

Support de cours

Cours:

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

Vidéo:

### W17-01-pbgeneral-CPP-pt2

Concepts (extraits des sous-titres générés automatiquement) :

**Prix de son mécanisme. Côté des mécanismes. Question de leur copie. Prochaine séquence vidéo. Problème de côté. Somme des prix de ses accessoires. Niveau des mécanismes. Copie de surface. Constructeur de copie. Prix des montres. Prix d'un produit. Copie profonde. Relation d'héritage. Séquence vidéo. Valeur de base.**



[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 : présentation et modélisation du problème

(Partie 2)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s



## Du problème au premier code

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



```
// ...

// =====
class Accessoire {
};

// =====
class Mecanisme {
};

// =====
class Montre : public Produit {
private:
    unique_ptr<Mecanisme> coeur;
    vector<unique_ptr<Accessoire>> accessoires;

    Montre(const Montre&)      = delete;
    Montre& operator=(Montre) = delete;
};
```

dès qu'on a des pointeurs, on doit en effet

notes

résumé

0m 1s



## Du problème au premier code

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

```
// ...

// =====
class Accessoire {
};

// =====
class Mecanisme {
};

// =====
class Montre : public Produit {
private:
    unique_ptr<Mecanisme> coeur;
    vector<unique_ptr<Accessoire>> accessoires;

    Montre(const Montre&) = delete;
    Montre& operator=(Montre) = delete;
};
```

se poser la question de leur copie, copie de surface, copie profonde, laissons ce problème de côté, ça sera abordé dans la dernière séquence vidéo de cette semaine, mais pour l'instant, contentons-nous donc de dire, que l'on ne peut pas copier de montre, ni affecter de montre, ni utiliser donc l'opérateur d'affectation, l'opérateur égal.

notes

résumé

0m 6s



## Du problème au premier code

- ▶ ...
- ▶ 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 : public Produit {
};

// =====
class MecanismeAnalogique : public Mecanisme
{};

// =====
class MecanismeDigital : public Mecanisme {
};

// =====
class MecanismeDouble : public Mecanisme {
};

