

Support de cours

Cours:

## Introduction à la programmation orientée objet (en Java)

Vidéo:

### W17-01-pbgeneral-JAVA-OG

Concepts (extraits des sous-titres générés automatiquement) :

**Différents mécanismes. Mécanisme digital. Mécanisme analogique. Mécanismes doubles. Calcul du prix des montres. Aiguille des montres digitales. Montres analogiques. Mécanismes de base. Modélisation du problème d'un point. Niveau du produit. Point de vue du prix. Affichage polymorphique. Notion de conception du problème. Sortes de mécanisme. Vue de la poo.**



[vers la recherche de séquences vidéo](#)

(dans Introduction à la programmation orientée objet (en Java).)



[vers la vidéo](#)

Center for Digital Education. Plus de matériel de soutien pédagogique ici :

<https://www.epfl.ch/education/educational-initiatives/cede/educational-technologies-gallery/boocs-en/>

# Etude de cas : présentation et modélisation du problème

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Voici donc la dernière série de séquence vidéo

notes

---

---

---

---

---

---

---

---

---

---

résumé

0m 1s



---

---

---

---

---

---

---

---

---

---

## Problématiques abordées :

- ▶ conception POO, héritage, classes abstraites (prix, affichage)
- ▶ affichage polymorphique
- ▶ interfaces
- ▶ copie polymorphique

de ce mooc d'introduction à la programmation orienté objet (POO) illustrée en Java. Dans cette dernière série de séquence vidéo, nous avons voulu vous présenter un problème dans sa globalité, un problème très général, vous présenter une étude de cas où il s'agira donc de représenter différentes montres, il y aura des montres analogiques, des montres à aiguille des montres digitales, des montres qui seront les deux, les montres auront des accessoires comme des bracelets, des boîtiers... Donc un cadre assez général qui pourrait par exemple servir à un logiciel de gestion de bijouterie ou un logiciel de vente ou même d'impression de catalogues, dans le cadre duquel nous allons pouvoir vous illustrer différents concepts présentés tout au long de ce cours. Les thématiques que nous avons choisi de vous présenter dans ces vidéos de synthèse, contiennent tout d'abord bien sûr la notion de conception du problème, la modélisation du problème d'un point de vue de la POO, quelles classes doivent hériter de quelle autre, est-ce qu'il doit y avoir des classes abstraites, comment par exemple rendre le calcul du prix des montres polymorphique, ou l'affichage polymorphique. L'affichage justement sera traité de façon très spécifique

## notes

## résumé

0m 6s



## Problématiques abordées :

- ▶ conception POO, héritage, classes abstraites (prix, affichage)
- ▶ affichage polymorphique
- ▶ interfaces
- ▶ copie polymorphique



avec redéfinition de la méthode toString héritée de Object, et puis donc révision de la façon de faire un affichage qui s'adapte à chacune des classes, un affichage polymorphique. Nous aborderons également les interfaces, avec les différents mécanismes, mécanisme digital ou mécanisme analogique, à aiguille, voire les mécanismes doubles qui sont à la fois à aiguille et digital, et puis nous terminerons pas une dernière séquence vidéo qui présentera le problème non encore abordé dans le cours pour l'instant, de la notion de copie polymorphique, comment copier des collections hétérogènes,

## notes

## résumé

1m 13s



- ▶ Les *montres* sont des *produits* (que l'on peut vendre : ont un prix)
- ▶ Les montres ont un *mécanisme* de base et sont constituées de différents *accessoires* (boîtier, bracelet, ...)
- ▶ Les *produits* ont un prix dont le calcul peut varier, à partir d'une valeur de base
- ▶ Tous les produits sont « affichables », chacun à sa façon
- ▶ Les *mécanismes* et *accessoires* de montre sont aussi des produits (on pourrait en acheter séparément)
- ▶ Il existe trois sortes de *mécanismes* : *analogiques* (montre à aiguilles), *digitaux* et *doubles*.
- ▶ Pour les *mécanismes doubles*, on supposera ici qu'ils n'indiquent qu'une seule heure, mais se comportent sinon à la fois comme des *mécanismes analogiques* et comme des *mécanismes digitaux*

