

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W12-01-constrintro-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Mauvaise solution. Bonne solution. Programmeur utilisateur de la classe. Instances de ces classes. Cas général. Classe rectangle. Séquences vidéos précédentes. Programmeur concepteur de la classe. But de l'encapsulation. Méthode init. Travers du manipulateur. Langage java. Choix d'implémentation. Valeur initiale. Set hauteur.



[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 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



Où en est-on ?

Dans les vidéos précédentes, nous avons vu comment déclarer des classes et des objets.

On peut par exemple déclarer une instance de la classe `Rectangle` :

```
Rectangle rect;
```

Une fois que l'on a fait cette déclaration, comment faire pour donner aux attributs de `rect` des valeurs autres que `0.0`

Dans les séquences vidéos précédentes, vous avez appris à déclarer des classes et à déclarer des objets, c'est-à-dire des instances de ces classes. Par exemple, pour déclarer une instance « `rect` » de la classe `Rectangle`, vous écririez « `Rectangle rect;` ». Maintenant, une des questions qu'on peut se poser c'est comment donner d'autres valeurs que les valeurs reçues

notes

résumé

0m 1s



Première solution : **affecter individuellement une valeur** à chaque attribut

```
Rectangle rect;  
  
System.out.println("Quelle hauteur? ");  
lu = clavier.nextDouble();  
rect.setHauteur(lu);  
  
System.out.println("Quelle largeur? ");  
lu = clavier.nextDouble();  
rect.setLargeur(lu);
```

par défaut, c'est-à-dire zéro, aux attributs par exemple « hauteur » et « largeur » de cette instance « rect ». Comment, même mieux, initialiser, c'est-à-dire donner une valeur initiale, au départ, dès la création de « rect » à ses attributs « hauteur » et « largeur ». Bien sûr, on pourrait faire appel aux manipulateurs « set hauteur », « set largeur ». C'est-à-dire, d'affecter individuellement, tour à tour, des valeurs à chacun des attributs comme par exemple, ici, où on a déclaré une instance « rect » de la classe Rectangle où l'on aurait déclaré au préalable un double « lu » et où on demande à l'utilisateur d'entrer

notes

résumé

0m 26s



Première solution : **affecter individuellement une valeur** à chaque attribut

```
Rectangle rect;  
  
System.out.println("Quelle hauteur? ");  
lu = clavier.nextDouble();  
rect.setHauteur(lu);  
  
System.out.println("Quelle largeur? ");  
lu = clavier.nextDouble();  
rect.setLargeur(lu);
```

Ceci est une **mauvaise solution** dans le cas général :

- ▶ elle implique que tous les **attributs fassent partie de l'interface (public)** ou soient assortis d'un **manipulateur** ➡ **casse l'encapsulation**
- ▶ oblige le programmeur-utilisateur de la classe à initialiser explicitement tous les attributs ➡ **risque d'oubli**

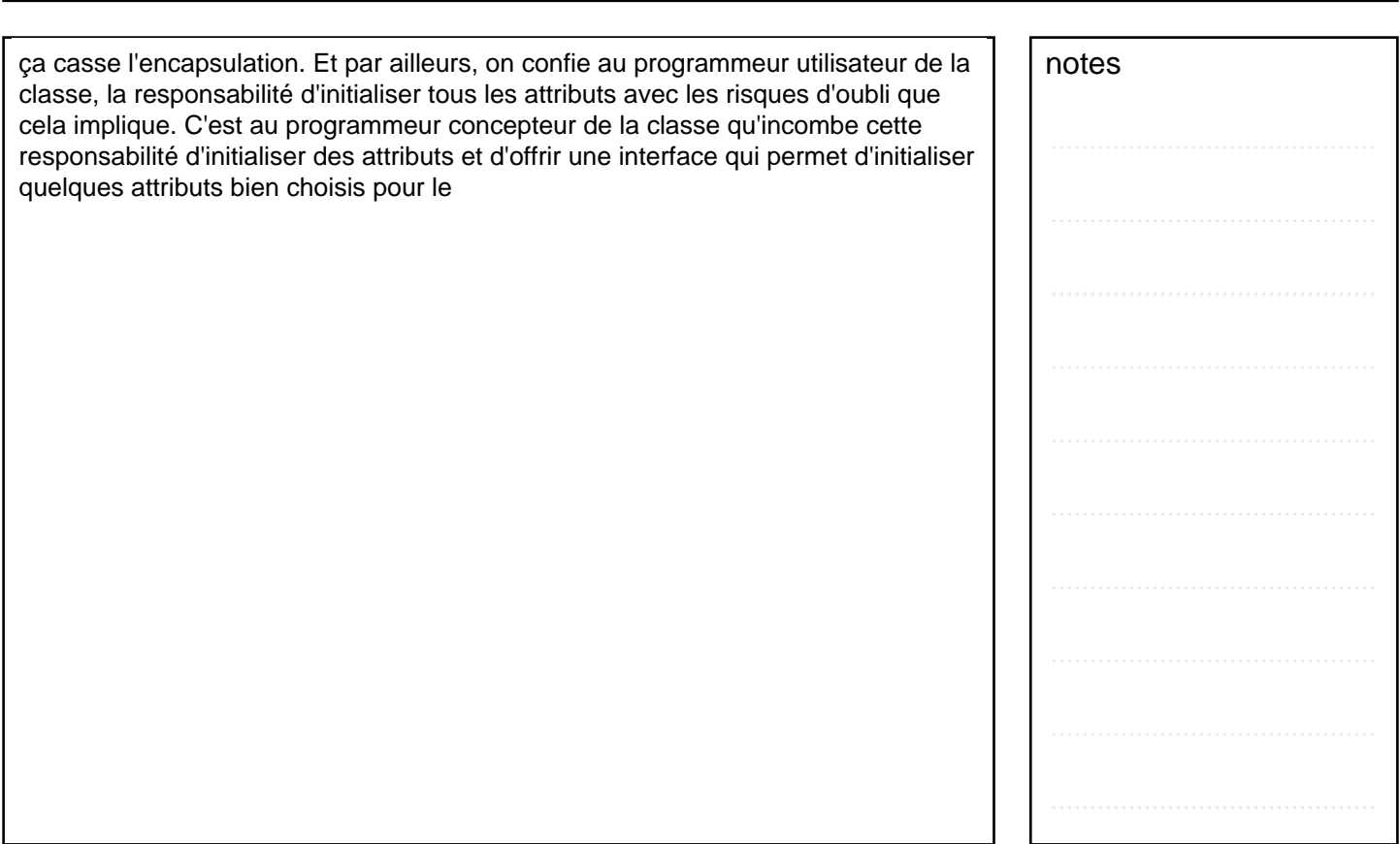
la hauteur au clavier et où au travers du manipulateur « set hauteur », on affecte la valeur que l'on a lue au clavier « lu » à la hauteur de l'instance « rect ». Et de même, via une interaction avec l'utilisateur, on affecte la valeur « lu » à la largeur de l'instance « rect ». Mais ceci est évidemment une très mauvaise solution dans le cas général, parce que tout d'abord, ça nécessite que chacun des attributs soit dans l'interface, ce qui est très mauvais et casse l'encapsulation comme on l'a vu dans les séquences de la semaine précédente. Soit que l'on ait un manipulateur pour chacun des attributs qui lui-même traduit la façon dont la classe a été implémentée et qui donc casse l'encapsulation puisque l'on aurait besoin d'un manipulateur pour chacun des attributs. Le but de l'encapsulation étant justement de séparer clairement l'interface et l'implémentation, si on avait pour chaque attribut, un manipulateur dans l'interface, on traduirait beaucoup trop dans l'interface les choix d'implémentation et c'est dans ce sens là que