// ...
```

Donc ici, c'est le constructeur de copie, « Montre » dans la classe « Montre », est bien un constructeur, ici constructeur de copie que l'on supprime, et puis donc l'opérateur égal, que l'on supprime aussi, on reviendra donc sur ce point plus tard. L'aspect suivant : les produits ont un prix. Donc ici tout simplement on pourrait encapsuler un prix naturellement représenté comme un « double », mais la contrainte nous dit, « dont le calcul peut varier », ce qui montre que l'on va devoir donc 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. Calculer le prix d'un produit ne fait pas modifier ce produit en tant que tel, donc la méthode va être qualifiée de « const », elle renvoie naturellement à un prix, c'est naturellement un « double ». Et ensuite, on veut dire que le prix peut varier, donc « peut varier », ça veut dire que l'on veut avoir ici un comportement polymorphique. Disons 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 simplement 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 dérivés de la classe « Produit ». De plus, tous les produits sont affichables. C'est-à-dire, concrètement, en C++, qu'on va surcharger l'opérateur d'affichage pour les produits. Les produits sont affichables

### notes

### résumé

0m 25s



## Du problème au premier code

- ▶ ...
- ▶ 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 : public Produit {
};

// =====
class MecanismeAnalogique : public Mecanisme
{};

// =====
class MecanismeDigital : public Mecanisme {
};

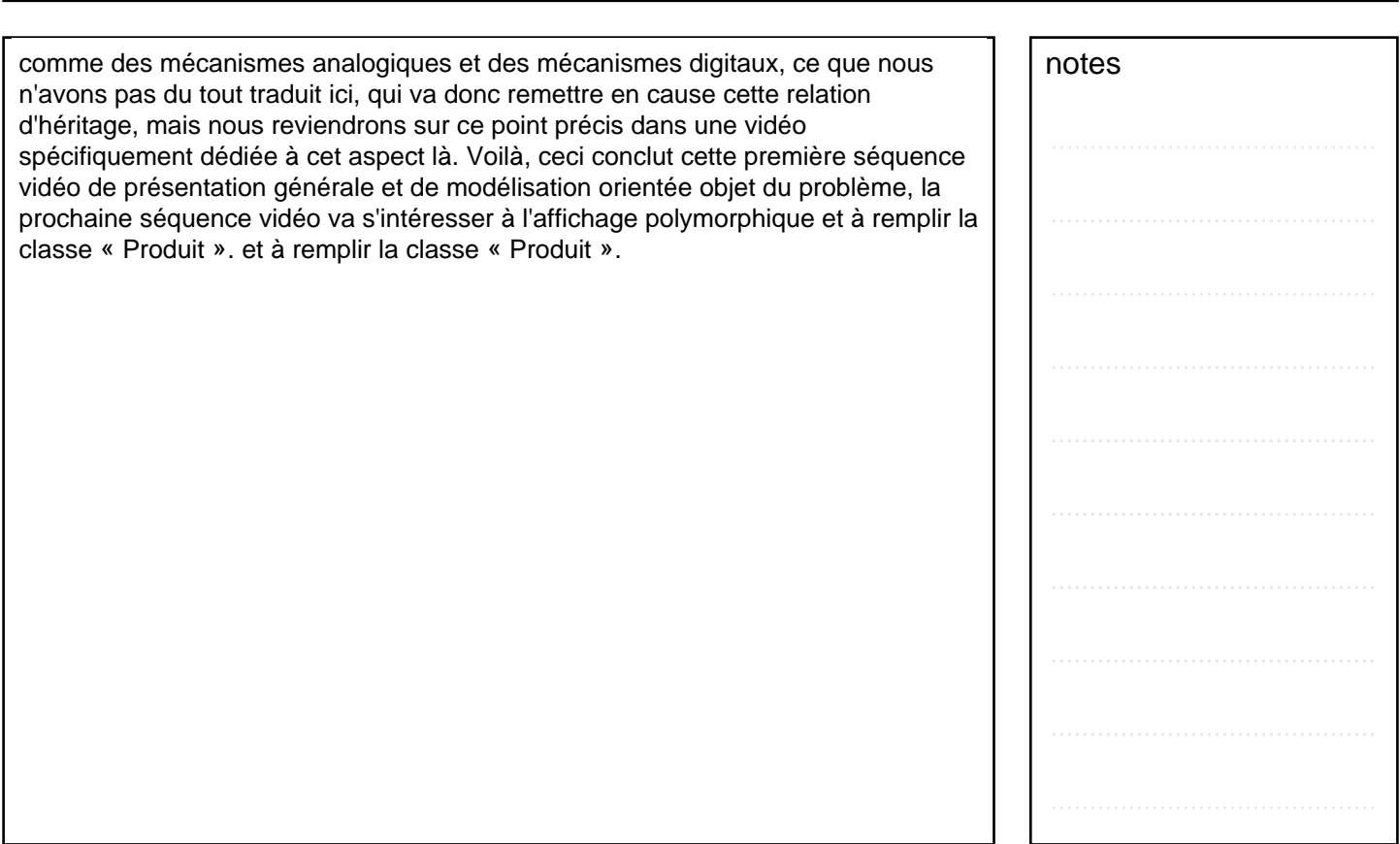
// =====
class MecanismeDouble : public Mecanisme {
};

// ...
```

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 donc juste un prototype pour se souvenir que l'on veut surcharger l'opérateur d'affichage pour les produits. Du côté des mécanismes et des accessoires, 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 « MecanismeAnalogique » qui est un mécanisme, qui hérite de « Mecanisme », « MecanismeDigital » qui hérite de « Mecanisme », et « MecanismeDouble » qui hérite de « Mecanisme ». 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

### notes

### résumé



3m 13s