comment allons-nous copier les montres qui contiennent différents composants qui peuvent se comporter de façon différente, soit du point de vue du prix, soit du point de vue de l'affichage. Voilà donc le menu de cette dernière semaine. Mais commençons par présenter le problème un peu plus en détails, ce que nous voulons donc modéliser ce sont des montres, on va dire que des montres sont des produits, au sens qu'un produit est quelque chose que l'on peut vendre, qui a un prix. Les montres par ailleurs, auront des mécanismes de base, typiquement donc des aiguilles, pour pouvoir afficher l'heure, et seront constituées de différents accessoires, comme un boîtier, un bracelet, le verre sur la montre.. le fermoir. Les produits dont nous allons parler tout à l'heure ont un prix, et le calcul de ce prix peut varier. Donc cet aspect là est intéressant pour la conception, à partir d'une valeur de base qui doit être fixée au niveau du produit. Tous les produits sont affichables, et cet affichage peut aussi varier, chaque produit doit être affiché à sa façon propre. Bien sûr, ce qui est derrière ces notions de "peut varier" et "affichable à sa façon propre", c'est la notion de polymorphisme. Ensuite, les mécanismes dont nous avons parlé juste au dessus, et les accessoires dont sont constituées les montres, ces mécanismes et ces accessoires sont aussi des produits. On voit ici plusieurs fois l'emploi du verbe être, qui va nous qualifier donc des relations d'héritage. On pourrait par exemple acheter séparément des accessoires ou mécanismes, donc chacun pourrait se comporter comme un produit, donc chacun avoir son prix propre, et sa façon propre de calculer son prix. Il existe fondamentalement trois sortes de mécanisme, les mécanismes qu'on va qualifier d'analogique, pour représenter les montres à aiguilles, les montres digitales seront représentées par un mécanisme digital, et

notes

résumé

1m 47s



- ▶ Les *montres* sont des *produits* (que l'on peut vendre : ont un prix)
- ▶ Les montres ont un *mécanisme* de base et sont constituées de différents *accessoires* (boîtier, bracelet, ...)
- ▶ Les *produits* ont un prix dont le calcul peut varier, à partir d'une valeur de base
- ▶ Tous les produits sont « affichables », chacun à sa façon
- ▶ Les *mécanismes* et *accessoires* de montre sont aussi des produits (on pourrait en acheter séparément)
- ▶ Il existe trois sortes de *mécanismes* : *analogiques* (montre à aiguilles), *digitaux* et *doubles*.
- ▶ Pour les *mécanismes doubles*, on supposera ici qu'ils n'indiquent qu'une seule heure, mais se comportent sinon à la fois comme des *mécanismes analogiques* et comme des *mécanismes digitaux*

puis on aura des montres qui seront à la fois à aiguille et à la fois à affichage digital.  
Enfin pour ces mécanisme qu'on a qualifiés de mécanismes doubles,

notes

résumé



on supposera qu'ils n'indiquent qu'une seule heure, donc on va dire qu'il y aura une heure associée à une montre ici, et qu'elle aura deux façons de se représenter,

notes

---

---

---

---

---

---

---

---

---

---

résumé

3m 49s



---

---

---

---

---

---

---

---

---

---



- ▶ Les **montres** sont des produits (que l'on peut vendre : ont un prix)
- ▶ Les montres ont un **mécanisme** de base et sont constituées de différents **accessoires** (**boîtier**, **bracelet**, ...)
- ▶ Les *produits* ont un prix dont le calcul peut varier, à partir d'une valeur de base
- ▶ Tous les produits sont « affichables », chacun à sa façon
- ▶ Les *mécanismes* et *accessoires* de montre sont aussi des produits (on pourrait en acheter séparément)
- ▶ Il existe trois sortes de mécanismes : **analogiques** (montre à aiguilles), **digitaux** et **doubles**.
- ▶ Pour les *mécanismes doubles*, on supposera ici qu'ils n'indiquent qu'une seule heure, mais se comportent sinon à la fois comme des *mécanismes analogiques* et comme des *mécanismes digitaux*

