

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W12-01-constrintro-JAVA-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Classe rectangle. Exemple de départ. Listes de paramètres différents. Déclaration usuelle. Nom de la classe. Certaine classe. Classe. Extérieur de la classe. Seul constructeur unique. Constructeurs possibles. Différents paramètres nécessaires. New rectangle. Appel du constructeur. Classe exemple. Différents accesseurs.



[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/>

Constructeurs (introduction)

(Partie 3)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



Les constructeurs (2)

Les constructeurs sont des méthodes presque comme les autres. Les différences sont :

- ▶ *pas* de type de retour (pas même `void`)
- ▶ *même nom* que la classe
- ▶ invoqués *systématiquement* à chaque fois qu'une instance est créée.

```
Rectangle(double h, double l)
{
    hauteur = h;
    largeur = l;
}
```

Comme les autres méthodes :

- ▶ les constructeurs peuvent être surchargés

(exemples dans la suite)

Une classe peut donc avoir **plusieurs constructeurs**, pour peu que leur liste de paramètres soit différente.

Voilà pour les différences. Mais les constructeurs peuvent,

notes

résumé

0m 1s





comme les autres méthodes, être aussi surchargés. C'est-à-dire qu'on va pouvoir avoir plusieurs listes de paramètres différents pour différents constructeurs. Ce qui fait que l'on a donc plusieurs constructeurs. Nous allons revenir sur ce point là dans la suite, mais nous voulions déjà insister ici sur le fait que

notes

résumé

0m 5s



```
class Rectangle {  
    private double hauteur;  
    private double largeur;  
  
    public Rectangle(double h, double l)  
    {  
        hauteur = h;  
        largeur = l;  
    }  
    public double surface()  
    { return hauteur * largeur; }  
    // accesseurs/modificateurs si nécessaire  
    // ...  
}
```

l'on parle donc des constructeurs possibles pour une classe. Il n'y a pas forcément un seul constructeur unique. Revenons maintenant à notre exemple de départ pour voir comment tout ceci s'écrit concrètement. Nous allons ici déclarer un seul constructeur dans notre classe Rectangle. Donc pour cela, on déclare la classe Rectangle comme d'habitude, avec donc ses attributs ici, dans la partie privée, hauteur et largeur. Et puis comme d'habitude, les différents accesseurs que l'on aurait décidé, modificateurs et méthodes que l'on a décidé de mettre dans l'interface. Mais ce que l'on a rajouté donc en plus maintenant, c'est un constructeur. Ce constructeur, c'est une méthode qui a le même nom que le nom de la classe. Bien entendu, ce constructeur doit être dans l'interface, puisque son but est d'être justement appelé par les personnes qui utilisent la classe.

notes

résumé

0m 25s



La *déclaration avec initialisation* d'un objet se fait selon la syntaxe suivante :

Syntaxe :

```
NomClasse instance = new NomClasse(valarg1, ..., valargN);
```

où *valarg1*, ..., *valargN* sont les valeurs des arguments du constructeur.

Exemple :

```
Rectangle r1 = new Rectangle(18.0, 5.8); // invocation du constructeur à 2 paramètres
```

Il faut qu'il soit donc accessible à l'extérieur de la classe, et ici en l'occurrence il reçoit deux paramètres, respectivement « h » et « l » pour pouvoir initialiser respectivement, l'attribut « hauteur » et l'attribut « largeur ». Voyons maintenant quand est appelé un constructeur. C'est-à-dire comment déclarer une instance que l'on initialise en même temps que l'on la déclare. Pour cela, il faut donc faire une déclaration usuelle, une instance ici d'une certaine classe, et puis ensuite faire suivre cette déclaration par « = new », le nom de la classe, ici qui va être un appel au constructeur auquel on passe, entre parenthèses, les différents paramètres nécessaires pour le constructeur. Donc par exemple dans notre classe Rectangle ici, on va déclarer une instance « r1 » de la classe Rectangle = new appel du constructeur « Rectangle » ici qui prend deux paramètres.

notes

résumé

1m 13s

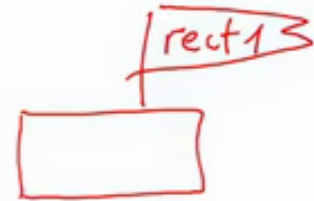


Notre programme (2/4)

```
class Rectangle {
    private double hauteur;
    private double largeur;

    public Rectangle(double h, double l)
    {
        hauteur = h;
        largeur = l;
    }
    public double surface()
    { return hauteur * largeur; }
    // ...
}

class Exemple {
    public static void main(String[] args) {
        Rectangle rect1 = new Rectangle(3.0, 4.0);
        // ...
    }
}
```



Donc concrètement dans l'exemple détaillé qui nous occupe depuis le début, on aurait donc la classe Rectangle avec ici, en public, un constructeur qui prend deux paramètres responsables de l'initialisation des paramètres, des attributs « largeur » et « hauteur » de la classe et par exemple ici dans une classe Exemple où on aurait le main, on déclarerait une instance « rect1 » de la classe Rectangle que l'on initialise directement ici avec un appel au constructeur qui passe la valeur trois au paramètre « h » pour le mettre dans la hauteur et la valeur quatre au paramètre « l » pour le mettre dans la largeur de « rect1 ». Concrètement, pour mémoire, on aurait donc la chose suivante. « rect1 » qui est donc une référence sur un rectangle, et lorsque l'on fait ce new Rectangle, on va créer effectivement une instance en mémoire, Rectangle qui aurait donc ces deux paramètres « hauteur » et « largeur ». Que l'appel au constructeur, ici, va donc initialiser aux valeurs trois et quatre. initialiser aux valeurs trois et quatre.

notes

résumé

2m 1s

