Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-03-4-Boucles-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Boucles conditionnelles. Formulation de la condition d'arrêt. Valeur inférieure. Genre d'erreurs. Nombre d'itérations. Sortir de la boucle. Fin de la condition d'une boucle. Erreur syntaxique. Condition d'arrêt. Introduction de cette séquence. Boucles de type. Traitements de façon répétitive. Intérieur du corps de la boucle. Chose suivante. Nombre d'étudiants.

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

vers la vidéo

EPFL

Boucles
(Partie 3)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Les erreurs les plus communes, relatives aux boucles conditionnelles sont sans doute liées à la formulation de la condition d'arrêt. La condition d'arrêt est mal formulée et l'on ne sort, par exemple, jamais de la boucle, ou on en sort trop tôt, on n'exécute jamais la boucle, alors qu'on souhaiterait le faire, ce genre d'erreurs. Il existe cependant une erreur syntaxique commune de base, que je vais décrire maintenant, et dans laquelle on tombe

notes

résumé

0m 1s

Erreurs classiques

Il n'y a pas de `;` à la fin de la condition du `while...`:

```
while (i < 10); // !!
  ++i;
```

Le point-virgule est considéré comme le corps de la boucle, et l'instruction `++i` est **après la boucle**.

Si `i` est inférieur à 10, on entre dans la boucle pour ne jamais en ressortir puisque la valeur de `i` ne sera jamais modifiée.

sera interprété comme

```
i = 1
while(i < 10){
  ; } vide
++i;
```

En revanche, il y a un point-virgule à la fin du `do..while`:

```
do {
  ++i;
} while(i < 10);
```

assez souvent, si on débute avec les boucles de type "while" et "do while". Alors l'erreur est la suivante : imaginez que, à la fin de la condition d'une boucle "while", je tape par inadvertance un point-virgule. Cette erreur est d'autant plus facile à commettre qu'il y a un point-virgule à la fin d'un "do while", à la fin de la condition d'un "do while", mais ce point-virgule est là pour clore toute l'instruction "do while". Il n'a pas vraiment sa place ici, puisque après la condition "do while", nous nous attendons à trouver un corps qui fait quelque chose. Ici, vraisemblablement, nous voulions mettre dans le corps le ++i. Or, ce qui va se passer est la chose suivante : donc au moment où j'exécute l'instruction "while", tout se passe comme si j'avais un corps qui contient une instruction unique, qui est vide. Donc, imaginons que j'aie, au préalable, initialisé mon i, à une valeur inférieure à 10, par exemple 1, comme ici, le ++i, est à l'extérieur, rien à l'intérieur du corps de la boucle, ne va permettre de faire évoluer le i, de sorte à lui faire atteindre une valeur qui lui permette de sortir de la boucle, et du coup, le problème est que l'on va cycler indéfiniment dans cette boucle "while", ce qui peut être un comportement un peu déconcertant, lorsque l'on débute. Donc soyez attentifs à ce genre de petits détails, pour ne pas tomber dans ce piège, lorsqu'on débute avec la programmation des boucles "while" et "do while".

notes

résumé

0m 25s

Vous connaissez désormais plusieurs façons d'effectuer des traitements de

notes

résumé

1m 37s

Quand utiliser la boucle `while` ?
Quand utiliser la boucle `for` ?

EPFL

Quand le nombre d'itérations (de répétitions) est connu avant d'entrer dans la boucle, utiliser `for`:

```
for(int i = 0; i < nombre_d_iterations; ++i) {
```

Sinon, utiliser `while`:

– quand les instructions doivent être effectuées au moins une fois, utiliser `do...while`:

```
do {
  instructions;
} while (condition);
```

– Sinon, utiliser la forme `while`...

