Support de cours

Cours:

## Initiation à la programmation (en Java)

Vidéo:

## Init-JAVA-03-4-Boucles-pt1

Concepts (extraits des sous-titres générés automatiquement) :

**Boucles conditionnelles. Nombre de notes. Nombre de répétitions. Instructions particulières. Fonction de conditions. Lignes d'instructions. Nombre de notes voulues. Observation importante. Nouvelle façon. Dernière observation. Instruction particulière. Corps de la boucle. Séquences vidéo précédentes. Boucles de type. Nombre de notes atteint.**

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

vers la vidéo

EPFL

# Boucles
## (Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Dans les séquences vidéo précédentes, nous avons abordé la

notes

résumé

0m 1s

EPFL

Les itérations, ou boucles `for`, permettent de répéter une partie du programme.

Elles sont utilisées quand le nombre de répétitions est connu *avant* d'entrer dans la boucle.

Selon le problème à résoudre, il arrive qu'on ne connaîsse pas combien de fois la boucle devra être exécutée.

On utilise alors une boucle conditionnelle, ou boucle `do..while / while`.

notion de structure de contrôle. Nous avons vu, qu'il s'agissait d'instructions particulières qui permettent d'exécuter des traitements en fonction de conditions, ou qui permettent de répéter des traitements. Nous allons aujourd'hui continuer sur cette lancée et aborder une nouvelle façon de répéter des traitements, que l'on appelle les boucles conditionnelles. Lorsque dans un programme, on souhaite répéter une séquence de traitements, et si le nombre de répétitions à exécuter, est connu à priori, c'est-à-dire, avant même d'entrer dans la boucle, nous avons vu qu'il était possible de recourir aux boucles "for", c'est-à-dire la notion d'itération. Il existe cependant des situations où l'on ne sait pas a priori, combien de fois on veut répéter le traitement. Typiquement, on va pouvoir être amené à répéter un traitement, tant qu'une condition est vérifiée, et on ne sait pas alors combien de répétitions vont être nécessaires, jusqu'à ce que la condition cesse d'être vérifiée. Dans ce cas, il faut avoir recours à d'autres types de structures de contrôle, ce que l'on appelle les boucles conditionnelles, c'est-à-dire des répétitions dont l'arrêt va dépendre

notes

résumé

0m 6s

```java
System.out.println("Entrez le nombre de notes");
int nombreDeNotes = clavier.nextInt();

double somme = 0;

if (nombreDeNotes > 0) {
  for(int i = 1; i <= nombreDeNotes; ++i) {
    System.out.println("Entrez la note numero " + i);
    double note = clavier.nextDouble();
    somme = somme + note;
  }

  System.out.println("Moyenne = " + somme / nombreDeNotes);
}
```

d'une condition. Plus précisément, cela va prendre la forme, en Java, de boucles de type "do while" ou "while" qui vont nous occuper dans la séquence d'aujourd'hui.

notes

résumé

1m 1s

```
System.out.println("Entrez le nombre de notes");
int nombre_de_notes = clavier.nextInt();

double somme = 0;

if (nombre_de_notes > 0) {
  for(int i = 1; i <= nombre_de_notes; ++i) {
    System.out.println("Entrez la note numero " + i);
    double note = clavier.nextDouble();
    somme = somme + note;
  }

  System.out.println("Moyenne = " + somme / nombre_de_notes);
}
```

Comment forcer l'utilisateur à entrer un nombre de notes supérieur à 0 ?

Pour illustrer notre propos, reprenons notre petit exemple de calcul de la moyenne d'un certain nombre de notes. Nous ne savons pas a priori de combien de notes nous devons calculer une moyenne, et donc nous commençons par le demander à l'utilisateur au travers d'une interaction clavier. Une fois le nombre de notes voulues saisi, et si ce nombre de notes est strictement positif, nous entamons alors un traitement répétitif, une boucle "for", qui va permettre de saisir toutes ces notes; via toujours une interaction clavier, et d'intégrer la note nouvellement saisie à la somme afin d'en calculer la moyenne plus tard. Une fois le nombre de notes atteint, lorsque nous avons saisi toutes les notes, nous sortons de la boucle "for", nous calculons la moyenne des notes, et nous l'affichons. Si l'on observe cet exemple d'un peu plus près, il peut sembler peu naturel de permettre à l'utilisateur de saisir un nombre de notes qui soit zéro. Ça n'a pas beaucoup de sens, et donc la question que nous nous posons naturellement,

notes

résumé

1m 9s

```java
int nombreDeNotes;

do {
   System.out.println("Entrez le nombre de notes");
   nombreDeNotes = clavier.nextInt();
} while(nombreDeNotes <= 0);
```

est la suivante : comment forcer l'utilisateur, le contraindre à fournir un nombre de notes supérieur à zéro? L'idée sous-jacente ici, serait donc de répéter ces traitements, jusqu'à ce que l'utilisateur, consente à introduire, un nombre de notes qui soit strictement positif. La solution pour répéter ces deux lignes d'instructions, tant que l'utilisateur n'a pas introduit un nombre de notes strictement positif, est d'avoir recours à une instruction particulière, une boucle conditionnelle, de type "do while". Alors comment se rédige ce genre de structure de contrôle?

