maestris BTS

Programmation Java
5. Boucles et Tableaux

BTS Services Informatiques aux Organisations

Sciences-U CAMPUS LYON

eductive EDUCATION GROUP

5.1 Les boucles

5.1.1 Les boucles conditionnelles

5.1 Les boucles

Les boucles

En programmation, **une boucle** est une technique qui permet de répéter une ou plusieurs instructions sans avoir à retaper le même ensemble d'instructions plusieurs fois.

5.1.1 Les boucles conditionnelles

Les **boucles conditionnelles** sont également appelées boucles `while`.

Il en existe deux types : `while` ; `do ... While`.

SYNTAXE

```
while (condition) {
  // bloc de code à exécuter
}
```

SYNTAXE

```
do {
  // bloc de code à exécuter
} while (condition)
```

5.1 Les boucles

La boucle while

La boucle doit continuer **tant que** la condition reste vraie.

Dans l'exemple ci-dessous, le code de la boucle s'exécutera, encore et encore, tant qu'une variable (i) est inférieure à 5 :

```
int i = 0;
while (i < 5) {
        System.out.println(i);
        i++;
}
```

Remarque : N'oubliez pas d'augmenter la variable utilisée dans la condition, sinon la boucle ne se terminera jamais !

La boucle do . . . while

La boucle **do... while** est très similaire à la première, mais la condition est placée à la **fin du bloc** de code correspondant. De cette façon, le bloc de code sera **toujours exécuté au moins une fois** :

```
int i = 0;
do {
        System.out.println(i);
        i++;
}
while (i < 5);
```

Remarque : N'oubliez pas d'augmenter la variable utilisée dans la condition, sinon la boucle ne se terminera jamais !

5.1.2 Les boucles énumérées

5.1.2 Les boucles énumérées

Les **boucles énumérées** sont des boucles qui sont utilisées si vous savez à l'avance combien de fois vous voulez faire une boucle.

En Java, cela s'appelle des boucles `for`. Avec elles, vous pouvez indiquer le **nombre d'itérations** à effectuer :

SYNTAXE

```
for (initialisation; condition; modification) {
  // bloc de code à exécuter
}
```

5.1 Les boucles

SYNTAXE

```
for (initialisation; condition; modification) {
  // bloc de code à exécuter
}
```

- L'initialisation est une expression qui s'initialise au début de la boucle. Elle déclare et assigne généralement un itérateur. Par exemple, nous déclarons un itérateur nommé i de type int avec une valeur de 0 : `int i = 0;` .
- La condition est l'expression qui est évaluée avant chaque exécution de boucle. Si elle est évaluée sur « false », la boucle s'arrête. Par exemple : `i < 5;`.
- La modification ou l'incrément est une expression qui est évaluée chaque fois qu'une itération de la boucle est effectuée. Par exemple par incrémentation de l' itérateur i : `i++;` .
- Le bloc de code à exécuter est située entre `{` et `}` . C'est la liste des instructions qui est exécuté chaque fois que la boucle est exécutée.

5.1 Les boucles

Rappel : Le **booléen** peut prendre deux valeurs : `true` ou `false`.

- Dans la boucle **for**, tant que la condition de terminaison est **true**, la boucle tourne. Dès que la condition n'est plus vérifiée, elle est donc **false**, et on sort de la boucle !
- Les variables itératives peuvent avoir n'importe quel nom. Les noms **i , j , k** sont souvent utilisés par exemple.
- Sinon, il est préférable, comme toujours, d'utiliser des noms descriptifs !

5.1 Les boucles

Voici un exemple d'une boucle `for` qui répète une instruction dix fois :

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

RÉSULTAT

```
0. Bonjour
1. Bonjour
2. Bonjour
3. Bonjour
4. Bonjour
```

Un autre exemple qui n'imprimera que les valeurs paires entre 0 et 10 :

```
for (int i = 0; i <= 10; i = i + 2) {
  System.out.println(i);
}
```

RÉSULTAT

```
0
2
4
6
8
10
```

Boucles imbriquées

Il est également possible de placer une boucle à l'intérieur d'une autre boucle. C'est ce qu'on appelle une boucle **imbriquée**.

La "boucle interne" sera exécutée une fois pour chaque itération de la "boucle externe":

```
// boucle externe
for (int i = 1; i <= 2; i++) {
  System.out.println("Externe: " + i); // S'exécute 2 fois

  // Boucle Interne
  for (int j = 1; j <= 3; j++) {
    System.out.println(" Interne: " + j); // S'exécute 6 fois (2 * 3)
  }
}
```

RÉSULTAT

```
Externe: 1
 Interne: 1
 Interne: 2
 Interne: 3
Externe: 2
 Interne: 1
 Interne: 2
 Interne: 3
```

5.1 Les boucles

Attention, une mauvaise condition de terminaison ou un mauvais incrément peut entraîner une **boucle infinie** !