c'est un point de vue choisi pour l'exercice, évidemment on aurait pu choisir un autre point de vue qui consistait à associer une heure à chacun des mécanismes de représentation, soit aiguille, soit digital, mais ce n'est pas le point de vue qu'on a pris ici, justement pour pouvoir introduire une problématique intéressante au niveau donc du codage de ces classes. Pour résumer, au niveau des classes, c'est-à-dire au niveau des types d'objets que nous aurons dans notre programme, nous allons avoir donc des montres, nous allons avoir des produits, nous allons avoir des mécanismes et des accessoires, les accessoires peuvent être des boîtiers, des bracelets, tous ceci seront des classes, au niveau des mécanismes, on peut avoir des mécanismes analogiques, des mécanismes digitaux, des mécanismes doubles. Voilà l'ensemble des classes. Au niveau des relations d'héritage, nous avons vu que les montres sont des produits, donc vont hériter de Produit, on a vu que les mécanismes analogiques digitaux et doubles sont des sortes de mécanismes, donc les trois classes ici, vont hériter naturellement de la classe Mécanisme.

notes

résumé

4m 0s





Nous avons aussi vu que les mécanismes et les accessoires sont également des produits. Tout ceci donc, va nous conduire à la modélisation hiérarchique suivante : nous avons donc tout en haut la notion de produit, les accessoires, les mécanismes, les montres sont des produits, par contre une montre aura un mécanisme et aura des accessoires, donc encapsulera ces classes là. Parmi les accessoires, nous aurons des boîtiers, des bracelets des fermoirs, des vitres, tout ceci sont des accessoires. Au niveau des mécanismes, les mécanismes analogiques et les mécanismes digitaux sont des mécanismes. Enfin, le mécanisme double, au départ, c'est un mécanisme, mais nous avons décidé par la dernière remarque précédente, qu'il n'indiquait qu'une seule heure, qu'on va certainement représenter au niveau de la notion de mécanisme, et qu'il se comporte à la fois comme des mécanismes analogiques et des mécanismes digitaux, d'où certainement la notion d'interface, mais nous verrons plus tard comment modéliser cela exactement. Regardons comment tout ceci se traduit au niveau du code.

notes

résumé

5m 1s



- Les *montres* sont des *produits*

```
class Produit {  
}  
  
// -----  
class Montre extends Produit {  
}  
  
// -----  
class Montres {  
    public static void main(String args[]) {}  
}
```

Nous vous conseillons d'ailleurs en même temps que vous suivez ces vidéos, de temps en temps faire des pauses, et écrire par vous-même le code dans votre environnement de développement habituel.

notes

résumé

6m 13s



- Les *montres* sont des *produits*

```
class Produit {  
}  
  
// =====  
class Montre extends Produit {  
}  
  
// =====  
class Montres {  
    public static void main(String args[]) {}  
}
```

Le premier point, les montres sont des produits, se traduit donc par une classe Montre, qui est un, qui hérite d'un produit,

notes

résumé

6m 22s



- ▶ Les *montres* sont des *produits*
- ▶ Les montres ont un *mécanisme* et sont constituées de différents *accessoires*

```
import java.util.ArrayList;
// ...

// =====
class Accessoire {
}

// =====
class Mecanisme {
}

// =====
class Montre extends Produit {
    private Mecanisme coeur;
    private ArrayList<Accessoire> accessoires;
}
```

et donc bien sûr on introduit une classe *Produit*, dont hérite *Montre*, et puis ensuite pour l'instant, on a simplement un main vide. Deuxième point, les montres ont un mécanisme,

notes

résumé

6m 32s



- ▶ Les *montres* sont des *produits*
- ▶ Les montres ont un *mécanisme* et sont constituées de différents *accessoires*
- ▶ Les *produits* ont un prix

```
class Produit {  
    private double prix;  
}  
  
// ...
```

donc elles vont encapsuler un mécanisme, les montres ont un mécanisme et sont constituées, donc ont différents accessoires. Donc comment on dit différents accessoires, c'est simplement une collection d'accessoires, un tableau dynamique d'accessoires. Et ceci introduit donc deux nouvelles classes. Une classe Mécanisme et une classe Accessoire. L'aspect suivant, les produits ont un prix. Donc ici tout simplement, on pourrait encapsuler un prix, naturellement représenté comme un double,

notes

résumé

6m 42s



- ▶ Les *montres* sont des *produits*
- ▶ Les montres ont un *mécanisme* et sont constituées de différents *accessoires*
- ▶ Les *produits* ont un prix dont le calcul peut varier, à partir d'une valeur de base

```
class Produit {  
    private double valeur;  
  
    public double prix()  
    { return valeur; }  
}  
  
// ...
```