notes

résumé

2m 1s

```java
int nombreDeNotes;
do {
  System.out.println("Entrez le nombre de notes");
  nombreDeNotes = clavier.nextInt();
} while(nombreDeNotes <= 0);
```

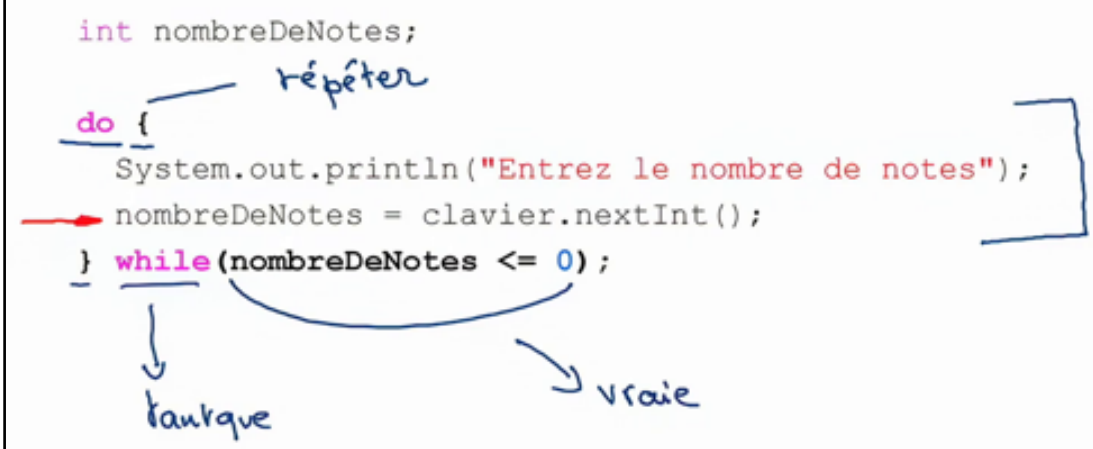
répéter

Nous avons le mot réservé "do", qui veut dire

notes

résumé

2m 35s




répéter, le mot "while", qui veut dire "tant que". Que veut-t'on répéter? Et bien nous voulons répéter l'exécution de ces deux instructions, qui se trouvent entre ces deux accolades, et nous voulons répéter ce traitement, tant que cette condition est vraie. Donc, si l'on imagine que l'utilisateur saisisse ici, un nombre de notes qui

notes

résumé

2m 39s

# Pas-à-pas

```
int nombreDeNotes;

do {
  System.out.println("Entrez le nombre de notes");
  nombreDeNotes = clavier.nextInt();
} while(nombreDeNotes <= 0);
```

soit inférieur ou égal à zéro, et bien l'évaluation de cette condition va retourner, "true", vrai, ce qui voudra dire qu'on continue à répéter les traitements, et donc nous allons repartir sur un nouveau cycle, d'exécution de la boucle. Examinons pas à pas, comment se

notes

résumé

3m 1s

# Pas-à-pas

```
int nombreDeNotes;

do {
    System.out.println("Entrez le nombre de notes");
    nombreDeNotes = clavier.nextInt();
} while(nombreDeNotes <= 0);
```

corps

0

n

déroulerait l'exécution d'une telle instruction "do while". Donc ici, nous commençons par déclarer une variable, qui contient le nombre de notes à saisir, et ensuite nous abordons directement l'exécution de notre instruction "do while". Ici, rien de particulier, on va directement entamer l'exécution de ce que l'on appelle le corps de la boucle, c'est-à-dire les instructions que l'on est censé pouvoir répéter si nécessaire. Donc ici, on va demander à notre utilisateur de saisir le nombre de notes, et imaginons dans le cas présent, que l'utilisateur saisisse zéro. Nous abordons donc ici, cette ligne, où il va être question d'évaluer la condition de sortie de la boucle. Ici comme le nombre de notes est égal à zéro, cette condition va être évaluée à "true", ce qui veut dire que l'on va

notes

résumé

3m 15s

# Pas-à-pas

```
int nombreDeNotes;

do {
  System.out.println("Entrez le nombre de notes");
  nombreDeNotes = clavier.nextInt();
} while(nombreDeNotes <= 0);
```

corps

0

-1

itérer à nouveau, on va répéter le traitement. Nous entamons alors une seconde itération de la boucle, nous demandons à nouveau à l'utilisateur de saisir un nombre de notes, et imaginons que notre utilisateur, cette fois-ci, saisisse un -1. Nous abordons à nouveau l'instruction où il faudra évaluer la condition de sortie; à nouveau, comme le nombre de notes est inférieur ou égal à zéro, elle sera évaluée à "true", et à nouveau, donc, nous allons répéter le traitement.

