

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W12-04-complements-JAVA-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Méthode particulière. Contenu du rectangle. Chaîne de caractères. Contenu de mon objet. Méthode toString. Valeur de ses attributs. Affichage des objets. Ligne de code. Classe rectangle. Sorte d'adresse. Forme d'une chaîne de caractère. Valeur du champ. Exécution de cette ligne de code. Références. Travers d'une instruction telle.



[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/>

Fin de vie, affectation, affichage et comparaison d'objets

(Partie 3)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

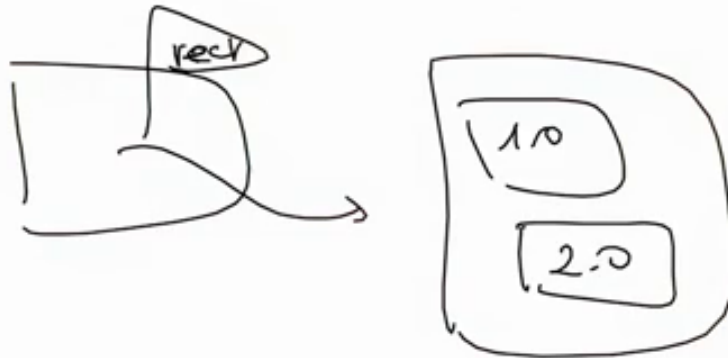
0m 0s



La portion de code suivante :

```
Rectangle rect = new Rectangle(1.0, 2.0);  
System.out.println(rect);
```

afficherait la valeur de la référence. Par exemple : `A@29c2fff0` !



Toujours sur le même thème, les objets manipulés par les références en Java, intéressons-nous à ce que cela implique pour l'affichage des objets. Si j'exécute cette ligne de code, ce qui est contenu dans ma variable « rect » est donc bien une référence. Une référence est une sorte d'adresse, donc quelque chose qui a un format un petit peu bizarre. Donc maintenant si j'essaye

notes

résumé

0m 1s



La portion de code suivante :

```
Rectangle rect = new Rectangle(1.0, 2.0);  
System.out.println(rect);
```

afficherait la valeur de la référence. Par exemple : `A@29c2fff0` !

Que faire si l'on souhaite faire afficher le contenu de l'objet en utilisant exactement le même code ?

Java prévoit que vous fournissiez une méthode qui retourne une représentation de l'instance sous forme d'une `String`.

Il est prévu que vous donniez un entête particulier à cette méthode :

```
String toString()
```

La méthode `toString` est invoquée automatiquement par `System.out.println`

d'afficher le contenu de mon objet « rect » au travers d'une instruction telle que celle-ci. Cette instruction va simplement afficher ce qui est contenu dans la variable « rect ». Et donc quelque chose comme ceci, ce qui est assez peu naturel et assez peu lisible, nous préférons sans doute pouvoir afficher le contenu du rectangle, c'est-à-dire la valeur de ses attributs plutôt que sa référence. Que faire donc si l'on souhaite faire afficher le contenu pointé par la référence plutôt que la référence elle-même en utilisant exactement cette même instruction ? Alors pour que ceci soit possible, pour qu'une ligne telle que celle-ci affiche le contenu pointé par « rect » plutôt qu'une référence un peu étrange, Il suffit de placer dans la classe Rectangle une méthode particulière. Une méthode dont l'en-tête est le suivant.

notes

résumé

0m 37s



Exemple :

```
class Rectangle
{
    private double hauteur;
    private double largeur;
    //...
    public String toString()
    {
        → return "Rectangle " + hauteur + " x " + largeur;
    }
}

class Exemple {
    public static void main(String[] args) {
        → System.out.println(new Rectangle(4.0, 5.0));
    }
}
```

affiche : Rectangle 4.0 x 5.0

Il s'agit de la méthode toString() dont la liste des paramètres est vide, et qui retourne une chaîne de caractères, une string. Cette méthode toString() va en réalité être automatiquement invoquée pour convertir ici votre rectangle en une représentation sous la forme d'une chaîne de caractère. C'est à vous de définir comment vous souhaitez que cette chaîne de caractères soit produite par la méthode toString(). Donc concrètement, dans la classe Rectangle, on placerait notre fameuse méthode toString() publique puisqu'on souhaite pouvoir l'utiliser depuis l'extérieur. Cette méthode toString() va simplement construire une chaîne de caractère qui va correspondre à la représentation que l'on souhaite avoir lorsque l'objet est fourni en argument de « System.out.println » typiquement. Donc ici nous avons décidé que la chaîne « String » serait construite comme étant la concaténation de la chaîne « Rectangle », de la valeur du champ « hauteur », de la valeur du champ « largeur » séparé par cette chaîne de caractère-là. Donc ici, puisque désormais tout objet Rectangle a une représentation construite au moyen de la méthode toString(), une représentation d'affichage sous la forme d'une chaîne de caractères. L'exécution de cette ligne de code, afficher un rectangle ayant ces valeurs particulières en guise d'attribut, va produire un affichage de cette nature. Donc un affichage beaucoup plus lisible que celui qu'on avait tout à l'heure. tout à l'heure.

notes

résumé

1m 25s

