

Support de cours

Cours:

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

Vidéo:

W11-04-resume-CPP-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Utilisateur externe de la classe. Monde extérieur. Aspect d'implémentation interne. Séquences vidéos précédentes. Particularités d'une classe. Éléments fondamentaux. Classe rectangle. Classe. Moyen d'un attribut hauteur. Interface d'utilisation. Nombre d'exemples. Détail d'implémentation. Nombre de données spécifiques. Super structure. Principes d'une bonne encapsulation.



[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/>

Encapsulation et abstraction : résumé

(Partie 1)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s





Dans les séquences vidéos précédentes, vous avez appris à écrire une classe en c++ vous savez définir des méthodes et des attributs, et vous savez mettre en place un certain nombre de fondamentaux liés à l'encapsulation. A savoir définir ce qui est publiquement accessible pour un utilisateur externe de la classe, et ce qui ne l'est pas. Nous allons étudier comment tout cela se met en place concrètement,

notes

résumé

0m 1s



Pour résumer à ce stade, une classe est une `struct`

- ▶ qui contient aussi des fonctions (« méthodes »)
- ▶ dont certains champs (internes) peuvent être cachés (`private:`)
- ▶ et dont d'autres constituent l'interface (`public:`)

sur un certain nombre d'exemples. Et nous allons voir que bien encapsuler nécessite de prendre un certain nombre de précautions.

notes

résumé

0m 25s

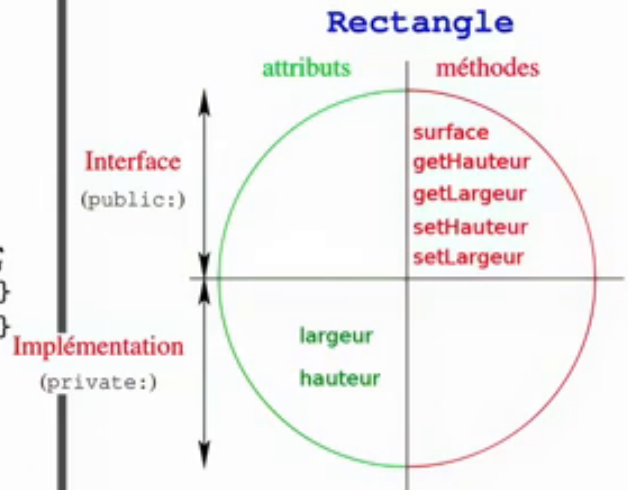


Un exemple complet de classe (1/2)

```
#include <iostream>
using namespace std;

// définition de la classe
class Rectangle {
public:
    // définition des méthodes
    double surface() const { return hauteur * largeur; }
    double getHauteur() const { return hauteur; }
    double getLargeur() const { return largeur; }
    void setHauteur(double h) { hauteur = h; }
    void setLargeur(double l) { largeur = l; }

private:
    // déclaration des attributs
    double hauteur;
    double largeur;
};
```



Pour rappel, à ce stade, une classe n'est autre qu'une super structure, dans lequel vous pouvez mettre non seulement des champs, que l'on appelle des attributs, mais aussi des fonctions, que l'on appelle désormais des méthodes. Nous avons vu, qu'une des particularités d'une classe, par rapport aux structures, c'est qu'il est possible de définir ce qui ne doit pas être accessible au monde extérieur, ce qui est privé, et ce qui est à contrario, accessible, c'est-à-dire public. Voici donc une implémentation possible de la fameuse classe Rectangle, qui nous a servi d'exemple introductif aux concepts fondamentaux de l'orienté objet. Donc lorsque l'on veut définir une classe, on utilise le mot réservé "classe", suivi du nom que l'on a choisi pour la classe. Comme désormais, ce nom va être un nom de type dans le programme, il est l'usage de le nommer en commençant par une majuscule. Une classe est usuellement caractérisée par un certain nombre de données spécifiques que l'on appelle les attributs. Ici, nous avons choisi de caractériser la classe Rectangle au moyen d'un attribut hauteur et largeur, que nous avons choisi d'implémenter sous la forme de deux doubles. Une classe est également caractérisée par un certain nombre de fonctionnalités spécifiques, que l'on appelle les méthodes. Une fois que l'on a établi les éléments fondamentaux, que sont les attributs et les méthodes, il faut se pencher sur la question de ce qui constitue parmi ces éléments, un aspect d'implémentation interne qui n'a pas besoin d'être connu du monde extérieur, et ce qui constitue l'interface d'utilisation, ce que le monde extérieur aura besoin de connaître et ce qu'il peut utiliser de la classe en question. Ici, nous avons choisi d'offrir comme fonctionnalité au monde extérieur, un certain nombre de méthodes permettant de consulter la valeur des attributs largeur et hauteur, de les modifier et de faire un calcul de surface, qui utilise cette

