Les conditions

Nous sommes en Programmation Objet, nous manipulons des classes des méthodes.

Mais dans une classe, une méthode, la lecture et l'exécution se fait de façon séquentielle, c'est-à-dire ligne par ligne.

Avec les conditions, nous allons pouvoir gérer différents cas de figure sans pour autant lire tout le code.

Tous vos projets ne sont que des enchaînements et des imbrications de conditions et de boucles

Avant de pouvoir créer et évaluer des conditions, vous devez savoir que pour y parvenir, nous allons utiliser ce qu'on appelle des *"opérateurs logiques"* . Ceux-ci sont surtout utilisés lors de conditions (si [test] alors [faire ceci]) pour évaluer différents cas possibles. Voici les différents opérateurs à connaître :

== : permet de tester l'égalité.

!= : permet de tester l'inégalité.

< : strictement inférieur.

<= : inférieur ou égal.

> : strictement supérieur.

>= : supérieur ou égal.

&& : l'opérateur ET. Il permet de préciser une condition

|| : le OU. Même combat que le précédent.

? : : l'opérateur ternaire. Pour celui-ci, vous comprendrez mieux avec un exemple plus tard.

La structure if… else

Les opérations en Java sont soumises à des priorités. Tous ces opérateurs se plient à cette règle, de la même manière que les opérateurs arithmétiques…

Imaginons un programme qui demande à un utilisateur d'entrer un nombre entier relatif (qui peut être soit négatif, soit nul, soit positif).

Les structures conditionnelles vont nous permettre de gérer ces trois cas de figure. La structure de ces conditions ressemble à ça

```
1 if(//condition)
2 {
3   //Exécution des instructions si la condition est remplie
4 }
5 else
6 {
7   //Exécution des instructions si la condition n'est pas remplie
8 }
```

Cela peut se traduire par ***"si… sinon… "*** .

Le résultat de l'expression évaluée par l'instruction if sera un boolean, donc soit true, soit false. La portion de code du bloc ***if*** ne sera exécutée que si la condition est remplie. Dans le cas contraire, c'est le bloc de l'instruction ***else*** qui le sera. Mettons notre petit exemple en pratique :

Dans un premier temps, la condition du ***if*** est testée. elle dit :
-si "***i***" est strictement inférieur à 0 alors fais ça.

Dans un second temps, vu que la condition précédente est fausse, le programme exécute le ***else***.

```
1  int i = 10;
2
3  if (i < 0)
4    System.out.println("le nombre est négatif");
5  else
6    System.out.println("le nombre est positif");
```

Ici vous remarquerez qu'il n'y a pas d'accolades. Alors que dans la présentation de if, il y en avait ???
les accolades sont présentes dans la structure *"normale"* des conditions, mais lorsque le code à l'intérieur de l'une d'entre elles n'est composé ***que d'une seule ligne***, les accolades deviennent facultatives.

```java
int i = 0;
if (i < 0)
{
  System.out.println("Ce nombre est négatif !");
}
else
{
  if(i == 0)
    System.out.println("Ce nombre est nul !");

  else
    System.out.println("Ce nombre est positif !");

}
```

Les 2 façons sont justes mais la deuxième est plus simple et à privilégier

Il faut absolument donner une condition au ***else if*** pour qu'il fonctionne.

```java
int i = 0;
if (i < 0)
  System.out.println("Ce nombre est négatif !");

else if(i > 0)
  System.out.println("Ce nombre est positif !");

else
  System.out.println("Ce nombre est nul !");
```

Les conditions multiples :

```java
int i = 58;
if(i < 100 && i > 50)
  System.out.println("Le nombre est bien dans l'intervalle.");
else
  System.out.println("Le nombre n'est pas dans l'intervalle.");
```

Ici, nous avons utilisé l'opérateur **&&**. La condition de notre ***if*** est devenue :

si "***i***" est inférieur à 100 ET supérieur à 50

Avec l'opérateur ***&&*** , la clause est remplie si et seulement si les conditions la constituant sont toutes remplies.
Si l'une des conditions n'est pas vérifiée, la clause sera considérée comme fausse.