notes

résumé

1m 1s





notes



Deuxième solution : définir une **méthode dédiée à l'initialisation** des attributs

```
class Rectangle {  
    private double hauteur;  
    private double largeur;  
  
    public void init(double h, double l)  
    {  
        hauteur = h;  
        largeur = l;  
    }  
    //...  
}
```

programmeur utilisateur de la classe. Une solution consisterait donc, comme à chaque fois que l'on a identifié une tâche spécifique, à créer une méthode dédiée à cette tâche. Donc ici, on veut faire une initialisation, la solution serait donc de créer une méthode dédiée à l'initialisation comme par exemple, une méthode init ici. Bien sûr qu'on offrirait dans l'interface pour que tous les programmeurs utilisateurs de la classe puissent l'utiliser. Donc une méthode init qui aurait pour but d'initialiser les deux attributs « hauteur » et « largeur » et pour ceci recevrait deux valeurs

notes

résumé

2m 25s



Deuxième solution : définir une **méthode dédiée à l'initialisation** des attributs

```
class Rectangle {  
    private double hauteur;  
    private double largeur;  
  
    public void init(double h, double l)  
    {  
        hauteur = h;  
        largeur = l;  
    }  
    //...  
}
```

*Rectangle rect;
rect.init(3.0, 4.0);*

« h » et « l », respectivement pour initialiser la hauteur et la largeur. Typiquement, on utiliserait donc cette méthode d'initialisation que nous avons créée de la façon suivante. En déclarant une instance « rect » de la classe Rectangle, et ensuite en appelant cette méthode « init » pour initialiser,

notes

résumé

3m 1s



Deuxième solution : définir une **méthode dédiée à l'initialisation** des attributs

```
class Rectangle {  
    private double hauteur;  
    private double largeur;  
  
    public void init(double h, double l)  
    {  
        hauteur = h;  
        largeur = l;  
    }  
    //...  
}
```

Pour faire ces initialisations, il existe en Java des méthodes particulières appelées **constructeurs**.

par exemple, la hauteur à trois et la largeur à quatre. C'est certainement une bonne solution, c'est tellement une bonne solution que le langage Java vous offre le moyen de le faire proprement. C'est-à-dire de faire vraiment une initialisation. De telles méthodes

notes

résumé

3m 25s



Les constructeurs

Un constructeur est une méthode :

- ▶ invoquée *systématiquement* lors de la déclaration d'un objet
- ▶ chargée d'effectuer toutes les opérations requises en « début de vie » de l'objet (dont *l'initialisation des attributs*)

Syntaxe de base :

```
NomClasse(liste_paramètres)
{
    /* initialisation des attributs
       en utilisant liste_paramètres */
}
```



chargées d'initialiser les instances, c'est ce que l'on appelle des constructeurs. Un constructeur est une méthode qui est systématiquement appelée lors de la déclaration d'un objet. C'est-à-dire appelé au début de vie de chaque instance. Il a donc pour rôle de faire tout ce que l'on doit faire au début de vie de chaque instance et en particulier d'initialiser les attributs. d'initialiser les attributs.

notes

résumé

3m 36s

