

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W15-02-methodstat-JAVA-pt2

Concepts (extraits des sous-titres générés automatiquement) :

Méthodes statiques. Méthodes de classe. Instances particulières d'une classe. Deuxième situation. Création d'un objet de type. Classes boîte. Méthodes outils. Méthode main. Constante de classe. Mot clé static. Cas de la classe. Constante pi de la classe. Méthode de classe. Méthodes auxiliaires. Programmation procédurale générale.



[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/>

Méthodes statiques

(Partie 2)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



☞ Méthodes qui ne sont pas liées à un objet

Exemple :

- ▶ Classe mettant à disposition des **utilitaires** mathématiques divers
- ▶ La création d'un objet de type `MathUtils` est artificielle
- ▶ La classe sert seulement à stocker des méthodes utilitaires

```
class MathUtils {  
    public final static double PI = 3.14159265358979323846;  
    public static double auCube(double d) {  
        return d*d*d;  
    }  
}
```

MathUtils.PI
MathUtils.auCube(3.4);

A quoi peuvent donc servir ces méthodes statiques, méthodes de classe ? Eh bien justement, à écrire des méthodes qui ne sont pas attachées à des instances particulières d'une classe, des méthodes très générales, typiquement des méthodes outils. On va faire comme ceci des classes boîte à outils, dont on ne créera pas d'instance, mais dont on utilisera simplement les méthodes de classe. Par exemple, une boîte à outils mathématiques, comme par exemple cette classe ici, « `MathUtils` », dans laquelle nous avons défini une constante de classe, dans laquelle on peut définir aussi une méthode de classe, c'est-à-dire typiquement ici, une fonction outil « `auCube` », qui permettrait donc de calculer le cube de son paramètre. Evidemment la création d'un objet de type « `MathUtils` », est totalement artificielle. On n'a pas besoin d'objet de type « `MathUtils` », et on pourrait directement appeler la constante `PI` de la classe « `MathUtils` », puisque je répète ici que `PI` est bien une variable de classe. On pourrait donc aussi appeler comme ceci « `MathUtils.auCube(3.4)` », puisque ici aussi, « `auCube` » est une méthode d'instance. On peut donc l'appeler sans aucune création d'un objet de « `MathUtils` », mais directement avec le nom de la classe. le nom de la méthode pour calculer le cube de la valeur 3.4. Cette classe « `MathUtils` » sert seulement à stocker des méthodes et des constantes utilitaires. Elle ne sert pas à créer d'objets concrets.

notes

résumé

0m 1s



Eviter la prolifération de `static` !

On l'utilise seulement dans des situations très particulières :

- ▶ définition d'une constante : attribut `final static` (situation très courante)
- ▶ utilisation d'une valeur commune : attribut `static` (plus rare)
- ▶ méthodes utilitaires qu'il est artificiel de lier à un objet : méthode `static`, *invocable sans objet* (plus rare aussi)

Math.PI

Exemples de méthodes statiques :

- ▶ `Math.sqrt`
- ▶ la méthode `main`

En fait, c'est exactement le cas de la classe « `Math.PI` », et si vous aviez à utiliser la constante `PI`, vous utilisez celle de l'API ; et ça s'écrit en Java standard, « `Math.PI` », la classe « `Math` » de l'API exactement comme notre petit exemple pédagogique « `MathUtils` ». Vous ne faites évidemment vous-même jamais ceci, mais vous utilisez les constantes qui sont définies dans l'API. Voilà donc typiquement à quoi peuvent servir les méthodes statiques, les méthodes de classe. Ceci dit, évitez absolument d'abuser de ce mot clé `static`. Évitez de faire proliférer ce mot clé `static`, et réservez-le pour des cas très particuliers. Typiquement, les trois cas suivants, on pourra utiliser `static` avec « `final` » pour un attribut, une variable donc de classe ici, pour définir des constantes. C'était l'exemple de la variable « `Math.PI` ». La deuxième situation c'est d'utiliser `static` sans « `final` », toujours pour un attribut. C'est plus rare, et c'est pour partager une valeur commune ici au niveau de la classe, valeur commune pour la classe, mais qui pourrait évoluer. Cette situation est plus rare et vous ne la rencontrez pas dans le cadre des exercices de ce cours. Concernant les méthodes enfin, on peut définir des méthodes de classe, des méthodes statiques, lorsqu'on a comme on a vu, des méthodes utilitaires, qu'il est artificiel de lier à un objet particulier, et qui sont donc invocable sans aucun objet. C'est en général aussi beaucoup plus rare que vous en écriviez par vous-mêmes, mais typiquement vous utiliserez celles fournies par l'API, comme par exemple la méthode racine carrée `sqrt` de la classe « `Math` » fournie dans l'API Java.

notes

résumé

1m 49s



Nous comprenons maintenant pourquoi les méthodes auxiliaires de la méthode `main` sont statiques (mais pas les méthodes dans les classes)

La méthode `main` a un en-tête fixe :

```
public static void main(String[] args)
```

Puisque la méthode `main` est obligatoirement statique :

- ▶ elle ne peut pas accéder à l'objet `this`
- ▶ elle ne peut pas accéder à des variables/méthodes d'instance
- ▶ elle peut seulement accéder à des variables/méthodes statiques

En dehors de cela, la classe de la méthode `main` est comme n'importe quelle classe.

Elle peut avoir des constructeurs, des méthodes et des variables

Un autre exemple de méthode statique, que vous avez eu l'habitude de pratiquer, c'est la méthode `main`. Et nous comprenons maintenant pourquoi les méthodes auxiliaires, ce qu'on appelle fonction en programmation procédurale générale, donc les méthodes auxiliaires de `main`, sont des méthodes statiques. Puisqu'en effet `main` a un entête fixé, c'est une méthode qui doit être statique dans une classe. En tant que méthode statique, elle ne peut accéder qu'à d'autres méthodes statiques, et donc les méthodes auxiliaires de `main` doivent nécessairement être des méthodes statiques. De même, la méthode `main`, puisqu'elle est statique,

notes

résumé

3m 37s





ne peut accéder à aucun objet de la classe dans laquelle elle a été définie. Donc elle ne peut pas accéder à des variables et des méthodes d'instance de cette classe dans laquelle elle est définie. A noter qu'en dehors de cela, la classe dans laquelle la méthode main est définie, est comme n'importe quelle autre classe, elle peut avoir des attributs, des méthodes, des constructeurs, etc. Voilà, ceci conclut donc cette séquence sur les méthodes statiques, sur les méthodes de classe. sur les méthodes de classe.

notes

résumé

4m 13s

