# LES VARIABLES ET LES OPERATEURS

## 1. Rappel

Une variable est un élément qui stocke des informations de toute sorte en mémoire : des chiffres, des résultats de calcul, des tableaux, des renseignements fournis par l'utilisateur… Nous avons vu un cours là-dessus quand on parlait de l'algorithmique au chapitre 1.

Vous n'êtes pas sans savoir que votre ordinateur ne parle qu'une seule langue : le binaire ! Le langage binaire est une simple suite de 0 et de 1. Vous devez vous rendre compte qu'il nous serait très difficile, en tant qu'êtres humains, d'écrire des programmes informatiques pour expliquer à nos ordinateurs ce qu'ils doivent faire, entièrement en binaire…Vous imaginez ! Des millions de 0 et de 1 qui se suivent ! Non, ce n'était pas possible ! De ce fait, des langages de programmation ont été créés afin que nous ayons à disposition des instructions claires pour créer nos programmes. Ces programmes sont ensuite compilés pour que nos instructions humainement compréhensibles soient, après coup, compréhensible par votre machine

Le langage binaire est donc une suite de 0 et de 1 qu'on appelle bit. Si vous êtes habitués à la manipulation de fichiers (audio, vidéos, etc.) vous devez savoir qu'il existe plusieurs catégories de poids de programme (Ko, Mo, Go, etc.). Tous ces poids correspondent au système métrique informatique. Le tableau suivant présente les poids les plus fréquemment rencontrés :

| Raccourcis | Traduction | Correspondance |
|---|---|---|
| b | Bit | C'est la plus petite valeur informatique : soit 0 soit 1 |
| o | Octet | regroupement de 8 bits, par exemple : 01011101 |
| Ko | Kilo Octet | regroupement de 1024 octets |
| Mo | Mega Octet | regroupement de 1024 ko |
| Go | Giga Octet | regroupement de 1024 Mo |
| To | Tera Octet | regroupement de 1024 Go |

## 2. Les différents types de variables

Nous allons commencer par découvrir comment créer des variables dans la mémoire. Pour cela, il faut les déclarer. Une déclaration de variable se fait comme ceci :

```
<type_de_la_variable>        <nom_de_la_variable> ;
```

Java code

Cette opération étant bien une instruction, se termine aussi par un point-virgule. Ensuite, on l'initialise en entrant une valeur. Les noms de variable bien entendu son des identificateurs.

En Java, nous avons deux types de variables :

1. Des variables de type simple ou « primitif » ;
2. Des variables de type complexe ou des « objets ».

Ce qu'on appelle des types simples ou types primitifs, en Java, ce sont tout bonnement des nombres entiers, des nombres réels, des booléens ou encore des caractères, et vous allez voir qu'il y a plusieurs façons de déclarer certains de ces types.

### Les variables de type numérique

- Le type **byte** (1 octet) peut contenir les entiers entre -128 et +127.

```
byte temperature;
temperature = 64;
```
Java code

- Le type **short** (2 octets) contient les entiers compris entre -32768 et +32767.

```
short vitesseMax;
vitesseMax = 32000;
```
Java code

- Le type **int** (4 octets) va de -2*10 9 à 2*10 9 (2 et 9 zéros derrière…ce qui fait déjà un joli nombre).

```
int temperatureSoleil;
temperatureSoleil = 15600000; //La température est exprimée en kelvins
```
Java code

- Le type **long** (8 octets) peut aller de à (encore plus gros…).

```
long anneeLumiere;
anneeLumiere = 9460700000000000L;
```
Java code

- Le type **float** (4 octets) est utilisé pour les nombres avec une virgule flottante

```
float pi;
pi = 3.141592653f;
```
Java code

- Le type **double** (8 octets) est identique à float, si ce n'est qu'il contient plus de chiffres derrière la virgule et qu'il n'a pas de suffixe.

```
double division; division =
0.33333333333333333333333333333333333333333333334d;
```
Java code

- Vous remarquerez que nous ne mettons pas une virgule, mais un point (.) ! Et vous remarquerez aussi que même si le nombre en question est rond, on écrit « .0 » derrière celui-ci, le tout suivi de « f » pour les **float**, « d » pour les **double**, « L » pour les **long**.

