Exemple code Java

#### Notion d'instance

Une instance est une occurrence d'une classe.

Instancier une classe consiste à créer un objet sur son modèle.

Instancier une classe consiste à créer un objet sur son modèle.

Entre classe et objet il y a, en quelque sorte, le même rapport qu'entre type et variable. Pour accéder à une classe il faut en déclarer une instance de classe ou objet.

---

### *new*

---

L'opérateur new permet d'instancier une classe, c'est-à-dire de créer une instance de cette classe.

---

### *static*

---

Un élément déclaré static appartient à une classe et non à ses instances. Les objets instanciés à partir d'une classe ne possèdent pas les éléments de cette classe qui ont été déclaré static. Un seul élément existe pour la classe et il est partagé par toutes les instances. Cela ne limite en aucune façon l'accessibilité mais conditionne le résultat obtenu lors des accès. Les primitives, les objets et les méthodes peuvent être déclaré static.

*Exemple code Java*

---

### *Notion d'héritage*

---

L'héritage est un principe clé de la programmation orientée objet. Cela implique le transfert de la structure existante d'une classe, y compris son constructeur, ses variables et ses méthodes, vers une classe différente.

La nouvelle classe est appelée classe enfant (ou sous-classe), tandis que celle dont elle hérite, est appelée classe parent (ou superclasse).

On dit que la classe enfant étend la classe parent. On dit que la classe enfant étend la classe parent dans le sens où non seulement elle hérite des structures définies par le parent, mais elle crée également de nouvelles structures.

*Exemple code Java*

---

*La visibilité en Java*

---

L'un des avantages des classes est que les classes peuvent protéger leurs variables et méthodes membres contre l'accès par d'autres objets.

Pourquoi est-ce important? Eh bien, considérez ceci. Vous écrivez une classe qui représente une requête sur une base de données contenant toutes sortes d'informations secrètes, par exemple des dossiers d'employés ou des déclarations de revenus pour votre entreprise en démarrage.

Certaines informations et requêtes contenues dans la classe, celles prises en charge par les méthodes et variables accessibles au public dans votre objet de requête, sont acceptables pour la consommation de tout autre objet du système. Les autres requêtes contenues dans la classe sont là simplement pour l'usage personnel de la classe. Ils prennent en charge le fonctionnement de la classe mais ne doivent pas être utilisés par des objets d'un autre type. Vous avez des informations secrètes à protéger. Vous aimeriez pouvoir protéger ces variables et méthodes personnelles au niveau du langage et interdire l'accès aux objets d'un autre type.

En Java, vous pouvez utiliser des spécificateurs d'accès pour protéger à la fois les variables d'une classe et ses méthodes lorsque vous les déclarez. Le langage Java prend en charge quatre niveaux d'accès distincts pour les variables membres et les méthodes : privé, protégé, public et, s'il n'est pas spécifié, package.

Le tableau suivant montre le niveau d'accès autorisé par chaque spécificateur.

Mot clé	Classe	Classe enfant	Package	Partout
<b>private</b>	✓			
<b>protected</b>	✓	✓*	✓	
<b>public</b>	✓	✓	✓	✓
<b>package</b>	✓		✓	

[Demo3](#)

**class**

- public
- final
- abstract
- default

**attribut**

- public
- final
- private
- protected
- default

**method**

- public
- final
- private
- protected
- default
- abstract
- static