

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W12-03-constrcopie-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Constructeur particulier. Constructeur de copie. Copies d'instances. Copie de la hauteur de r1. Copie de la largeur de r1. Copie de r1. Classe rectangle. Cas de notre classe rectangle. Liste de paramètres. Construction de r2. Copie de la hauteur. Première instance r1 de la classe rectangle. Seconde instance. Seul paramètre. Propre hauteur de l'instance.



[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/>

Constructeur de copie

(Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Nous allons dans cette séquence-ci

notes

résumé

0m 1s

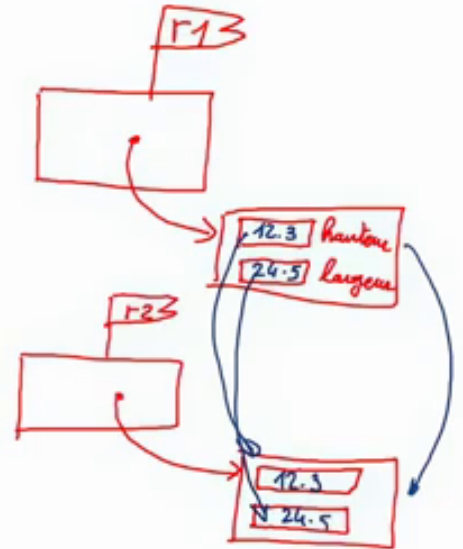


Constructeur de copie

Java offre un moyen de créer la **copie** d'une instance :
le *constructeur de copie*

```
Rectangle r1 = new Rectangle(12.3, 24.5);
Rectangle r2 = new Rectangle(r1);
```

r1 et **r2** sont deux *instances distinctes*
mais ayant des mêmes valeurs pour leurs attributs
(au moins juste après la copie).



nous intéresser à un autre constructeur particulier, le constructeur de copie.
C'est-à-dire que nous allons regarder ce qu'il se passe lorsqu'un objet est initialisé
avec une copie d'un autre objet de la même classe. JAVA offre en effet un moyen
de créer des copies d'instances, comme par exemple ici, où nous avons une
première instance r1 de la classe Rectangle que nous initialisons avec un appel ici
au constructeur qui prend donc 2 valeurs, ce qui nous donne donc le schéma
suivant. Donc initialisée avec les valeurs 12.3 et 24.5 et où nous avons donc une
seconde instance r2 de la classe Rectangle, qui elle, est initialisée donc avec une
copie de r1. On va donc effectuer ici la copie de la hauteur de r1 dans r2 et la copie
de la largeur de r1 dans r2.

notes

résumé

0m 5s



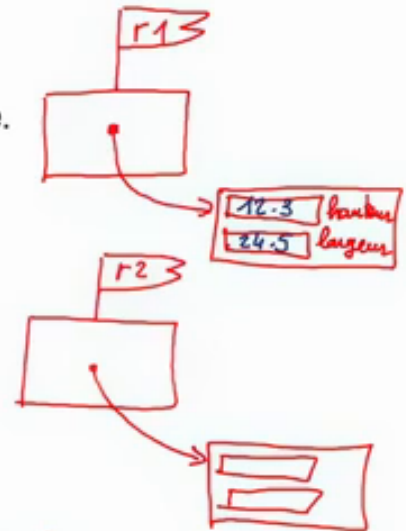
Le constructeur de copie permet d'initialiser une instance en *copiant* les attributs d'une *autre instance* du même type.

Syntaxe :

```
NomClasse(NomClasse autre) { ... }
```

Exemple :

```
public Rectangle(Rectangle r1 autreRectangle)
{
    hauteur = autreRectangle.hauteur;
    largeur = autreRectangle.largeur;
}
```



Rectangle r2 = new Rectangle(r1);

Mais on a bien que r1 et r2 sont deux instances séparées ici, où simplement on a initialisé r2 avec une copie de r1. C'est le constructeur de copie qui réalise ce type d'initialisation. Le constructeur de copie a donc pour but d'initialiser une instance avec une copie d'une autre instance de la même classe. Son entête est donc parfaitement définie puisque c'est un constructeur. Donc il a le même nom que le nom de la classe. Mais comme il reçoit une autre instance de la même classe, sa liste de paramètres est réduite à un seul paramètre, qui est une autre instance de la même classe. Donc par exemple dans le cas de notre classe Rectangle, on aurait ici un constructeur bien sûr qui est toujours dans l'interface, de la classe Rectangle, donc qui s'appelle Rectangle, mais qui reçoit comme unique autre paramètre, un autre rectangle, autre instance de la classe Rectangle ; et qui donc, par exemple, serait naturellement la copie de la hauteur de l'autre rectangle reçu en paramètre, dans notre propre hauteur de l'instance qu'on est en train d'initialiser. Et puis copie de la largeur de l'autre rectangle, dans la largeur de l'instance qu'on est en train d'initialiser. Si je reprends donc l'exemple précédent où nous avons un rectangle r1 avec une hauteur de 12.3 et une largeur de 24.5, et un rectangle r2 initialisé avec le constructeur de copie. Dans cette ligne donc on a bien appel à ce constructeur de copie. Donc ici c'est bien la construction de r2 qui va lancer l'appel au constructeur de copie en passant comme paramètre r1, qui est donc le paramètre ici autreRectangle. Ici donc dans ce cas-là, c'est bien r1

notes

résumé

1m 1s



Constructeur de copie (2)

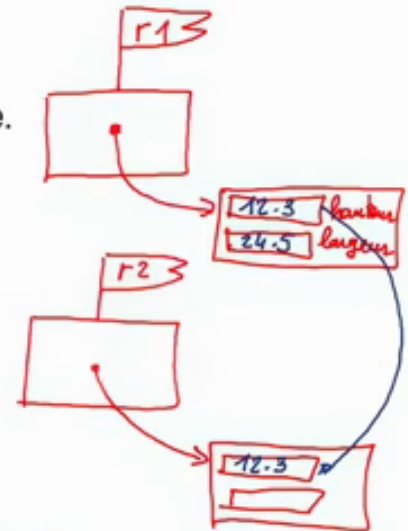
Le constructeur de copie permet d'initialiser une instance en *copiant* les attributs d'une *autre instance* du même type.

Syntaxe :

```
NomClasse(NomClasse autre) { ... }
```

Exemple :

```
public Rectangle(Rectangle r1 autreRectangle)
{
    hauteur = autreRectangle.hauteur; ←
    largeur = autreRectangle.largeur; ←
}
```



Rectangle r2 = new Rectangle(r1);

qui est passé au constructeur de copie de r2. Et donc ce qui se passe, c'est qu'on a ici la hauteur de r2 qui va recevoir la hauteur de r1 en copie. Donc ici, ce qu'il va se passer à cette ligne, c'est qu'on va recopier la hauteur de r1 dans la hauteur de r2. Et dans la ligne suivante, ce que l'on fait, c'est que l'on recopie donc la largeur, ici c'est largeur du rectangle qu'on est en train de construire, de r2. Donc ici c'est bien la largeur de r2 qui reçoit la largeur de r1. Et donc ici on va recopier la largeur de r1 dans celle de r2. A noter que contrairement à d'autres langages, java ne fournit pas par défaut une version du constructeur de copie. C'est-à-dire que si on ne fait rien de particulier, on n'a pas le droit comme ça, de faire des copies. Ce constructeur ici n'existe pas, ce qu'il faut faire c'est, si on veut faire une construction de copie, explicitement donc, l'écrire. explicitement donc, l'écrire.

notes

résumé

3m 1s

