

Support de cours

Cours:

## Introduction à la programmation orientée objet (en Java)

Vidéo:

### W11-01-intropoo-JAVA-pt2

Concepts (extraits des sous-titres générés automatiquement) :

**Représentation du rectangle. Niveau de l'utilisation. Fonctionnalité du calcul de surface. Nouveaux types de données. Travers de la notion de classe. Détail d'implémentation interne. Interface d'utilisation. Unique entité des données. Nombre de rectangles. Programmeur utilisateur. Programmation orienté. Calcul de surface. Niveaux de perception d'un objet. Niveau externe. Types de données.**



[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/>

# Introduction

## (Partie 2)

### Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



## Notions d'encapsulation

### Principe d'encapsulation :

regrouper dans le même objet informatique («concept»), les données et les traitements qui lui sont *spécifiques* :

- ▶ **attributs** : les données incluses dans un objet
- ▶ **méthodes** : les fonctions (= traitements) définies dans un objet

Les objets sont définis par leurs attributs et leurs méthodes.



L'encapsulation, c'est pouvoir regrouper dans une unique entité des données et les traitements qui agissent sur ces données. Donc typiquement dans cet exemple, nous allons regrouper dans une seule et même entité, la largeur et la hauteur, qui caractérise la représentation du rectangle, et la fonctionnalité du calcul de surface. Donc en terme de jargon, on va parler pour les données, de la notion d'attributs. Et on va parler pour les fonctionnalités, de la notion de méthode. Donc en programmation orienté objet, on va être dans la possibilité de définir des nouveaux types de données, au travers de la notion de classe, nous allons le voir plus tard. Ces types de données peuvent être utilisés pour travailler concrètement avec des données,

notes

résumé

0m 1s



## Notion d'abstraction

Pour être véritablement intéressant, un objet doit permettre un certain degré d'**abstraction**.

Le processus d'abstraction consiste à identifier pour un ensemble d'éléments :

- ▶ des caractéristiques communes à tous les éléments
- ▶ des mécanismes communs à tous les éléments

description **générique** de l'ensemble considéré :  
*se focaliser sur l'essentiel, cacher les détails.*

$\left\{ \begin{array}{l} \text{double hauteur}_1 = 2.0; \\ \text{double largeur}_1 = 1.0; \end{array} \right.$

$\left\{ \begin{array}{l} \text{double hauteur}_2 = 4.5; \\ \text{double largeur}_2 = 2.5; \end{array} \right.$

$\left\{ \begin{array}{l} \text{Surface (hauteur}_1, \text{largeur}_2); \\ \text{Surface (hauteur}_2, \text{largeur}_1); \end{array} \right.$

de type plus abstrait, le rectangle. Et ces données vont être des objets, qui vont cohabiter dans le programme et interagir dans le programme. Un programme orienté objet va donc travailler avec des objets qui sont caractérisés par leurs attributs et leurs méthodes. Parlons un peu de la notion d'abstraction. Imaginez que je souhaite un programme qui manipule un certain nombre de rectangles, pas uniquement un seul. Donc dans une approche procédurale, je devrais déclarer autant de variables hauteur et largeur, que j'ai de rectangles. Donc ceci pour le premier. Et puis je devrais faire exactement la même chose pour le second rectangle. Donc on voit que c'est relativement fastidieux. Imaginez par exemple, que je travaille maintenant avec des rectangles en trois dimensions où, j'ai une hauteur et une largeur, mais aussi une profondeur, et bien je me retrouverais avec six variables à devoir initialiser, de façon appropriée. Pour mon calcul de surface, pour chaque rectangle, je devrais invoquer ma méthode de calcul de surface, en passant à chaque fois les bons arguments. Donc on voit rapidement que c'est fastidieux. Et puis c'est source d'erreurs. Imaginez que je me trompe, et qu'ici je fasse un calcul de surface qui implique hauteur1 et largeur2, et je perds la cohérence, je n'ai plus le bon calcul de surface pour chacun de mes rectangles. Ce mécanisme, le processus de l'abstraction, est celui par lequel je me rends compte, qu'effectivement, pour chacun des rectangles qui apparaît dans mon programme, j'ai la même représentation générique, par une largeur et une hauteur,

notes

résumé

1m 1s



## Notion d'abstraction

Pour être véritablement intéressant, un objet doit permettre un certain degré d'**abstraction**.

Le processus d'abstraction consiste à identifier pour un ensemble d'éléments :

- ▶ des caractéristiques communes à tous les éléments
- ▶ des mécanismes communs à tous les éléments

description **générique** de l'ensemble considéré :  
se focaliser sur l'essentiel, cacher les détails.

$\left\{ \begin{array}{l} \text{double hauteur}_1 = 20; \\ \text{double largeur}_1 = 10; \end{array} \right\}$  rect<sub>1</sub>

$\left\{ \begin{array}{l} \text{double hauteur}_2 = 4.5; \\ \text{double largeur}_2 = 2.5; \end{array} \right\}$  rect<sub>2</sub>

$\left\{ \begin{array}{l} \text{Surface (hauteur}_1, \text{largeur}_2); \text{ rect}_1.\text{Surface}() \\ \text{Surface (hauteur}_2, \text{largeur}_2); \end{array} \right\}$

