

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W15-01-attributstat-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Sortes de variables. Instance de la classe. Notion de variable statique. Zone mémoire. Petit exemple du rectangle. Objet de type rectangle. Variables d'instances. Intérieur d'une classe. Intérieur du corps de méthode. Instance de rectangle. Modificateur statique. Variables locales. Séquences précédentes. Propres zones mémoire. Objet de la 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/>

Attributs statiques (Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Dans les séquences précédentes, nous avons vu que

notes

résumé

0m 1s



Nos variables jusqu'à maintenant :

1. Variables d'instance (= attributs)
 - Décrivent les attributs d'un objet
2. Variables locales
 - Déclarées à l'intérieur d'une méthode
3. Paramètres
 - Pour envoyer des valeurs à une méthode
 - S'utilisent comme des variables locales

Nouveauté :

4. Variables statiques = variables de classe
 - Indiquées par le modificateur `static`
 - Ressemblent aux variables d'instance
 - Déclarées en dehors des méthodes
 - Visibles partout dans la classe
 - Héritées par les sous-classes

les attributs définis à l'intérieur d'une classe représentent des informations qui sont spécifiques à chaque instance de la classe. Si l'on reprend notre petit exemple du rectangle, chaque instance de rectangle, chaque objet de type rectangle, va avoir sa propre largeur, sa propre hauteur, des informations qui lui sont spécifiques et qui la caractérisent. Que se passe-t-il cependant si plusieurs instances d'une même classe ont besoin d'accéder à une même information, une information commune ? C'est l'objet de la séquence qui suit. Nous avons jusqu'ici utilisé différentes sortes de variables dans nos programmes, nous avons utilisé des variables d'instances, ce que l'on appelle communément des attributs, nous avons utilisé des variables locales, c'est-à-dire des variables déclarées à l'intérieur du corps de méthode, ainsi que des paramètres que l'on peut assimiler à des variables locales dans les méthodes, et qui servent à communiquer des données à des méthodes pour qu'elles puissent travailler. Dans cette séquence, vous allez découvrir la notion de variable statique, qu'on appelle également des variables de classe, par opposition aux variables d'instance. Il s'agit en fait d'attributs particuliers dont la déclaration est précédée du mot réservé « `static` », ils ressemblent aux attributs classiques, aux variables d'instances, ils sont déclarés de la même façon, hormis le modificateur statique qui vient en plus, déclarés comme les attributs d'instance,

notes

résumé

0m 6s



- ▶ Variable d'instance :
 - ▶ Réserve d'une zone pour chaque objet construit avec `new`
 - ▶ Résultat : chaque objet a sa propre zone/valeur pour la variable d'instance
- ▶ Variable de classe (statique) :
 - ▶ Déclaration précédée par `static`
 - ▶ Réserve d'une zone lors du chargement de la classe
 - ▶ Aucune zone réservée quand un objet est construit avec `new`
 - ▶ Résultat : tous les objets se réfèrent à la même zone/valeur pour la variable de classe

comme les variables d'instance, en dehors des méthodes. Tout comme les variables d'instance, ils sont visibles partout dans la classe et sont également hérités par les sous-classes. Nous allons dans cette séquence surtout nous attarder sur les variables statiques, mais il faut savoir qu'il existe également des méthodes statiques, ce que nous allons couvrir un peu plus tard. En clair, le modificateur statique s'utilise en Java aussi bien pour les variables que pour les méthodes, dans les deux cas, un membre statique peut être utilisé sans avoir construit d'objet au préalable. Vous noterez qu'en Java, le modificateur statique ne s'applique pas aux variables locales, c'est-à-dire aux variables déclarées à l'intérieur du corps de méthode. Mais que veut dire qu'un membre statique puisse être accédé sans construire d'objets ? Si l'on reprend l'exemple du rectangle, ce que nous avons utilisé jusqu'ici étaient des attributs déclarés à l'intérieur de la classe, ce que l'on appelle désormais aussi des variables d'instance, et nous avons des méthodes, et dans ce contexte, il est strictement impossible d'utiliser un membre de la classe rectangle sans construire au préalable une instance de rectangle. Donc ici par exemple, si je veux utiliser la méthode surface, je suis obligé de construire un objet de type rectangle, et au travers de cet objet, appeler la méthode qui calcule la surface, je calcule la surface du rectangle. Un membre statique, que ce soit une variable ou une méthode, est invocable sans avoir au préalable créé une instance en utilisant uniquement le nom de la classe selon des modalités que nous allons examiner un peu plus tard. Donc une des facettes est que l'on peut utiliser un membre statique sans construire d'objets, l'autre facette, relative aux variables statiques, est qu'elles sont partagées entre toutes les instances de la classe. Vous avez appris jusqu'ici que chaque

notes

résumé

1m 25s



- ▶ Variable d'instance :
 - ▶ Réserve d'une zone pour chaque objet construit avec `new`
 - ▶ Résultat : chaque objet a sa propre zone/valeur pour la variable d'instance
- ▶ Variable de classe (statique) :
 - ▶ Déclaration précédée par `static`
 - ▶ Réserve d'une zone lors du chargement de la classe
 - ▶ Aucune zone réservée quand un objet est construit avec `new`
 - ▶ Résultat : tous les objets se réfèrent à la même zone/valeur pour la variable de classe

instance dispose d'une zone mémoire qui lui est propre pour chacun de ses attributs. En clair, chaque rectangle instancié dans un programme aura ses propres zones mémoire pour la largeur et la hauteur. Donc ici par exemple, pour le rectangle « r1 », nous aurons des zones mémoire stockant la valeur des largeurs et hauteurs pour le rectangle en question, pour un autre rectangle, il y aura d'autres zones mémoire qui vont stocker la largeur et la hauteur de ce deuxième rectangle. C'est pour cela d'ailleurs qu'on parle de variable d'instance, car chaque instance dispose de ses propres variables. Par opposition, une variable statique serait une zone mémoire unique liée à la classe plutôt qu'aux instances, mais également accessible via chaque instance. Donc en clair, pour les variables d'instance, les attributs, il y a réserve d'une zone mémoire pour chaque objet que l'on a instancié, donc créé avec « new », donc chaque objet a sa propre zone mémoire, chaque valeur pour la variable d'instance en question,

notes

résumé

pour une variable de classe, c'est-à-dire un attribut de la classe que l'on aurait déclaré comme statique, il y a réservation d'une zone mémoire unique, cette réservation se fait dès lors que l'on charge la classe, c'est-à-dire pour simplifier, dès lors que la classe est mentionnée dans un programme. Aucune zone mémoire ne sera réservée pour cet attribut lorsque l'on crée une nouvelle instance avec « new ». Cet attribut reste néanmoins accessible comme les attributs classiques utilisés jusqu'ici, via tous les objets de la classe, lesquels se référeront à la même zone mémoire, la zone mémoire unique réservée pour cet attribut statique. pour cet attribut statique.

5m 1s