mais la contrainte nous dit, dont le calcul peut varier, ce qui montre que l'on va devoir calculer le prix, par exemple plus tard, on décidera que le prix des montres, c'est le prix de son mécanisme et la somme des prix de ses accessoires, et donc un prix ne devient plus une donnée mais devient un traitement que l'on va représenter sous la forme d'une méthode ici. Elle renvoie naturellement à un prix, c'est naturellement un double,

notes

résumé

7m 17s



- ▶ Les *montres* sont des *produits*
- ▶ Les montres ont un *mécanisme* et sont constituées de différents *accessoires*
- ▶ Les *produits* ont un prix dont le calcul peut varier, à partir d'une valeur de base
- ▶ Tous les produits sont « affichables » chacun à sa façon
- ▶ Les *mécanismes* et *accessoires* sont aussi des produits

```
// ...

// =====
class Accessoire extends Produit {
}

// =====
class Mecanisme extends Produit {
}

// =====
class Montre extends Produit {
    private Mecanisme coeur;
    private ArrayList<Accessoire> accessoires;
}
```

et ensuite on veut dire que le prix peut varier, donc peut varier, ça veut dire que l'on peut avoir un comportement polymorphique. Disons donc, que de base, on va retourner le prix de base, qui sera qualifié ici, de valeur de base, et donc ce que l'on avait appelé avant prix, va devenir un support, une valeur de base, pour par défaut, être le prix du produit, mais peut-être possiblement plus tard, dans d'autres sous-classes de la classe Produit, être simplement utilisé dans des calculs différents du prix d'autres produits, sous-classes dérivées de la classe Produit. De plus, tous les produits sont affichables, c'est-à-dire concrètement qu'on va ajouter une méthode toString polymorphique, qui sera capable de nous donner une chaîne de caractère correspondant à l'affichage d'un produit. Les produits sont affichables mais chacun à sa façon, c'est-à-dire que donc, on va avoir un affichage polymorphique, on laisse ça pour la prochaine séquence vidéo, on y reviendra dans cette séquence vidéo, pour l'instant on se contente de mettre juste un entête pour se souvenir que l'on veut redéfinir la méthode toString pour les produits. Du côté des mécanismes et des accessoires,

notes

résumé

7m 37s





- ▶ ...
- ▶ Tous les produits sont « affichables » chacun à sa façon
- ▶ Les *mécanismes* et *accessoires* sont aussi des produits
- ▶ Il existe trois sortes de *mécanismes* : *analogiques*, *digitaux* et *doubles*
- ▶ Pour les *mécanismes doubles*, on supposera ici qu'ils n'indiquent qu'une seule heure, mais se comportent sinon à la fois comme des *mécanismes analogiques* et comme des *mécanismes digitaux*

```
// ...

// =====
class Mecanisme extends Produit {
}

// =====
class MecanismeAnalogique extends Mecanisme {
}

// =====
class MecanismeDigital extends Mecanisme {
}

// =====
class MecanismeDouble extends Mecanisme {
}

// ...
```

on a dit que les mécanismes et les accessoires sont des produits. Donc on a ici aussi une relation d'héritage, la classe Accessoire hérite de la classe Produit, qui était définie précédemment, et la classe Mecanisme hérite également de la classe Produit. Au niveau des mécanismes, il existe trois sortes de mécanismes, donc là aussi ces trois mécanismes sont des mécanismes bien sûr, donc nous avons mécanisme analogique qui est un mécanisme, qui hérite de Mecanisme, mécanisme digital, qui hérite de mécanisme, et mécanisme double, qui hérite de mécanisme.

notes

résumé

8m 49s





Et enfin le dernier point, pour les mécanismes doubles, on supposera qu'il n'existe qu'une seule heure au niveau du mécanisme double, mais que les mécanismes doubles se comportent comme des mécanismes analogiques, et des mécanismes digitaux, ce que nous n'avons pas du tout traduit ici, qui va donc remettre bien sûr en cause cette relation d'héritage, mais nous reviendrons sur ce point précis dans une vidéo spécifiquement dédiée à cet aspect là. Voilà, ceci conclut cette première séquence vidéo de présentation générale et de modélisation orienté objet du problème. La prochaine séquence vidéo va s'intéresser à l'affichage polymorphique et à remplir la classe produit. et à remplir la classe produit.

notes

résumé

9m 27s

