

Support de cours

Cours:

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

Vidéo:

W15-03-polymasq-CPP-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Méthode virtuelle. Nouveaux mots clé. Super-classe. Substitution future de la méthode. Différents aspects. Classe a. Méthode f2. Premier temps. Qualificatif const. Mot cle override. Mots-clé. Mot-clé final. Différentes méthodes. Substitution de la méthode. Classe b.



[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/>

Masquage, substitution et surcharge

(Partie 3)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s





Voilà ! Tout ceci est certainement très technique et dans un premier temps je pense un petit peu difficile. Vous reviendrez sur cette séquence plus tard lorsque vous aurez

notes

résumé

0m 1s



En C++11, le programmeur *peut* (optionnel) indiquer ses intentions lors de la (re)déclaration d'une méthode :

- ▶ avec le qualificatif `override` pour dire qu'il pense substituer/redéfinir une méthode virtuelle
- ▶ avec le qualificatif `final`, empêcher la substitution/redéfinition future d'une méthode virtuelle.

un petit peu plus pratiquer le polymorphisme et ses différents aspects : masquage, substitution et surtout surcharge qui elle, à mon avis, ne devrait plus poser de problème. Justement, c'est parce que ces aspects sont un petit peu difficiles que C++ 2011 a introduit deux nouveaux mots clé pour permettre au programmeur, s'il le souhaite, de plus clairement spécifier ses intentions.

notes

résumé

0m 13s



```

class A {
    // ...
    virtual void f1();
    virtual void f2() const;
        void f3();          // non virtuelle (oubli?)
    virtual void f4() final; // pas de redéfinition
};

class B: public A {
    // ...
    virtual void f1() override; // OK
    virtual void f1() override; // Erreur
    virtual void f2() override; // Erreur
        void f3() override; // Erreur: non virtuelle
    virtual void f4();          // Erreur : f4 était final
};

```

Ces mots-clé sont respectivement override qui permet d'indiquer que l'on pense substituer une méthode virtuelle héritée d'une super-classe ; et puis le mot-clé final qui empêche toute substitution future de la méthode en question dans les futurs possibles sous-classes. Voyons tout ceci en détail sur un exemple. Supposons donc que l'on ait une classe A qui contient ici 4 méthodes, une méthode f1 définie simplement comme virtuelle, une méthode f2 qui est aussi virtuelle et qui a le qualificatif const, qui ne modifie pas l'instance, une méthode f3, qui elle n'est pas virtuelle, et la méthode f4 sur laquelle on rajoute le qualificatif final. Puis nous avons donc une classe B qui hérite de A et qui bien sûr va redéfinir, substituer, masquer ces différentes méthodes. Nous avons donc une substitution de la méthode f1 dans la classe B c'est bien le même prototype exactement. Donc le mot cle override est correctement utilisé. En effet, f1 est bien une substitution de cette méthode. Là, nous avons donc une autre méthode. Je vous laisse deviner de quoi il s'agit. Je vous laisse deviner de quoi il s'agit.

notes

résumé

0m 37s

