

Support de cours

Cours:

## Introduction à la programmation orientée objet (en C++)

Vidéo:

### W17-04-heritmultip-CPP-pt1

Concepts (extraits des sous-titres générés automatiquement) :

**Super-classe. Thématique de l'héritage multiple. Hiérarchie d'héritage. Nouvelle hiérarchie. Seule heure. Première hiérarchie. Fait conforme. Différents mécanismes. Séquences précédentes. Mécanismes analogiques. Niveau de la classe. Hiérarchie de sorte. Étude de cas. Mécanisme digitaux. Héritage multiple.**



[vers la recherche de séquences vidéo](#)

(dans Introduction à la programmation orientée objet (en C++).)



[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 : modélisation des mécanismes

## (Partie 1)

Introduction à la programmation orientée objet (en C++)

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s





Nous poursuivons notre étude de cas sur les montres

notes

---

---

---

---

---

---

---

---

---

---

résumé

0m 1s



---

---

---

---

---

## Rappel du problème

- ▶ Les montres ont un *mécanisme* de base [...]
- ▶ Les *mécanismes* [...] 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 : public Produit {
};

class MecanismeAnalogique : public Mecanisme
{};

class MecanismeDigital : public Mecanisme {
};

class MecanismeDouble : public Mecanisme {
};

// =====
class Montre : public Produit {
    // ...
private:
    unique_ptr<Mecanisme> coeur;
    // ...
};
```

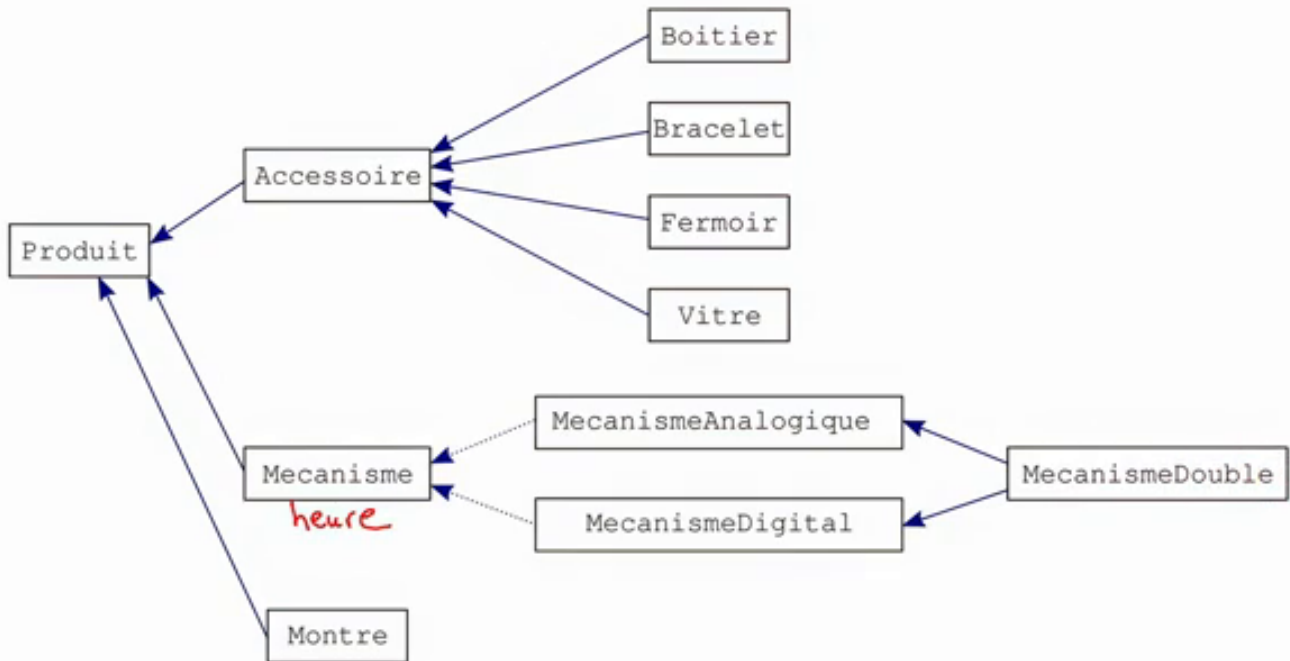
pour cette fois, aborder la thématique de l'héritage multiple au travers de la modélisation des différents mécanismes impliqués. Pour rappel, dans les séquences précédentes, la conception à laquelle nous étions parvenus modélisait la « Montre » en lui attribuant un cœur qui est un pointeur sur un « Mecanisme ». Une hiérarchie d'héritage avait été ébauchée pour les différents mécanismes impliqués. Donc nous avons une super-classe « Mecanisme » de laquelle héritaient trois types de mécanismes : le « MecanismeAnalogique », le « MecanismeDigital », et le « MecanismeDouble ». Cette première hiérarchie n'est cependant pas tout à fait conforme à nos souhaits concernant les « MecanismeDouble ». En effet dans les contraintes que nous nous étions fixées, nous souhaitions faire en sorte que les « MecanismeDouble » soient, à la fois, des mécanismes analogiques et des mécanisme digitaux, tout en ayant, tout en reportant une seule heure. Le fait de faire hériter « MecanismeDouble » de « Mecanisme » au lieu de le faire hériter de « MecanismeDigital » et de « MecanismeAnalogique » ne permet pas de modéliser proprement ce que nous souhaitons. Nous allons donc maintenant réviser notre hiérarchie de sorte à rendre apparent le lien

### notes

### résumé

0m 5s





qui lie les « MecanismeDouble » aux mécanismes digitaux et aux mécanismes analogiques. En C++ puisque l'héritage multiple est possible la voie est toute naturelle, il faut faire hériter « MecanismeDouble » à la fois, de « MecanismeAnalogique » et de « MecanismeDigital ». Si l'on met en place un héritage multiple de cette nature il faut être attentif à un point. Nous souhaitons qu'un « MecanismeDouble » ne reporte qu'une seule heure, héritée de plus haut, l'heure est modélisé au niveau de la classe « Mecanisme ». Si l'on se contente de mettre en place l'héritage multiple : « MecanismeDouble » héritant de « MecanismeAnalogique » et de « MecanismeDigital », tout instances de « MecanismeDouble »

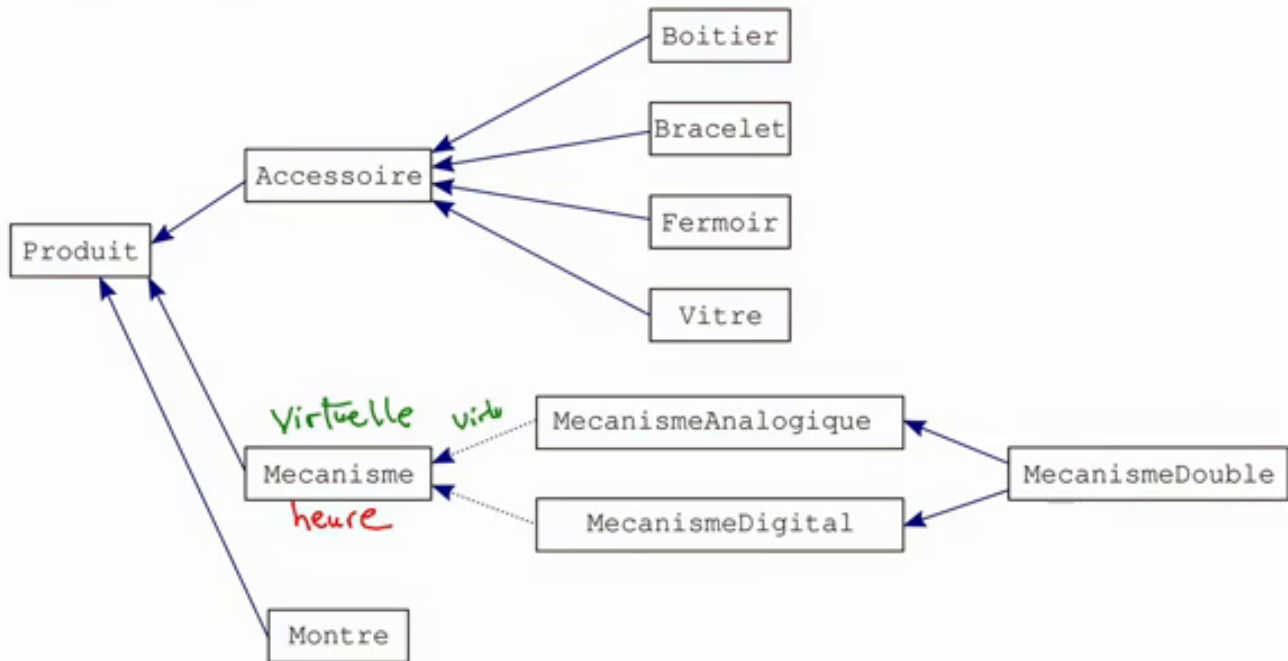
notes

résumé

1m 13s



## Hierarchie de classe



héritera de deux fois l'attribut « heure ». Une fois, en tant que mécanisme Analogique et une seconde fois en tant que mécanisme digital, Or, ce n'est pas ce que nous souhaitons nous voulons qu'un « MecanismeDouble » ne reporte qu'une seule heure. Il faut donc faire en sorte que cette super-classe soit virtuelle de sorte à ce que la sous-classe « MecanismeDouble » n'hérite qu'une seule fois de l'attribut « heure » et pour ceci il faut que tous les liens d'héritages liant « Mecanisme » à ses sous-classes directes soit déclarés comme virtuels.

### notes

### résumé

1m 49s