### Les variables stockant un caractère

- Le type **char** contient un caractère stocké entre apostrophes (« ' ' »), comme ceci :

```
char caractere;
caractere = 'A';
```
Java code

### Des variables de type booléen

Le type **boolean**, lui, ne peut contenir que deux valeurs : **true** (vrai) ou **false** (faux), sans guillemets (ces valeurs sont natives dans le langage, il les comprend directement et sait les interpréter).

```
boolean question;
question = true;
```

Java code

### Et aussi le type String

Le type String permet de gérer les chaînes de caractères, c'est-à-dire le stockage de texte. Il s'agit d'une variable d'un type plus complexe que l'on appelle objet. Vous verrez que celle-ci s'utilise un peu différemment des variables précédentes :

```
//Première méthode de déclaration
String phrase;
phrase = "Titi et Grosminet";

//Deuxième méthode de déclaration
String str = new String();
str = "Une autre chaîne de caractères";

//Troisième méthode de déclaration
String string = "Une autre chaîne";

//Quatrième méthode de déclaration
String chaine = new String("Et une de plus !");
```

Java code

- *Attention : String commence par une majuscule ! Et lors de l'initialisation, on utilise des guillemets doubles (« " " »). Cela a été mentionné plus haut : String n'est pas un type de variable, mais un objet. Notre variable est un objet, on parle aussi d'une instance : ici, une instance de la classe String. Nous y reviendrons lorsque nous aborderons les objets.*

En fait, comme String, les noms des classes commencent par une lettre majuscule. C'est juste une convention… Une habitude que les développeurs ont adoptée… Cette convention, la voici :

- Tous vos noms de classes doivent commencer par une majuscule ;
- Tous vos noms de variables doivent commencer par une minuscule ;
- Si le nom d'une variable est composé de plusieurs mots, le premier commence par une minuscule, le ou les autres par une majuscule, et ce, sans séparation ;
- Tout ceci sans accentuation !

Pour notre première classe créé plus haut, je n'avais pas fait la précision sur ce fait. Mais désormais, le nom d'une classe commence par une lettre majuscule.

Voici quelques exemples de noms de classes et de variables :

```
public class Toto{}
public class Nombre{}
public class TotoEtTiti{}
String chaine;
String chaineDeCaracteres;
int nombre;
int nombrePlusGrand;

/*Et lorsque nous avons plusieurs variables d'un même type, nous pouvons résumer tout ceci à une déclaration : */
int nbre1 = 2, nbre2 = 3, nbre3 = 0;
```

Java code

## 3. Les opérateurs arithmétiques

Les opérateurs arithmétiques sont ceux que l'on apprend à l'école primaire…ou presque :

- « + » : permet d'additionner deux variables numériques (mais aussi de concaténer des chaînes de caractères ; ne vous inquiétez pas, on aura l'occasion d'y revenir).
- « - » : permet de soustraire deux variables numériques.
- « * » : permet de multiplier deux variables numériques.
- « / » : permet de diviser deux variables numériques (mais je crois que vous aviez deviné).
- « % » : permet de renvoyer le reste de la division entière de deux variables de type numérique ; cet opérateur s'appelle le modulo.

### Quelques exemples de calcul

```
int nbre1, nbre2, nbre3; //Déclaration des variables
nbre1 = 1 + 3; //nbre1 vaut 4
nbre2 = 2 * 6; //nbre2 vaut 12
nbre3 = nbre2 / nbre1; //nbre3 vaut 3
nbre1 = 5 % 2; //nbre1 vaut 1, car 5 = 2 * 2 + 1
nbre2 = 99 % 8; //nbre2 vaut 3, car 99 = 8 * 12 + 3
nbre3 = 6 % 3; //là, nbre3 vaut 0, car il n'y a pas de reste
```