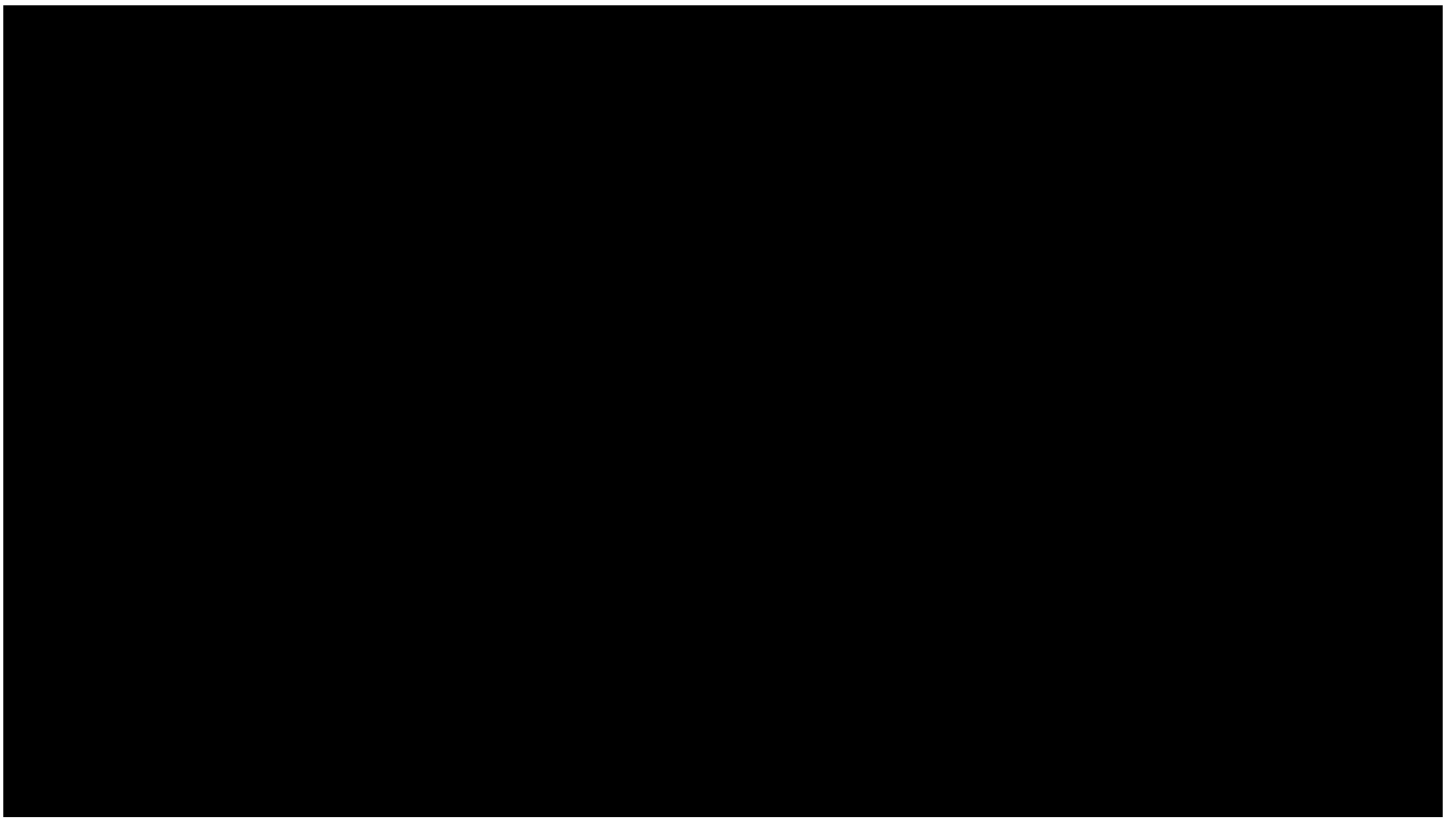
```
while (condition) {
  instructions;
}
```

façon répétitive, la boucle "for", la boucle "do while", la boucle "while"; et la question va se poser de savoir comment choisir entre l'une ou l'autre des formes. En fait, le choix entre l'une ou l'autre des formes, relève de principes relativement simple, ce que nous allons voir maintenant. Comme nous l'avons vu dans l'introduction de cette séquence, lorsque je connais mon nombre d'itérations, a priori, le choix va naturellement se porter sur une boucle "for". Imaginez par exemple, que je souhaite calculer, la moyenne des notes de tous les étudiants suivant ce cours. Le nombre d'étudiants suivant ce cours est connu a priori, c'est lui qui va constituer mon nombre d'itérations, et donc naturellement, je vais porter mon choix sur une boucle "for". Si je ne connais pas mon nombre d'itérations, alors le choix va se porter sur une instruction, une boucle conditionnelle de type "while" ou "do while". Alors, quand les instructions doivent être effectuées au moins une fois, comme nous l'avons vu tout à l'heure, nous allons typiquement, utiliser plutôt une boucle "do while". Alors dans quelles situations ça a lieu? Typiquement, un exemple très fréquent, est celui de l'interaction avec l'utilisateur. J'ai besoin de demander une valeur à l'utilisateur, et cette valeur doit répondre à certains critères; et j'ai besoin que l'utilisateur introduise au moins une fois cette valeur, avant de pouvoir tester si les critères sont rencontrés, si la condition est satisfaite ou non.

notes

résumé

1m 39s



Et dans ce cas-là, je vais très naturellement, porter mon choix sur une boucle "do while". Dans toutes les autres situations, si je n'exécute le traitement, que dans le cas où la condition est vérifiée, et bien, j'ai recours à une boucle "while", avec évaluation de la condition à priori. condition à priori.

notes

résumé

2m 49s