Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-03-2-IterationsApprofondissementExemple-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Boucles for. Instruction de l'incrémentaion. Exemples d'autres formes. Instruction d'incrémentation. Boucle for. Variable p. Deuxième exemple. Tour de boucle. Variable i. Instruction de l'affichage. Tel cas. Dernier exemple. Instruction vide. Sortir de la boucle. Intérieur du corps de la boucle.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en Java).)

vers la vidéo

EPFL

Itérations : approfondissement et exemples
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Commençons cette vidéo, avec des exemples d'autres

notes

résumé

0m 1s

Exemples d'autres formes de boucles `for`

```
for(int p = 0; p < 10; p += 2) {
   ...
```

formes de «boucles for». Cette «boucle for» commence par

notes

résumé

0m 9s

Exemples d'autres formes de boucles `for`

```
for(int p = 0; p < 10; p += 2) {
   ...
```

déclarer une variable p et l'initialise à zéro. La condition est «p strictement inférieur à 10» et l'instruction de l'incrémentaion est «p += 2». Et je vous rappelle qu'écrire «p += 2», c'est équivalent à écrire «p = p + 2», c'est-à-dire qu'on va ajouter 2 à la variable p, à chaque tour de boucle. p va donc, commencer à la valeur 0. À la fin du premier tour de boucle, on va ajouter 2 à p ; p va, donc, prendre la valeur 2. Ensuite, p va prendre la valeur 4. Ainsi de suite, jusqu'à ce

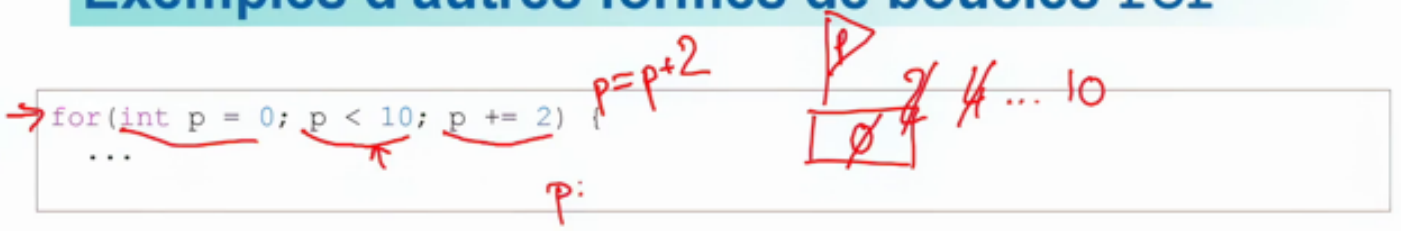
notes

résumé

0m 14s

Exemples d'autres formes de boucles `for`

```
for(int p = 0; p < 10; p += 2) {
   ...
```



que p prenne la valeur 10. Dans ce cas, la condition devient fausse, et on va sortir de la boucle. Donc, dans ce cas, p va prendre les

notes

résumé

0m 49s

Exemples d'autres formes de boucles `for`

EPFL

```
for(int p = 0; p < 10; p += 2) {
  ...
```

la variable `p` prendra les valeurs de 0, 2, 4, 6, 8 (`p += 2` est équivalent à `p = p + 2`);

```
for(int k = 10; k > 0; --k) {
  ...
```

la variable `k` prendra les valeurs 10, 9, 8 ... jusqu'à 1;

```
for(int i = 0; i >= 0; ++i) {
  ...
```

la condition est toujours vraie (du moins dans le principe).
La boucle est répétée indéfiniment et la variable `i` prendra toutes les valeurs positives que le type `int` peut représenter