Java code

Maintenant on va voir d'autre opérations et leurs formes réduite.

```
int nbre1, nbre2, nbre3; //Déclaration des variables
/*Initialisation
*des variables
*/
nbre1 = 0 ;
nbre2 = 0 ;
nbre3 = 0;
nbre1 = nbre1 + 1; //nbre1 = lui-même, donc 0 + 1 => nbre1 = 1
nbre1 = nbre1 + 1; //nbre1 = 1 (cf. ci-dessus), maintenant, nbre1 = 1 + 1 = 2
nbre2 = nbre1; //nbre2 = nbre1 = 2
nbre2 = nbre2 * 2; //nbre2 = 2 => nbre2 = 2 * 2 = 4
nbre3 = nbre2; //nbre3 = nbre2 = 4
nbre3 = nbre3 / nbre3; //nbre3 = 4 / 4 = 1
nbre1 = nbre3; //nbre1 = nbre3 = 1
nbre1 = nbre1 - 1; //nbre1 = 1 - 1 = 0
nbre1 = nbre1 - 1; //nbre1 = 1 - 1 = 0
```

Java code

```
int nbre1, nbre2, nbre3; //Déclaration des variables
nbre1 = nbre2 = nbre3 = 0; /*Initialisation des variables*/
nbre1 += 1; //nbre1 = lui-même, donc 0 + 1 => nbre1 = 1
nbre1++; //nbre1 = 1 (cf. ci-dessus), maintenant, nbre1 = 1 + 1 = 2
nbre2 = nbre1; //nbre2 = nbre1 = 2
nbre2 *= 2; //nbre2 = 2 => nbre2 = 2 * 2 = 4
nbre3 = nbre2; //nbre3 = nbre2 = 4
nbre3 /= nbre3; //nbre3 = 4 / 4 = 1
nbre1 = nbre3; //nbre1 = nbre3 = 1
nbre1 -= 1; //nbre1 = 1 - 1 = 0
--nbre1; //nbre1 = 1 - 1 = 0
/*L'ajout de 1 (ou la réduction de 1) a une variable entière est appeler l'incrémentation (la décrémentation) … en forme réduite, on peut mettre « ++ » ou « -- » avant ou après la variable. On expliquera mieux dans un exercice*/
```

Java code

- *Très important : on ne peut faire du traitement arithmétique que sur des variables de même type sous peine de perdre de la précision lors du calcul. On ne s'amuse pas à diviser un int par un float, ou pire, par un char ! Ceci est valable pour tous les opérateurs arithmétiques et pour tous les types de variables numériques. Essayez de garder une certaine rigueur pour vos calculs arithmétiques.*

### La concaténation

La concaténation est une opération qui doit contenir au moins un opérateur objet String… C'est une opération au cour de laquelle une chaine de caractère est annexée à la suite de l'autre. L'opérateur est toujours **« + »**, et l'opération retourne toujours une chaine de caractère.

```
String salutation, nom, both;
salutation = "Hééé… bonjour " ;
nom = "Sadio Mané" ;
both = salutation + nom ; //both = « Hééé… bonjour Sadio Mané »
```

Java code

### L'instruction d'écriture

Vous vous rappeler surement des méthodes de l'objet out que nous avons vu : print() et println(). Et bien, ces méthodes sont aussi capables de prendre en paramètre une variable de type String.

```
String salutation, nom, both;
salutation = "Hééé… bonjour " ;
nom = "Sadio Mané" ;
both = salutation + nom ;
//afficher une variable String
System.out.println(both) ;
//afficher avec une concatenation
System.out.println(both + "." + " Comment va-tu ???") ;
```

Java code

```
Hééé… bonjour Sadio Mané. Comment va-tu ???
```

console code

## 4. Les conversions, ou « cast »

Comme expliqué précédemment, les variables de type **double** contiennent plus d'informations que les variables de type **int**. Ici, Nous allons voir un truc super important en Java. Vous serez amenés à convertir des variables. Ce type de conversion s'appelle une « conversion d'ajustement », ou cast de variable.