notes

résumé

0m 32s

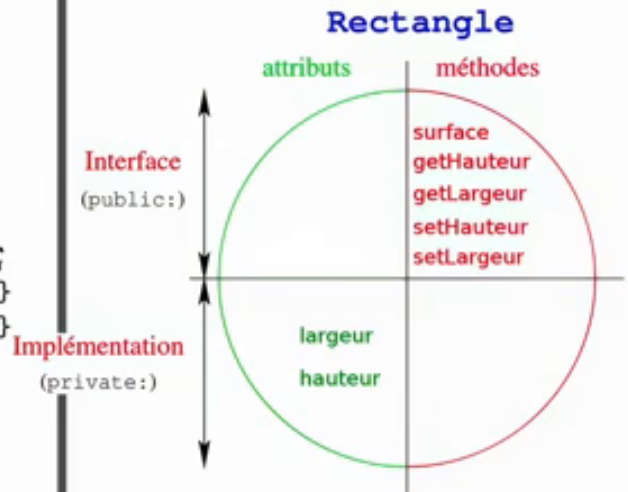


Un exemple complet de classe (1/2)

```
#include <iostream>
using namespace std;

// définition de la classe
class Rectangle {
public:
    // définition des méthodes
    double surface() const { return hauteur * largeur; }
    double getHauteur() const { return hauteur; }
    double getLargeur() const { return largeur; }
    void setHauteur(double h) { hauteur = h; }
    void setLargeur(double l) { largeur = l; }

private:
    // déclaration des attributs
    double hauteur;
    double largeur;
};
```



notes

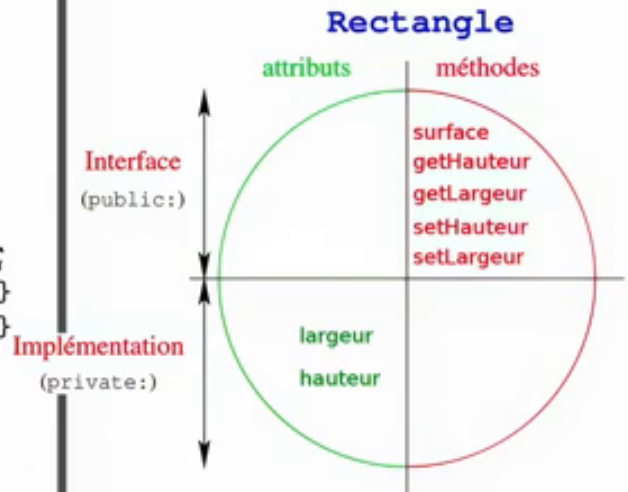
résumé

Un exemple complet de classe (1/2)

```
#include <iostream>
using namespace std;

// définition de la classe
class Rectangle {
public:
    // définition des méthodes
    double surface() const { return hauteur * largeur; }
    double getHauteur() const { return hauteur; }
    double getLargeur() const { return largeur; }
    void setHauteur(double h) { hauteur = h; }
    void setLargeur(double l) { largeur = l; }

private:
    // déclaration des attributs
    double hauteur;
    double largeur;
};
```



que vous avez sous les yeux, par rapport à ce point ? par rapport à ce point ?

notes

résumé