La "boucle interne" sera exécutée une fois pour chaque itération de la "boucle externe":

```
for (int i = 0; i <= 0; i--) {
    System.out.println("i sera toujours inférieur à 0!");
}
```

Exercice 1 :

Utilisez une boucle `for` pour imprimer "**Oui**" 10 fois.

Exercice 1 :

Utilisez une boucle `for` pour imprimer "**Oui**" 10 fois.

CORRIGÉ

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

5.2 Les tableaux

5.2 Les tableaux

Les tableaux sont utilisés pour stocker **plusieurs** valeurs dans une seule variable, au lieu de déclarer des variables distinctes pour chaque valeur.

- Les tableaux permettent de stocker un ensemble **fini** d'éléments d'un **type** particulier.
- L'accès à un élément particulier se fait grâce à son **indice**.
- Le **premier** élément d'un tableau possède l'indice **0**.

Même si leur déclaration est spécifique, ce sont des **objets** : ils sont donc dérivés de la classe **Object**. Il est possible d'utiliser les méthodes héritées.

5.2 Les tableaux

Pour déclarer un **tableau**, définissez le type de la variable entre **crochets** :

SYNTAXE

```
String[] apprentis;
```

Nous venons de déclarer une variable contenant un **tableau de chaînes**. Pour y insérer des valeurs, vous pouvez placer les **valeurs** dans une liste séparée par des **virgules**, à l'intérieur **d'accolades**.

On appellera cela « initialisation explicite d'un tableau » :

SYNTAXE

```
String[] apprentis = {"Jean", "Emma", "Louis", "Lucas"};
```

Pour créer un **tableau d'entiers**, vous pouvez écrire :

SYNTAXE

```
int[] monTableau = {10, 20, 30, 40};
```

5.2 Les tableaux

Accéder aux éléments d'un tableau

Pour accéder à un **élément** du tableau, on utilisera le nom de la variable ainsi que son **indice** (une valeur numérique indiquant la **position** de l'élément dans le tableau) comprit entre **deux crochets**.

Cette instruction accède à la valeur du premier élément dans « `apprentis` » :

```
String[] apprentis = {"Jean", "Emma", "Louis", "Lucas"};
System.out.println(apprentis[0]);
// la sortie : Jean
```

Remarque : Les indices de tableau commencent par 0 : [0] est le premier élément. [1] est le deuxième élément, etc.

5.2 Les tableaux

Modifier un élément de tableau

Pour modifier la valeur d'un élément spécifique du tableau, on fait une affectation en utilisant le nom de la variable (tableau) ainsi que l'**indice** de l'élément comprit entre **deux crochets**.

Cette instruction modifie la valeur du deuxième élément dans « `apprentis` » :

```
String[] apprentis = {"Jean", "Emma", "Louis", "Lucas"};
apprentis[1] = "Jules";
System.out.println(apprentis[1]);
// la sortie : Jules
```

5.2 Les tableaux

La taille d'un tableau

Il est possible de demander la taille d'un tableau via l'attribut (la méthode) **length** : celui-ci est en lecture seule. Il ne sera donc plus possible de changer sa taille après son initialisation.

Voici un exemple d'utilisation d'un tableau de 10 valeurs **entières**. Ici le tableau est défini via une liste d'initialisation (la séquence de valeurs comprises entre les deux accolades).

```java
// Le tableau est initialisé avec 10 valeurs (les multiples de 10).
int [] monTableau = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };

// Quelle est la taille du tableau monTableau ?
System.out.println("Taille == " + monTableau.length);

// On accède à certaines valeurs de ce tableau
System.out.println(monTableau[0]);       // Affiche 10
System.out.println(monTableau[5]);       // Affiche 60
System.out.println(monTableau[9]);       // Affiche 100
```

RÉSULTAT

```
Taille == 10
10
60
100
```

5.2 Les tableaux

Les différentes syntaxes possibles

Pour introduire une variable de type tableau il vous faut utiliser la syntaxe [ ]. Néanmoins les crochets peuvent être placés à différents endroits. Le tableau ci-dessus vous propose quelques exemples de définition de tableaux Java. L'initialisation d'un tableau peut s'effectuer soit via une liste d'initialisation, soit via un appel à l'opérateur new.

