

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W11-04-resume-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Travers d'exemples. Notion de rectangle. Détails d'implémentation. Utilisateur externe. Bons principes d'encapsulation. Classe rectangle. Partie de l'interface d'utilisation. Exemple concret. Aspects d'encapsulation. Séquences vidéos précédentes. Programmeur de la classe rectangle. Classe. Façon générale. Exemple introductif. Nombre de précautions.



[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/>

Encapsulation et abstraction : résumé

(Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Dans les séquences vidéos précédentes, vous avez appris à écrire une classe en java. Vous savez y définir des méthodes et des attributs. Vous savez également mettre en place un certain nombre de fondamentaux lié à l'encapsulation, à savoir, définir ce qui dans votre classe est publiquement accessible et ce qui ne l'est pas. Nous allons aujourd'hui examiner tout cela, au travers d'exemples Et nous allons voir que bien encapsuler une classe nécessite

notes

résumé

0m 1s



Pour résumer à ce stade, une classe permet de définir un nouveau type caractérisé par :

- ▶ des attributs (des données spécifiques)
- ▶ des méthodes (« fonctions »)
- ▶ dont certains attributs et méthodes (internes) peuvent être cachés (`private`)
- ▶ et dont d'autres constituent l'interface (`public`)

un certain nombre de précautions.

notes

résumé

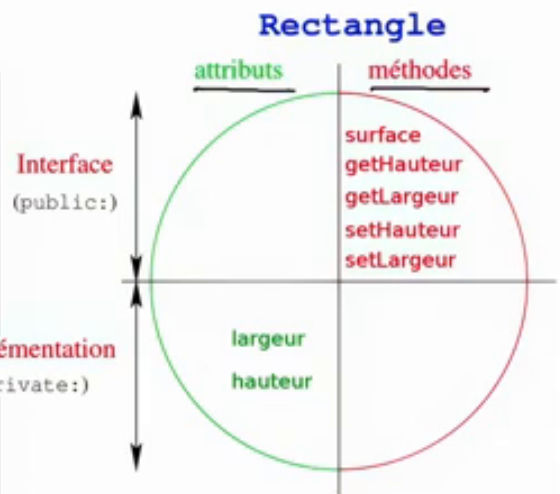
0m 25s



Un exemple complet de classe (1/2)

```
// définition de la classe
class Rectangle {
    // déclaration des attributs
    { private double hauteur; double [] dims;
      private double largeur;

    // définition des méthodes
    { public double surface()
      { return hauteur * largeur; }
      public double getHauteur() { return hauteur; }
      public double getLargeur() { return largeur; }
      public void setHauteur(double h)
      { hauteur = h; }
      public void setLargeur(double l)
      { largeur = l; }
    }
}
```



A ce stade, définir une classe dans un programme c'est définir un nouveau type caractérisé par des attributs, des méthodes et vous allez encapsuler cette classe en définissant ce qui est privé, détails d'implémentation, ce qui est public, ce qui fait partie de l'interface d'utilisation que vous prévoyez pour un utilisateur externe. Il est important de regarder maintenant comment tout cela s'articule sur un exemple concret et c'est l'objectif de la séquence qui suit. Si l'on veut permettre à un programme de travailler avec la notion de rectangle, il va falloir commencer par faire un travail de conception, concevoir la classe rectangle la déterminer en termes des attributs qui la caractérisent et des méthodes qui lui sont spécifiques ; également réfléchir à ce qui constitue l'interface d'utilisation et ce qui constitue la partie cachée des détails d'implémentation. Voici donc une implémentation possible de la classe rectangle qui nous a servi d'exemple introductif à la programmation d'un tel objet. Si je veux définir la classe dans un programme, j'utilise le mot réservé classe suivi du nom que j'ai choisi pour la classe avec la petite convention qu'un nom de classe commence par une majuscule en java. Ensuite, nous nous sommes intéressés à définir quelles étaient les attributs spécifiques de la classe ainsi que les méthodes. Nous nous sommes intéressés à des aspects d'encapsulation, à savoir, définir ce qu'est un détail d'implémentation, typiquement l'implémentation des attributs, et ce qui fait partie de l'interface, ce que l'utilisateur externe pourra utiliser et que l'on qualifie de public ici. La classe rectangle qui est ici sous vos yeux, va mettre à disposition de l'utilisateur externe un certain nombre de fonctionnalités qui permettent de consulter les attributs d'en modifier la valeur, et de calculer la surface du rectangle. Notez bien que de façon générale, il n'est pas nécessaire pour une classe de définir

notes

résumé

0m 28s



Un exemple complet de classe (1/2)

```
// définition de la classe
class Rectangle {
    // déclaration des attributs
    { private double hauteur; double [] dims;
      private double largeur;

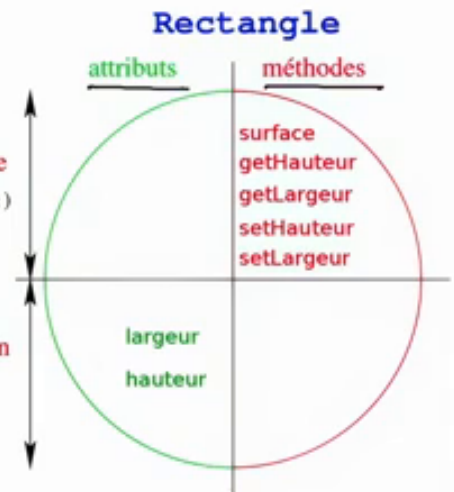
    // définition des méthodes
    { public double surface()
      { return hauteur * largeur; }
      public double getHauteur() { return hauteur; }
      public double getLargeur() { return largeur; }
      public void setHauteur(double h)
      { hauteur = h; }
      public void setLargeur(double l)
      { largeur = l; }
    }
}
```

double [] dims;

dims[0]

Interface
(public:)

Implémentation
(private:)



des méthodes get et set pour chacun des attributs. Il ne s'agit ici que d'un exemple parce que nous avons, ici, à ce stade une façon très basique d'interagir avec la classe. Nous avons essayé de respecter quelques bons principes d'encapsulation, encapsulation, qui veut dire que nous ne permettons à l'utilisateur externe d'accéder aux fonctionnalités qu'à travers l'interface. Ceci veut dire que désormais lorsqu'on est programmeur de la classe rectangle, on a la latitude de pouvoir modifier l'implémentation interne, par exemple, ici, je peux décider de modifier la représentation, les dimensions en utilisant plutôt un tableau à deux éléments qui me permettrait de stocker la hauteur et la largeur. A ce moment là, c'est à moi, programmeur de la classe rectangle d'adapter les méthodes par exemple, au lieu d'avoir return hauteur, nous aurions un return dims[0]. C'est à moi d'adapter ces méthodes de sorte à ce que celui qui utilise la classe ne soit nullement impacté. Il pourra toujours continuer à utiliser getHauteur, setHauteur et calcul de surface sans avoir connaissance des modifications que j'ai fait en interne sur la classe. Notez également que puisqu'on contraint l'utilisateur à passer par un certain nombre de méthodes prédéfinies pour accéder au contenu d'un objet, il devient possible de mettre en oeuvre un certain nombre de protections, de garde-fous pour empêcher des manipulations incorrectes de ce contenu. Voyez-vous ce que l'on pourrait améliorer à cette classe dans ce sens ? à cette classe dans ce sens ?

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

.....

.....

.....

.....

.....