```
class Mecanisme : public Produit {
private:
    string heure;
};

class MecanismeAnalogique : virtual public Mecanisme {
private:
    int date;
};

class MecanismeDigital : virtual public Mecanisme {
private:
    string reveil;
};

class MecanismeDouble : public MecanismeAnalogique
                        , public MecanismeDigital
{
};
```

La nouvelle hiérarchie ainsi esquissée correspond en fait au code que vous avez ici sous les yeux, nous avons donc, une super-classe « Mecanisme » qui va fournir l'attribut indiquant l'heure, de cette super-classe « Mecanisme » vont hériter deux sous-classes directes : « MecanismeAnalogique » et « MecanismeDigital ». Nous prenons donc le soin de déclarer les liens d'héritages comme virtuelle afin de permettre à la sous-classe « MecanismeDouble » de n'hériter qu'une seule fois de l'attribut provenant de « Mecanisme ». Étant entendu ici que « MecanismeDouble » hérite à la fois de « MecanismeAnalogique » et de « MecanismeDigital ». Pour rendre notre exemple un peu plus intéressant nous allons doter nos sous-classes « MecanismeAnalogique » et « MecanismeDigital » d'attributs spécifiques comme par exemple une date pour le mécanisme analogique et un réveil pour le mécanisme digital. Pour rappel, en C++, une super-classe dont les sous-classes héritent virtuellement et dites « classes virtuelles » à ne pas confondre avec une classe abstraite. avec une classe abstraite.

## notes

## résumé

2m 23s