Définition de tableau	Explications complémentaires
`int [] tb;`	Définit un tableau d'entiers non encore initialisé.
`int [] tb = new int[10];`	Définit un tableau de dix entiers. Il n'est plus possible de changer la taille de ce tableau.
`int [] tb1, tb2;`	Comme les crochets sont placés à gauche des noms de variables ils s'appliquent à toutes ces variables. On est donc en train de définir deux tableaux d'entiers, tous deux non initialisés.
`int [] tb1 = new int[5], tb2 = new int[10];`	Définit un premier tableau de cinq entiers, puis un second de dix entiers.
`int tb[] = new int[10], value = 10;`	Définit un tableau de dix entiers ainsi qu'une variable (value) de type entier. Les crochets ne s'appliquent uniquement qu'à *tb* car ils sont placés à la droite du nom de la variable.
`String [] strValues = new String[5];`	Définit un tableau pouvant contenir jusqu'à cinq chaînes de caractères.
`String [] strValues = { "toto", "titi", "tata" };`	Définit un tableau de chaînes de caractères contenant les trois valeurs stockées entre les accolades : on parle de liste d'initialisation.

5.2 Les tableaux

Exercice 2 :

1. Créer un tableau de type **String** appelé « apprentis » avec l'initialisation explicite de 5 prénoms.

SYNTAXE

```
type[] nomDeTableau = {valeur1, valeur2, etc…};
```

2. Imprimez le deuxième élément du tableau « apprentis ».

SYNTAXE

```
System.out.println(variable);
```

.

5.2 Les tableaux

Exercice 2 :

1. Créer un tableau de type **String** appelé « apprentis » avec l'initialisation explicite de 5 prénoms.

CORRIGÉ

```
String[] apprentis = {"Jean", "Emma", "Louis", "Lucas", "Jules"};
```

2. Imprimez le troisième élément du tableau « apprentis ».

CORRIGÉ

```
System.out.println(apprentis[2]);
```

.

5.2 Les tableaux

Parcourir un tableau : avec un for traditionnel

Il existe plusieurs manières de parcourir un tableau. Nous allons en voir deux : Parcours avec un **for** traditionnel et Parcours avec un **foreach**.

Pour parcourir un tableau, il est possible d'utiliser l'instruction **for**. Dans cet exemple on se contente d'afficher la valeur stockée dans chaque cellule du tableau (la valeur à l'indice `i`).

```
 // Déclaration et initialisation du tableau
int [] monTableau = { 10, 20, 30, 40, 50 };

// Parcours du tableau
for (int i = 0; i < monTableau.length; i++) {
    System.out.println(monTableau[i]);
}
```

RÉSULTAT

```
10
20
30
40
50
```

5.2 Les tableaux

Parcourir un tableau : avec un foreach

ATTENTION : On parlent de l'instruction « for each » mais, il n'y a pas de mot clé foreach qui existe en Java. Le mot clé associé est `for`, mais avec une **syntaxe** légèrement différente.

L'intérêt d'un « for each », est que vous n'avez pas à gérer l'indice de boucle.

```
// Déclaration et initialisation du tableau
int [] monTableau = { 10, 20, 30, 40, 50 };

System.out.println( "*** Parcours du tableau avec un for traditionnel ***" );
for (int i = 0; i < monTableau.length; i++) {
        int valeur = monTableau[i];
        System.out.println(valeur);
}
System.out.println( "*** Parcours du tableau avec un for each ***" );
for (int valeur : monTableau) {
        System.out.println(valeur);
}
```

Parcourir un tableau : avec un foreach

Ce qui diverge, entre les deux syntaxes, c'est surtout ce qui est placé entre les parenthèses.

- Au lieu d'y gérer un **indice de parcours**, on dit simplement qu'on veut extraire toutes les données du tableau mentionné à la droite du caractère « `:` ».
- Le parcours sera, bien entendu, réaliser en commençant de la **première donnée** et jusqu'à **la dernière**.
- A la gauche du caractère « `:` », on spécifie le type (`int`) et le nom (`valeur`) de la variable qui va recevoir, **une à une**, ces données.

5.2 Les tableaux

Exercice 3 :

1. Créer un tableau de type **String** appelé « apprentis » avec l'initialisation explicite de 5 prénoms. 2. Afficher tous les éléments du tableau « apprentis », en utilisant un parcours avec un **for** traditionnel et un parcours avec un **foreach**.

SYNTAXE

```
for (initialisation; condition; modification) {
  // bloc de code à exécuter
}
```

SYNTAXE

```
for (type variable : nomDeTableau) {
  // bloc de code à exécuter
}
```

5.2 Les tableaux

Exercice 2 :

1. Créer un tableau de type **String** appelé « apprentis » avec l'initialisation explicite de 5 prénoms. 2. Afficher tous les éléments du tableau « apprentis », en utilisant un parcours avec un **for** traditionnel et un parcours avec un **foreach**.