notes

résumé

4m 1s

# Pas-à-pas

```
int nombreDeNotes;

do {
    System.out.println("Entrez le nombre de notes");
    nombreDeNotes = clavier.nextInt();
} while(nombreDeNotes <= 0);
```

corps

false

0

-1

6

À la troisième itération de la boucle, nous continuons à demander un nombre de notes à notre utilisateur, qui, de guerre lasse, va finir par comprendre ce qu'on lui demande, et donner un nombre de notes strictement positif. Lorsque nous abordons l'instruction où il va être nécessaire d'évaluer la condition de sortie, cette fois, l'évaluation de la condition de sortie va retourner "false", puisque le nombre de notes est strictement positif, dans ce cas-là, nous cessons d'itérer, et nous allons donc sortir de notre boucle "do while", ce qui veut dire concrètement, que nous allons poursuivre l'exécution des traitements, après la fin de l'exécution de la boucle "do while" signalée par ce point virgule, et donc entamer ici, l'exécution d'éventuels traitements,

notes

résumé

4m 31s

# Syntaxe de l'instruction `do...while`

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

- Comme pour l'instruction `if`:
  - La condition peut utiliser des opérateurs logiques.
  - Les parenthèses autour de la condition sont obligatoires.
- Les instructions à l'intérieur de la boucle `do...while` sont toujours exécutées au moins une fois.
- Si la condition ne devient jamais fausse, les instructions dans la boucle sont répétées indéfiniment !

qui se trouveraient après la boucle "do while". Nous voyons dans le cas de cet exemple, qu'il n'est pas possible d'anticiper le nombre de fois, le nombre d'itérations à exécuter, avant de pouvoir sortir de la boucle. En effet, le nombre de répétitions doit dépendre de la volonté de l'utilisateur, qu'on ne peut pas anticiper ici. Une boucle "for" n'aurait donc pas été adaptée au traitement que l'on veut mettre en oeuvre dans le cas présent. Un peu plus formellement maintenant : la syntaxe de l'instruction "do while" en Java. Vous avez donc les mots réservés "do" et "while", qui encadrent ce que l'on appelle le corps de la boucle, un bloc d'instructions que l'on va pouvoir répéter, lequel se trouve entre deux accolades ouvrantes et fermantes. Suivant le mot réservé "while", il y a la condition de continuation de la boucle, nous allons répéter l'exécution du corps de la boucle, tant que cette condition est évaluée à "true". En Java, tout comme pour l'instruction "if", les parenthèses autour de la condition sont des éléments de syntaxe obligatoires, et vous devez clore votre instruction "do while", par le fameux petit point-virgule, à la fin de l'instruction. Tout comme pour l'instruction "if", la condition d'arrêt de la boucle, peut se formuler de façon relativement sophistiquée, notamment, en ayant recours aux opérateurs logiques. Par exemple, on peut imaginer que l'on va répéter les traitements tant qu'une variable x a la même valeur que y plus z, et que z est différent de 0, ou que, y est inférieur à 3 et ainsi de suite. Donc, nous voyons que nous pouvons formuler des conditions aussi sophistiquées que nous le souhaitons, en ayant recours aux opérateurs logiques, notamment. Une observation importante à faire à propos de la boucle "do while", est que son corps est toujours exécuté au moins une fois. En effet, dans notre petit exemple pas à pas de tout à l'heure,

notes

résumé

5m 8s

# Syntaxe de l'instruction `do...while`

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

- Comme pour l'instruction `if`:
  - La condition peut utiliser des opérateurs logiques.
  - Les parenthèses autour de la condition sont obligatoires.
- Les instructions à l'intérieur de la boucle `do...while` sont toujours exécutées au moins une fois.
- Si la condition ne devient jamais fausse, les instructions dans la boucle sont répétées indéfiniment !

nous avons vu que, lorsque nous abordons l'exécution d'une instruction "do while", nous rentrons directement dans le corps de la boucle.

notes

résumé

Ce qui veut dire que nous allons exécuter, la toute première fois que nous entrons dans la boucle, le bloc d'instructions, ensuite tout va dépendre de l'évaluation de la condition. Le bloc d'instructions ne sera exécuté qu'une seule fois, si à la première itération, l'évaluation de la condition retourne "false". Dans ce cas-là effectivement, nous allons sortir de la boucle "do while" et continuer les traitements après, mais entre temps, nous aurons, au moins une fois, exécuté le bloc d'instructions. Ceci constitue une caractéristique importante de la boucle conditionnelle de type "do while". Une dernière observation que l'on peut faire à propos de la boucle "do while" est que si, la condition est formulée de sorte à ce qu'elle ne devienne jamais fausse, et bien, le corps de la boucle va être répété indéfiniment. Donc il est important d'être rigoureux, attentif à la façon de formuler la condition, de sorte à pouvoir sortir de la boucle au moment voulu. la boucle au moment voulu.

notes

résumé

7m 1s