- D'un type **int** en type **float**, et inversement :

```
int i = 123;
float j = (float) i; //j=123

float x = 2.3f;
int y = (int) x; //y=2
```

Java code

- D'un type **int** en type **double**, et inversement :

```
int i = 123;
double j = (double) i; //j=123

double x = 2.3d;
int y = (int) x; //y=2
```

Java code

- D'un type **String** en type **numérique**, et inversement :

```
int i = 12;
String j = new String();
j = j.valueOf(i);
int k = Integer.valueOf(j).intValue();
/*valueOf(params) //ici params peut être en float, double…
* Integer.valueOf(j).intValue() //on peut aussi avoir
* Long.valueOf(i).longValue() //pour une conversion String to long
*…*/
```

Java code

- Vous devez savoir qu'en Java, comme dans d'autres langages d'ailleurs, il y a la notion de priorité d'opération ; et là, nous en avons un très bon exemple !
- Sachez que l'affectation, le calcul, le cast, le test, l'incrémentation…toutes ces choses sont des opérations ! Et Java les fait dans un certain ordre, il y a des priorités.

```java
int nbre1 = 3, nbre2 = 2;
double resultat = (double)(nbre1 / nbre2);
System.out.println("Le résultat est = " + resultat);
```

Java code

Dans le cas qui nous intéresse, il y a trois opérations :

- un calcul ;
- un cast sur le résultat de l'opération ;
- une affectation dans la variable resultat.

Eh bien, Java exécute notre ligne dans cet ordre ! Il fait le calcul (ici 3/2), il caste le résultat en double, puis il l'affecte dans notre variable resultat.

```
Le résultat est = 1.0
```

console code

Ceci dit pour celui qui aurait voulu obtenir 1.5 comme résultat, il fallait convertir les nombres en double avant d'effectuer la division (qui ne sera plus alors une division entière).

```java
int nbre1 = 3, nbre2 = 2;
double resultat = (double) nbre1 / (double) nbre2;
System.out.println("Le résultat est = " + resultat);
```

Java code

```
Le résultat est = 1.5
```

console code

## 5. Depuis Java 7 : le formatage des nombres

Comme vous le savez sûrement, le langage Java est en perpétuelle évolution. Les concepteurs ne cessent d'ajouter de nouvelles fonctionnalités qui simplifient la vie des développeurs. Ainsi dans la version 7 de Java, vous avez la possibilité de formater vos variables de types numériques avec un séparateur, l'underscore (_), ce qui peut s'avérer très pratique pour de grands nombres qui peuvent être difficiles à lire. Voici quelques exemples :

```java
double nombre = 10000000000000d; // cast en d
//Peut s'écrire ainsi double
nombre = 1____000____000___000_000d; // cast en d
//Le nombre d'underscore n'a pas d'importance

//Voici quelques autres exemple d'utilisation
int entier = 32_000;
double monDouble = 12_34_56_78_89_10d; // cast en d
double monDouble2 = 1234_5678_8910d; // cast en d
```

java code

Les underscore doivent être placés entre deux caractères numériques : ils ne peuvent donc pas être utilisés en début ou en fin de déclaration ni avant ou après un séparateur de décimal. Ainsi, ces déclarations ne sont pas valides :

Avant Java 7, il était possible de déclarer des expressions numériques en hexadécimal, en utilisant le préfixe « 0x» :

```java
int entier = 255; //Peut s'écrire « int entier = 0xFF; »
int entier = 20; //Peut s'écrire « int entier = 0x14; »
int entier = 5112; //Peut s'écrire « int entier = 0x13_F8; »
```

java code

Depuis java 7, vous avez aussi la possibilité d'utiliser la notation binaire, en utilisant le préfixe « 0b » :

```java
int entier = 0b1111_1111; //Est équivalent à : « int entier = 255; »
int entier = 0b1000_0000_0000; //Est équivalent à : « int entier = 2048; »
int entier = 0b100000000000; //Est équivalent à : « int entier = 2048; »
```

java code