Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-03-4-Boucles-pt2

Concepts (extraits des sous-titres générés automatiquement) :

Évaluation de la condition de sortie. Boucle conditionnelle. Corps de la boucle. Condition de continuation de la boucle. Condition d'arrêt. Exécution de l'instruction. Exécution de cette ligne. Variable i. Variante de la boucle. Évaluation de cette condition. Boucle. Situation analogue. Condition de continuation. Principe de fonctionnement. Paire d'accolades ouvrante.

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

vers la vidéo

EPFL

Boucles
(Partie 2)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

La boucle "do while" que nous venons de voir est une boucle conditionnelle dont l'évaluation de la condition de sortie se fait a posteriori. On a, au moins une fois exécuté le corps

notes

résumé

0m 1s

L'instruction `while...`

Il existe également la forme suivante:

```
while (condition) {
  bloc
}
```

corps

Le principe est similaire à celui de la boucle `do...while` que nous venons de voir.

La différence est que la condition est testée **avant** d'entrer dans la boucle. Si la condition est fausse, les instructions dans la boucle ne sont donc pas exécutées.

de la boucle, avant de savoir s'il faut arrêter les traitements, ou les répéter. Dans certaines situations, il est nécessaire de pouvoir tester la condition d'arrêt a priori, avant même d'avoir exécuté une seule fois le corps de la boucle. Dans ce cas-là, il faut avoir recours à une variante de la boucle "do while", la boucle "while", que nous allons voir maintenant. Si l'on veut décrire une boucle "while" en Java, on a recours à la syntaxe suivante : donc, on utilise le mot réservé "while", suivi immédiatement de la condition de continuation de la boucle; ensuite, entre une paire d'accolades ouvrante et fermantes, il y a toujours le corps de la boucle, le bloc d'instructions que l'on souhaite répéter.

notes

résumé

0m 13s

Exemples

```
int i = 100;
do {
  System.out.println("bonjour");
} while (i < 10);
```

affichera une fois `bonjour`.

Dans les 2 cas,
la condition `i < 10` est fausse.

```
int i = 100;
while (i < 10) {
 System.out.println("bonjour");
}
```

n'affichera rien.

Le principe de fonctionnement est analogue à celui d'une boucle "do while", on répète les traitements tant que l'évaluation de cette condition retourne "true", la différence avec la boucle "do while" est que l'on va, évaluer la condition de sortie, au moment même où on aborde l'exécution de l'instruction "while". Si cette condition est d'emblée évaluée à "false", on n'exécutera jamais le bloc d'instructions.

notes

résumé

0m 49s

Exemples

```
int i = 100;
do {
  System.out.println("bonjour");
} while (i < 10);
```

affichera une fois `bonjour`.

```
int i = 100;
while (i < 10) {
 System.out.println("bonjour");
}
```

n'affichera rien.

Dans les 2 cas,
la condition `i < 10` est fausse.

Donc, illustrons maintenant sur deux petits exemples basiques, la différence fondamentale entre une boucle "do while" et une boucle "while". Nous avons ici affaire à deux boucles très similaires, dont la condition de continuation, dépend dans les deux cas, d'une variable i, initialisée dans les deux cas, à la valeur 100. Commençons par examiner ce qui se passe dans le cas de la boucle "do while". Lorsque nous abordons l'exécution de cette ligne, rien ne nous empêche d'entrer directement dans le corps de la

notes

résumé

1m 13s

Exemples

```
int i = 100;
do {
  System.out.println("bonjour");
} while (i < 10);
```

affichera une fois `bonjour`.

```
int i = 100;
while (i < 10) {
 System.out.println("bonjour");
}
```

n'affichera rien.

Dans les 2 cas,
la condition `i < 10` est fausse.

boucle, et à ce moment-là, le message "bonjour", sera affiché sur le terminal. Nous abordons ensuite l'exécution de cette ligne, où la condition de continuation de la boucle est évaluée. Comme i vaut 100, évidement cette condition est évaluée à "false", et dans ce cas-là, nous

notes

résumé

1m 37s

Exemples

```
int i = 100;
do {
  System.out.println("bonjour");
} while (i < 10);
```

affichera une fois bonjour.

```
int i = 100;
while (i < 10) {
 System.out.println("bonjour");
}
```

n'affichera rien.

