

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W14-02-polymcomplements-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Travers de la classe b. Constructeur polymorphe. Cas d'une super-classe. Corps m. Méthode polymorphe. Cadre de la construction des objets. Appel explicite. Constructeur de la classe a. Corps d'un constructeur. Classe b. Défaut de la classe b.. Objets de type sous-classe. Premier point abordé. Méthode abstraite. But de cette séquence.



[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/>

Héritage et polymorphisme : compléments

(Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Le but de cette séquence est de présenter divers petits compléments

notes

résumé

0m 1s



Un constructeur est une méthode spécifiquement dédiée à la construction de l'**instance courante** d'une classe, il n'est pas prévu qu'il ait un comportement polymorphique.

Il est cependant possible d'invoquer une méthode polymorphique dans le corps d'un constructeur

- ☞ Ceci est cependant **déconseillé** : la méthode agit sur un objet qui n'est peut-être alors que partiellement initialisé !

Un exemple ...

en relation avec le polymorphisme et l'héritage. Le premier point abordé est qu'en est-il du polymorphisme dans le cadre de la construction des objets ? Les constructeurs sont des méthodes un peu particulières dans le sens où elles sont spécifiquement dédiées à la construction de l'instance courante d'une classe.

notes

résumé

0m 5s



Constructeurs et polymorphisme (2)

EPFL

~~A a = new A();~~

```
abstract class A
{
    public abstract void m();
    public A() {
        m(); // méthode invocable de manière polymorphique
    }
}

class B extends A {
    private int b;
    public B() {
        b = 1; // A() est invoquée implicitement juste avant
    }
    public void m() { // définition de m pour la classe B
        System.out.println("b vaut : " + b);
    }
}

// .... dans le programme principal :
B b = new B();
```

affiche : b vaut : 0

Il n'est pas prévu qu'ils aient un comportement polymorphique. Imaginer un constructeur polymorphique voudrait dire que ce dernier serait dédié à initialiser des objets autres que l'instance courante ; des objets de type sous-classe, par exemple, ce qui n'a pas vraiment de sens. Rien n'empêche cependant d'invoquer une méthode polymorphique dans le corps d'un constructeur. Mais ceci est cependant déconseillé. En effet, comme nous allons voir sur l'exemple suivant, la méthode agit alors sur un objet qui n'est peut-être que partiellement initialisé. Nous avons dans cet exemple le cas d'une super-classe A, qui contient un constructeur par défaut et également une méthode abstraite, c'est-à-dire une méthode sans corps m. Il se trouve que le constructeur invoque la méthode m. Rappelons qu'une classe abstraite ne peut jamais être instanciée, ce qui signifie que l'on ne peut jamais appeler le constructeur de la classe A comme ceci, du fait que la classe est abstraite. Invoquer une méthode abstraite dont le constructeur de A, c'est-à-dire une méthode sans corps, ne pose pas problème parce qu'on ne va jamais appeler ce constructeur pour instancier un objet de type A, c'est-à-dire un objet pour lequel m n'a pas de définition concrète. Ceci est donc tout à fait licite. Concrètement, la méthode m ne sera invoquée que si le constructeur d'une sous-classe invoque ce constructeur de la super-classe et que cette sous-classe a une définition concrète de la méthode m. Nous en avons un exemple ici, au travers de la classe B. La classe B hérite de A et redéfinit concrètement la méthode m. La redéfinition consiste simplement à afficher un message qui comporte la valeur de l'attribut b de la classe B. La classe B est instanciable, puisqu'elle redéfinit toutes les méthodes abstraites héritées de plus haut, en l'occurrence, il ne s'agit que de la méthode m. Voyons donc ce qui

notes

résumé

0m 25s



Constructeurs et polymorphisme (2)

EPFL

~~A a = new A();~~

```
abstract class A
{
    public abstract void m();
    public A() {
        m(); // méthode invocable de manière polymorphique
    }
}
class B extends A {
    private int b;
    public B() {
        b = 1; // A() est invoquée implicitement juste avant
    }
    public void m() { // définition de m pour la classe B
        System.out.println("b vaut : " + b);
    }
}
// .... dans le programme principal :
B b = new B();
```

affiche : b vaut : 0

se passe lorsque nous créons une instance de B. L'instance est créée ici au moyen du constructeur par défaut de la classe B. Nous savons que tout constructeur de sous-classe doit nécessairement invoquer un constructeur de la super-classe, lorsqu'il n'y a pas d'appel explicite à un constructeur via la tournure `super`, alors nous savons qu'il y a un appel implicite au constructeur par défaut. Cet appel se fait pour initialiser une instance courante qui est un B. Au moment de l'appel à `m` dans le constructeur de A, puisqu'il y a nécessairement résolution dynamique des liens en Java, la méthode va être choisie en fonction de la nature réelle de l'instance et c'est donc cette méthode `m` qui va être appelée. Rappelez-vous qu'à ce stade, nous ne sommes pas encore passés par l'instruction qui consiste à initialiser l'attribut `b` de B avec une valeur particulière. Par conséquent, l'attribut `b` a la valeur qui est donnée par défaut avant toute initialisation explicite, qui est la valeur zéro. Ceci signifie que lorsque la méthode `m` s'exécute, l'attribut `b` a pour valeur zéro

notes

résumé

Un constructeur est une méthode spécifiquement dédiée à la construction de l'**instance courante** d'une classe, il n'est pas prévu qu'il ait un comportement polymorphique.

Il est cependant possible d'invoquer une méthode polymorphique dans le corps d'un constructeur

- ☞ Ceci est cependant **déconseillé** : la méthode agit sur un objet qui n'est peut-être alors que partiellement initialisé !

Un exemple ...

et donc que la construction d'un B va résulter dans l'affichage, du message " b vaut : 0 " Si pour nous, l'objet B n'est proprement initialisé que si son attribut b vaut 1, alors nous voyons bien que la méthode m est en train de travailler sur un objet qui est partiellement initialisé. D'où le conseil de départ : ne pas invoquer de méthode polymorphique dans le corps des constructeurs. dans le corps des constructeurs.

notes

résumé

3m 25s