Rectangle

hauteur  
largeur  
surface,

j'ai un même calcul de surface qui intervient, du coup je vais pouvoir travailler avec une notion beaucoup plus abstraite, qui est celle de rectangle. Le mécanisme de l'encapsulation va me permettre de regrouper dans la notion de rectangle, tout ce qui est nécessaire à le modéliser. Du coup, je vais permettre à mon programme de désormais pouvoir travailler avec des entités plus abstraites, avec un premier rectangle et un deuxième rectangle, tous les deux de type Rectangle. Je vais invoquer des calculs de surface sur ces rectangles. On va voir qu'en poo, j'anticipe un peu, je calcule la surface du premier rectangle. Donc on va voir par la suite, comment tout ceci s'exprime de façon beaucoup plus précise. Du coup, je permets à mon programme de se focaliser sur l'essentiel. Je ne suis plus en train de me préoccuper du fait qu'un rectangle est constitué d'une largeur et d'une hauteur, je me focalise sur les aspects essentiels. Je travaille avec des rectangles, je fais un calcul de surface sur un rectangle.

notes

résumé

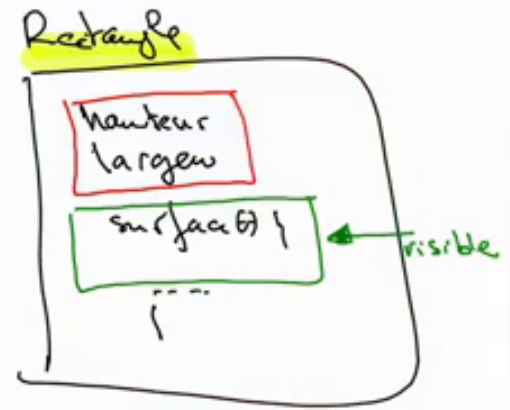
3m 1s



## Abstraction et Encapsulation

En plus du regroupement des données et des traitements relatifs à une entité, l'encapsulation permet en effet de définir **deux niveaux** de perception des objets :

- ▶ **niveau externe** : partie « visible » (par les programmeurs-utilisateurs) :
  - ▶ **l'interface** : *entête* de quelques méthodes bien choisies
  - ☞ résultat du processus d'*abstraction*
- ▶ **niveau interne** : (détails d')**implémentation**
  - ▶ **corps** :
    - ▶ méthodes et attributs accessibles uniquement depuis l'intérieur de l'objet (ou d'objets similaires)
    - ▶ définition de toutes les méthodes de l'objet



Au niveau de l'utilisation, je ne vois désormais du rectangle, que ce que l'on appelle en jargon orienté objet, son interface d'utilisation, c'est-à-dire les fonctionnalités qui sont offertes pour la manipulation du rectangle, comme ici le calcul de surface. Faisons une analogie avec un objet de la vie courante. En conduisant votre voiture, vous n'avez besoin d'en connaître que l'interface d'utilisation. Concrètement, vous avez besoin d'avoir un volant, un accélérateur, une pédale de frein et à nul moment, il ne vous est utile de savoir comment le moteur est concrètement construit. Donc, pour conduire votre voiture, nous n'avons besoin d'en connaître que l'interface d'utilisation. C'est exactement le même principe en poo. Pour utiliser un nouveau type, une nouvelle classe, vous n'avez besoin d'en connaître que l'interface d'utilisation qui est prévue par son programmeur. Le reste, détail d'implémentation interne, n'est pas utile à connaître. Il y a donc deux niveaux de perception d'un objet. Un niveau externe, qui est utile, au programmeur utilisateur, celui qui utilise la notion de rectangle, dans un programme. Donc il existe désormais un type Rectangle, je peux déclarer des variables de type Rectangle, les initialiser de façon appropriée. Et ensuite, ce qui m'intéresse, en tant que programmeur utilisateur, c'est les fonctionnalités utiles de calcul de surface. Donc ce niveau externe, est le niveau qui intéresse le programmeur utilisateur, celui qui utilise le type Rectangle. Second niveau : le niveau interne, celui qui a programmé le nouveau type, le type Rectangle, a dû se préoccuper de tous les détails d'implémentation. À savoir qu'un rectangle, c'est une hauteur et une largeur, il a dû définir comment se fait concrètement le calcul de surface. Donc ceci constitue la partie implémentation, à un niveau interne, pas forcément utile, à celui qui utilise le rectangle. Donc en programmation orienté objet, nous aurons non

### notes

### résumé

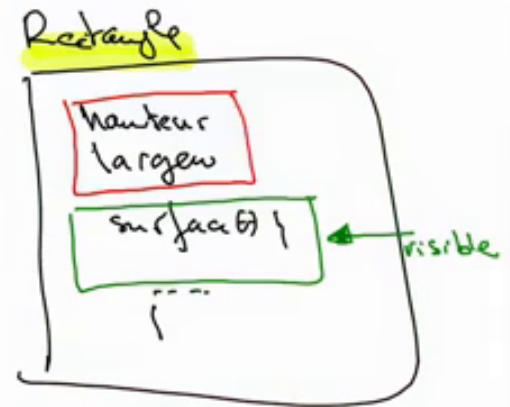
4m 13s



## Abstraction et Encapsulation

En plus du regroupement des données et des traitements relatifs à une entité, l'encapsulation permet en effet de définir **deux niveaux** de perception des objets :

- ▶ **niveau externe** : partie « *visible* » (par les programmeurs-utilisateurs) :
  - ▶ l'**interface** : *entête* de quelques méthodes bien choisies
  - ▶ résultat du processus d'*abstraction*
- ▶ **niveau interne** : (détails d')**implémentation**
  - ▶ **corps** :
    - ▶ méthodes et attributs accessibles uniquement depuis l'intérieur de l'objet (ou d'objets similaires)
    - ▶ définition de toutes les méthodes de l'objet



notes

résumé