valeurs 0, 2, 4, 6 et 8. Dans ce deuxième exemple, on commence par déclarer une variable k initialisée à 10. La condition est «k > 0». Donc, faites attention, on a changé le sens de comparaison, ici. Et l'instruction d'incrémentation est «--k», qui est équivalent à écrire «k = k -1», c'est-à-dire qu'on va retrancher 1 à la variable k, à chaque tour de boucle. k part donc, de la valeur 10. À la fin du premier tour de boucle, k vaut 9, ensuite 8, ainsi de suite, jusqu'à ce que k prenne la valeur 0. Dans ce cas, la condition devient fausse, et on sort de la boucle. Donc k, dans cet exemple, va prendre la valeur 10, 9, 8, ..., jusqu'à 1. Dans ce dernier exemple, on commence par déclarer une variable i, initialisée à 0. La condition est «i ≥ 0», et l'instruction d'incrémentation est «++i» qui va ajouter 1 à chaque tour de boucle. i commence donc, à la valeur 0. Ensuite, i va prendre la valeur 1. La condition va être vraie. Et ensuite, i va donc prendre la valeur 2. La condition va être toujours vraie. Et en fait, dans ce cas, la condition sera toujours vraie, du moins dans le principe. Et on ne va jamais sortir de la boucle. Alors, pourquoi «du moins dans le principe» ? Pour des raisons techniques, que je ne

notes

résumé

0m 56s

Boucles infinies

Une boucle `for` peut ne pas s'arrêter, ce qui se produit quand la condition est toujours vraie.
Plusieurs causes sont possibles:

1. On s'est trompé sur la condition:
   Par exemple:

```
for(int i = 0; i > -1; ++i) { // !!!
```

vais pas vous détailler, ce n'est pas complètement vrai, mais on va tout de même répéter la boucle un très grand nombre de fois.

notes

résumé

2m 37s

Boucles infinies

Une boucle `for` peut ne pas s'arrêter, ce qui se produit quand la condition est toujours vraie. Plusieurs causes sont possibles:

1. On s'est trompé sur la condition:

   Par exemple:

```
for(int i = 0; i > -1; ++i) { // !!!
```

2. On s'est trompé sur l'incrémentation:

```
for(int i = 0; i < 10; ++j) { // !!!
```

`j` est incrémenté au lieu de `i`, `i` garde donc toujours la valeur `0`, et la boucle ne s'arrête pas.

Une «boucle for» peut donc ne pas s'arrêter, si la condition est toujours vraie. Alors, plusieures causes sont possibles. Comme dans l'exemple précédent, peut-être qu'on s'est trompé sur la condition, comme ici. Alors, encore une fois, ce n'est pas complètement vrai. Dans, dans un tel cas, on va tout de même sortir de la boucle, pour des raisons techniques qu'on ne va pas détailler ici, mais après un très grand nombre d'itérations.

notes

résumé

2m 48s

Pas de point-virgule (;) à la fin de l'instruction `for`

```
for(int i = 0; i < 10; ++i);
  System.out.println("bonjour");
```

Une deuxième erreur possible, est de se tromper sur l'incrémentation. Par exemple, ici, ma condition est sur la variable i. Et je me suis trompé, j'incrémente la variable j. Comme i est initialisée à 0, cette condition va toujours être vraie, et on ne va jamais sortir de la boucle.

notes

résumé

3m 15s

Pas de point-virgule (;) à la fin de l'instruction `for`

Les instructions suivantes n'affichent qu'une seule fois la chaine "bonjour":

```
for(int i = 0; i < 10; ++i);
  System.out.println("bonjour");
```

Le point-virgule seul est considéré comme une instruction (qui ne fait rien).

Le corps de la boucle est donc constitué de cette instruction qui ne fait rien:

```
for(int i = 0; i < 10; ++i)
  ;
System.out.println("bonjour");
```