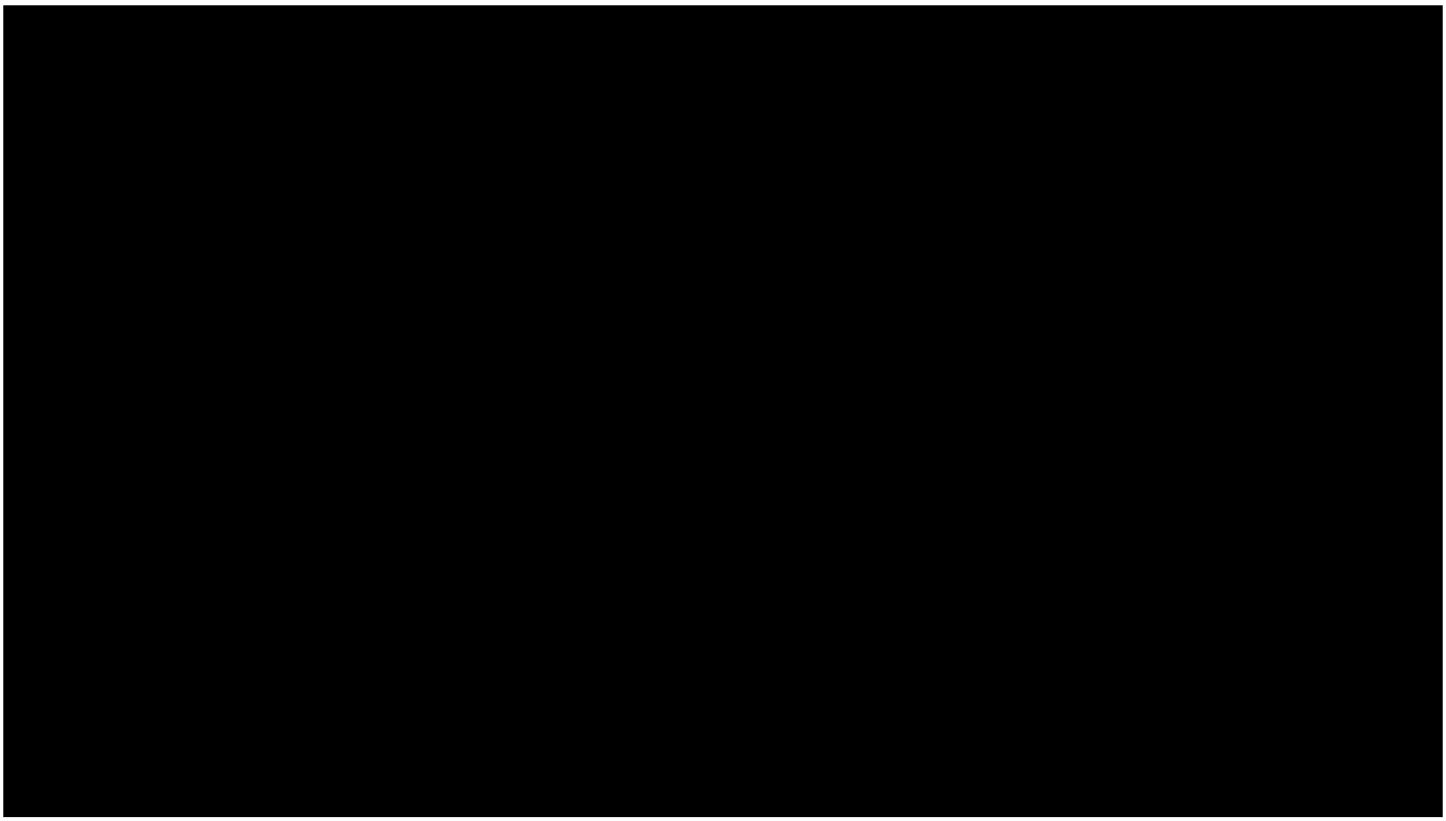
Dans les 2 cas,
la condition `i < 10` est fausse.

allons immédiatement sortir de la boucle, et éventuellement continuer l'exécution des instructions qui suivraient le "do while". Entre-temps, le message "bonjour" aura été affiché. Situation analogue, mais cette fois avec une boucle "while". Lorsque nous abordons l'exécution de cette ligne, nous allons d'emblée évaluer la condition de continuation, qui, comme dans le premier cas est évaluée à "false". Donc vous noterez, que dans les deux cas, la condition de continuation de la boucle, est évaluée à "false".

notes

résumé

1m 49s



Ici, puisque la condition est fausse, nous n'allons pas du tout entrer dans le corps de la boucle, mais continuer l'exécution après le corps de la boucle "while". Ce qui signifie que dans ce cas, la boucle n'affichera rien. Donc, vous voyez qu'on a affaire à deux boucles très similaires, dont l'une, dans un cas affichera "bonjour" et l'autre n'affichera rien. "bonjour" et l'autre n'affichera rien.

notes

résumé

2m 13s