Réfléchissez bien à l'intervalle que vous voulez définir!

```java
int i = 58;
if(i < 100 && i > 100)
  System.out.println("Le nombre est bien dans l'intervalle.");
else
  System.out.println("Le nombre n'est pas dans l'intervalle.");
```

Ici, la condition ne sera jamais remplie, car aucun nombre n'est à la fois plus petit et plus grand que 100 !

Le ***switch*** est surtout utilisé lorsque nous voulons des conditions "à la carte". Prenons l'exemple d'une interrogation comportant deux questions :

pour chacune d'elles, on peut obtenir uniquement 0 ou 10 points, ce qui nous donne au final trois notes et donc trois appréciations possibles, comme ceci :

0/20

10/20

20/20

Dans ce genre de cas, on utilise un ***switch*** pour éviter des ***else if*** à répétition et pour alléger un peu le code.

```
1 switch (/*Variable*/)
2 {
3    case /*Argument*/:
4       /*Action*/;
5       break;
6    default:
7       /*Action*/;
8 }
```

"Variable" évalué par le switch

Valeur possible de la "Variable"

Action si valeur "variable" vérifiée

Permet de quitter instantanément le switch

Sécurité si aucune actions n'est vérifiées,
une action par défaut sera exécutée.

```java
int note = 10; //On imagine que la note maximale est 20

switch (note)
{
  case 0:
    System.out.println("Ouch !");
    break;
  case 10:
    System.out.println("Vous avez juste la moyenne.");
    break;
  case 20:
    System.out.println("Parfait !");
    break;
  default:
    System.out.println("Il faut davantage travailler.");
}
```

Je n'ai écrit qu'une ligne de code par instruction case, mais rien ne vous empêche d'en mettre plusieurs.

Depuis la version 7 de Java, l'instruction ***switch*** accepte les objets de type String en paramètre. De ce fait, cette instruction est donc valide :

```java
String chaine = "Bonjour";

switch(chaine) {
  case "Bonjour":
    System.out.println("Bonjour monsieur !");
    break;
  case "Bonsoir":
    System.out.println("Bonsoir monsieur !");
    break;
  default:
    System.out.println("Bonjoir ! :p");
}
```

La condition ternaire :

Les conditions ternaires sont assez complexes et relativement peu utilisées. Surtout utilisées dans des programmes mathématiques.
La particularité de ces conditions réside dans le fait que trois opérandes (c'est-à-dire des variables ou des constantes) sont mis en jeu, mais aussi que ces conditions sont employées pour affecter des données à une variable.
Voici à quoi ressemble la structure de ce type de condition :

```
1  int x = 10, y = 20;
2  int max = (x < y) ? y : x ; //Maintenant, max vaut 20
```

Nous cherchons à affecter une valeur à notre variable max, mais de l'autre côté de l'opérateur d'affectation se trouve une condition ternaire…

Ce qui se trouve entre les parenthèses est évalué : x est-il plus petit que y ?

Donc, deux cas de figure se profilent à l'horizon :

si la condition renvoie true (vrai), qu'elle est vérifiée, la valeur qui se trouve après le ***?*** sera affectée ;

sinon, la valeur se trouvant après le symbole ***:*** sera affectée.

Vous pouvez également faire des calculs (par exemple) avant d'affecter les valeurs :

```
1  int x = 10, y = 20;
2  int max = (x < y) ? y * 2 : x * 2 ; //Ici, max vaut 2 * 20 donc 40
```

N'oubliez pas que la valeur que vous allez affecter à votre variable doit être du même type que votre variable.

Résumé :

--Les conditions vous permettent de n'exécuter que certains morceaux de code.

Il existe plusieurs sortes de structures conditionnelles :

la structure if... elseif... else

la structure switch... case... default

la structure ? :

--Si un bloc d'instructions contient plus d'une ligne, vous devez l'entourer d'accolades afin de bien en délimiter le début et la fin.

--Pour pouvoir mettre une condition en place, vous devez comparer des variables à l'aide d'opérateurs logiques.

--Vous pouvez mettre autant de comparaisons renvoyant un boolean que vous le souhaitez dans une condition.

--Pour la structure switch, pensez à mettre les instructions break; si vous ne souhaitez exécuter qu'un seul bloc case.