

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W11-03-publicprivate-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Java des classes. Partie visible des classes. Exemple de la classe rectangle. Méthode surface. Mot clé private. Notion d'interface. Partie implémentation. Séquence vidéo précédente. Extérieur de la classe. Partie publique. Partie de l'interface. Instance de la classe rectangle. Figure de résumé de la programmation. Accès direct. Objet suivante.



[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/>



public **et** private

(Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





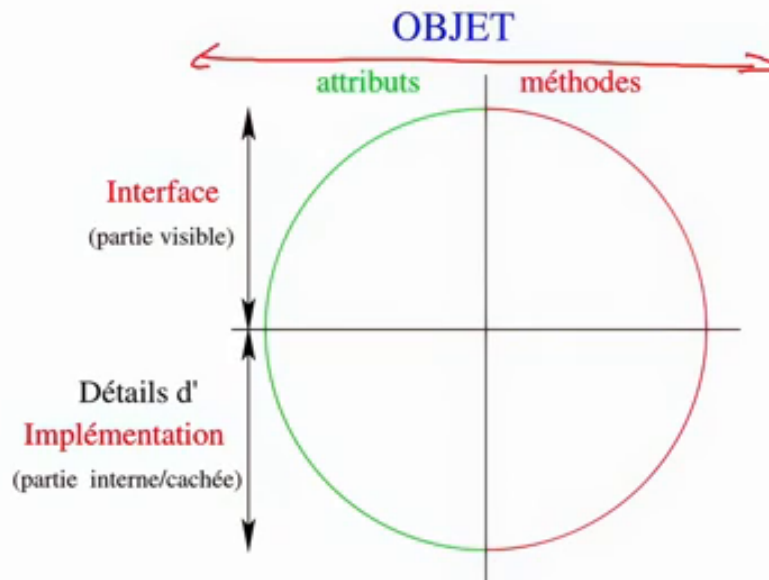
Dans la séquence vidéo précédente nous avons vu comment

notes

résumé

0m 1s





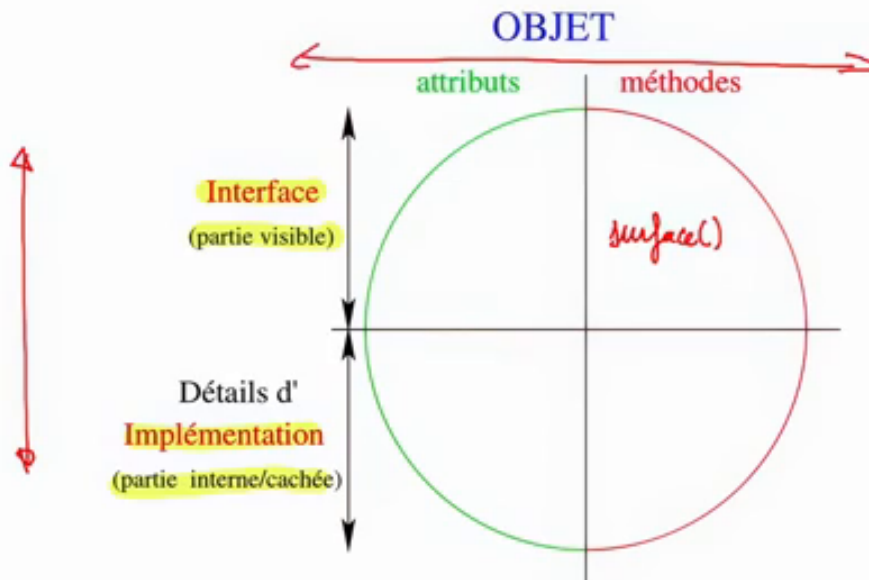
concrètement déclarer en java des classes, des objets, comment leur définir des attributs et des méthodes, c'est à dire, sur la figure de résumé de la programmation orientée objet suivante, essentiellement cet axe là que nous avons développé dans la vidéo précédente. Mais comment maintenant en java

notes

résumé

0m 5s





concrètement donner corps à la notion d'interface, c'est-à-dire la partie visible des classes et la partie implémentation, c'est-à-dire cachée c'est-à-dire concrètement comment implémenter en java cet axe là de la figure ; c'est l'objet de la présente séquence vidéo. Concrètement si on veut reprendre l'exemple de la classe rectangle nous avons décidé d'avoir une méthode surface qui faisait partie

notes

résumé

0m 25s



Tout ce qu'il n'est pas nécessaire de connaître à l'extérieur d'un objet devrait être dans le corps de l'objet et identifié par le mot clé `private` :

```
class Rectangle {  
    private double hauteur;  
    private double largeur;  
    double surface();  
}
```

*Rectangle rect = new Rectangle();
rect.hauteur = 3.6;*

Attribut d'instance **privée** = inaccessible depuis l'extérieur de la classe.
C'est également valable pour les méthodes.

Erreur de compilation si référence à un(e) attribut/méthode d'instance privée :

`error: hauteur has private access in Rectangle`

Note : Si aucun droit d'accès n'est précisé il s'agit des droits d'accès par défaut («friendly»).

de l'interface qu'on pouvait utiliser de partout dans le programme et puis d'avoir deux attributs privés qui étaient la largeur et la hauteur. En java ceci va s'écrire : pour déclarer la partie privée on va rajouter le mot clé `private` devant chacun des attributs et des méthodes que l'on veut cacher, rendre inaccessibles depuis l'extérieur de la classe que l'on ne veut pas mettre dans l'interface. inaccessibles depuis l'extérieur de la classe concrètement veut dire la chose suivante : si par exemple dans la méthode `main` ou dans une autre classe on écrivait ceci, donc on déclare une instance de la classe `Rectangle`, et qu'on souhaite accéder à la hauteur en écrivant comme ça, en accès direct par exemple, le compilateur donnera une erreur

notes

résumé

0m 49s





qui dit que le champ hauteur est d'accès privé dans la classe Rectangle. A noter que si aucun droit n'est précisé, comme par exemple ici, surface, alors on a ce qu'on appelle les accès par défaut que je vous expliquerai dans quelques instants. Nous avons donc vu que pour déclarer la partie implémentation, c'est-à-dire

notes

résumé

1m 37s



À l'inverse, l'interface, qui est accessible de l'extérieur, se déclare avec le mot-clé `public`:

```
class Rectangle {  
    public double surface() { ... }  
    ...  
}
```

```
Rectangle rect = new Rectangle();  
z = rect.surface();
```

privée, on utilisait le mot clé `private`, à l'inverse pour la partie interface, c'est-à-dire la partie publique que l'on souhaite offrir à l'extérieur, offrir à toutes les autres classes, on utilise le mot clé `public`, donc par exemple la méthode `surface` étant dans l'interface de `Rectangle`, on va déclarer ici `public` devant l'en tête et la définition de la méthode `surface`. Si je reprends l'exemple précédent d'une méthode `Main` qui déclarerait une instance `rect` de la classe `Rectangle`, celle-ci aurait donc ici le droit d'appeler la méthode `surface` de la classe `Rectangle` par exemple pour la stocker dans une autre variable et ceci est possible parce que la méthode `surface` est déclarée ici dans l'interface comme `public` ; si jamais on avait déclaré ceci comme `private`, alors comme précédemment on n'aurait pas le droit d'accéder à la méthode `surface` de l'instance `rect` de la classe `Rectangle`.

notes

résumé

2m 1s