Passons, maintenant, à quelques erreurs commises fréquemment par les débutants. Notez, tout d'abord, qu'il n'y a pas de point-virgule à la fin de la ligne qui commence par le mot clé «for», c'est-à-dire qu'il n'y a pas de point-virgule, ici. Mais il se trouve que le compilateur va accepter votre programme, si vous mettez tout de même un point-virgule. Mais votre programme va se comporter de façon un peu surprenante. Plus exactement, avec le point-virgule, ce code va n'afficher qu'une seule fois le mot "bonjour". Alors, pourquoi ? C'est parce qu'en fait, le point-virgule que j'ai mis, ici, va être considéré comme l'instruction vide, une instruction qui ne fait rien. Et ce code va être interprété de la façon suivante,

notes

résumé

3m 37s

Attention aux accolades

```
for(int i = 0; i < 5; ++i)
  System.out.println("i = " + i);
  System.out.println("Bonjour");
```

affiche:

c'est-à-dire que seul le point-virgule se trouve à l'intérieur de la boucle. Cette boucle va donc répéter dix fois l'instruction vide, c'est-à-dire ne rien faire. L'instruction de l'affichage qui est, ici, se trouve, en fait, à l'extérieur de la boucle, et va donc n'être exécutée qu'une seule fois. Et donc, ce code affiche une seule fois, le mot "bonjour".

notes

résumé

4m 25s

Attention aux accolades

```
for(int i = 0; i < 5; ++i)
    System.out.println("i = " + i);
    System.out.println("Bonjour");
```

affiche:

```
i = 0
i = 1
i = 2
i = 3
i = 4
Bonjour
```

Il faut, également, fait attention aux accolades. Par exemple, ce code-ci peut produire cet affichage ; c'est-à-dire que cette instruction va être répétée cinq fois, mais cette instruction va n'être exécutée qu'une seule fois. Alors, pourquoi ?

notes

résumé

4m 52s

Attention aux accolades

```
for(int i = 0; i < 5; ++i)
    System.out.println("i = " + i);
    System.out.println("Bonjour");
```

affiche:

```
i = 0
i = 1
i = 2
i = 3
i = 4
Bonjour
```

Interprétation:

```
for(int i = 0; i < 5; ++i)
    System.out.println("i = " + i);
System.out.println("Bonjour");
```

Parce que ce code va être interprété de la façon suivante ; c'est-à-dire que cette instruction est à l'intérieur de la boucle, mais celle-ci est en dehors de la boucle.

notes

résumé

5m 8s

Evitez de modifier une variable compteur à l'intérieur d'une boucle `for`

```
for(int i = 0; i < 10; ++i) {
  ...
  if (...)
    --i; // !!!
}
```

Attention, donc ! L'indentation, c'est-à-dire le fait de déplacer ces deux lignes sur la droite, ici, ne suffit pas à faire que les deux instructions soient dans la boucle. Si je veux mettre les deux instructions dans la boucle, il me faut mettre une accolade ouvrante avant la première instruction, une accolade fermante après la dernière instruction. Et dans ce cas-là, j'obtiendrai l'affichage i égal zéro, Bonjour, i égal un, Bonjour, et cetera. Evitez, également, de modifier la valeur de la variable qui permet de contrôler le nombre de tours de boucle, à l'intérieur du corps de la boucle ;

notes

résumé

5m 25s

Evitez de modifier une variable compteur à l'intérieur d'une boucle `for`

```
for(int i = 0; i < 10; ++i) {
  ...
  if (...)
    --i; // !!!
}
```

1. Ça ne fera sans doute pas ce que vous voulez: n'oubliez pas que la boucle `for`, de son côté, incrémente la variable `i`.

c'est-à-dire, évitez de faire comme dans cet exemple, où j'ai commencé par déclarer une variable i, et l'initialiser à zéro, que j'utilise dans la condition et dans l'instruction d'incrémentation, et que je modifie également, cette variable i, à l'intérieur du corps de la boucle. Alors, pourquoi il faut éviter de faire ce genre de choses ?

notes

résumé

6m 13s

C'est parce que premièrement, ça ne fera sans doute pas ce que vous voulez. N'oubliez-pas que la «boucle for», avec l'instruction d'incrémentation, va de son côté, modifier la valeur de la variable i. Et deuxièmement, ça va être difficile, pour un relecteur humain, de comprendre ce que fait le programme, puisqu'il ne va pas forcément s'attendre à ce que la variable définie, déclarée dans la «boucle for», soit modifiée dans le corps de la boucle. de la boucle.

notes

résumé

6m 39s