CORRIGÉ

```
// Déclaration et initialisation du tableau
String[] apprentis = {"Jean", "Emma", "Louis", "Lucas", "Jules"};
System.out.println( "*** Parcours avec un for ***" );
for(int i = 0; i < apprentis.length; i++) {
        System.out.println(apprentis[i]);
}
System.out.println( "*** Parcours avec un for each ***" );
for(String prenom : apprentis) {
        System.out.println(prenom);
}
```

5.2 Les tableaux

Tableau de tableaux (ou tableau multidimensionnels)

Un tableau multidimensionnel est un **tableau de tableaux**.

- Les tableaux multidimensionnels sont utiles lorsque vous souhaitez stocker des données sous forme de tableau, comme un tableau avec **des lignes et des colonnes**.
- Pour créer un tableau à **deux dimensions**, ajoutez chaque tableau dans son propre ensemble d'accolades :

SYNTAXE

```
type[][] nomDeTableau = { {valeur1, valeur2, etc.}, {valeur1, valeur2, etc.}, ... };
```

Exemple de tableau bidimensionnel avec deux tableaux en tant qu'éléments.

```
int[][] monTableau = { {1, 2, 3, 4}, {5, 6, 7} };
```

5.2 Les tableaux

Pour accéder aux éléments du tableau « **monTableau** », spécifiez **deux indices** : un pour le tableau et un pour l'élément à l'intérieur de ce tableau.

Cet exemple accède au troisième élément (2) du deuxième tableau (1) de « **monTableau** » :

```
int[][] monTableau = { {1, 2, 3, 4}, {5, 6, 7} };
System.out.println(monTableau[1][2]); // Affiche 7
```

N'oubliez pas que : Les indices de tableau commencent par 0 : [0] est le premier élément. [1] est le deuxième élément, etc.

5.2 Les tableaux

Boucle sur un tableau multidimensionnel

Nous pouvons également utiliser une boucle **for** à l'intérieur d'une autre boucle **for** pour obtenir les éléments d'un tableau à deux dimensions (nous devons toujours pointer vers **les deux indices**):

```
int[][] monTableau = { {1, 2, 3, 4}, {5, 6, 7} };
for (int i = 0; i < monTableau.length; ++i) {
        for(int j = 0; j < monTableau[i].length; ++j) {
                System.out.println(monTableau[i][j]);
        }
}
```

Exercice

Supposons que nous avons un tableau de 2 dimensions représentant les notes des 4 élèves dans 3 différentes matières. Nous voulons calculer la moyenne des notes de chaque élève et afficher les résultats.

N'oubliez pas que : Les indices de tableau commencent par 0 : [0] est le premier élément. [1] est le deuxième élément, etc.

5.2 Les tableaux

CORRIGÉ

```java
// Initialisation du tableau de notes
double[][] notes = {{12.5, 14.0, 13.0}, {15.0, 16.0, 18.5}, {10.0, 9.5, 11.0}, {11.5, 13.0, 12.0}};

// Calcul de la moyenne de chaque élève
double[] moyennes = new double[notes.length];
        for (int i = 0; i < notes.length; i++) {
                double somme = 0;
                for (int j = 0; j < notes[i].length; j++) {
                        somme += notes[i][j];
                }
                moyennes[i] = somme / notes[i].length;
        }

// Affichage des résultats
for (int i = 0; i < notes.length; i++) {
        System.out.println("Moyenne de l'élève " + (i+1) + " : " + moyennes[i]);
}
```

5.2 Les tableaux

Exercice

Supposons que nous avons un tableau de 2 dimensions représentant les notes des 4 élèves dans 3 différentes matières. Nous voulons calculer la moyenne des notes de chaque élève et afficher les résultats.

Dans cet exemple, nous avons initialisé le tableau de notes en utilisant un tableau de 2 dimensions contenant les notes de chaque élève dans chaque matière. Ensuite, nous avons utilisé une boucle **for** imbriquée pour parcourir chaque note de chaque élève et calculer la moyenne de chaque élève.

La boucle extérieure parcourt les élèves (**notes.length**), tandis que la boucle intérieure parcourt les notes de chaque élève (**notes[i].length**). Nous utilisons une variable somme pour stocker la somme des notes de chaque élève, et nous calculons la moyenne de chaque élève en divisant cette somme par le nombre de notes de chaque élève.

Nous avons également créé un tableau « **moyennes** » pour stocker les moyennes de chaque élève. Enfin, nous avons utilisé une boucle **for** pour afficher les moyennes de chaque élève à la console.

maestris BTS

Merci pour l'attention !

Sources

- Programmation Java/Transtypage : https://fr.wikibooks.org/wiki/Programmation_Java/Transtypage
- Apprenez à programmer en Java – OpenClassrooms
- Java Tutorial : https://www.w3schools.com/java/
- Java Platform, Standard Edition Documentation : https://docs.oracle.com/en/java/javase/
- Le tutoriel/cours sur le langage de programmation Java : https://koor.fr/Java/Tutorial/Index.wp