# *Les Variables et Opérateurs en Java avec Eclipse*

## Les différents types de variables

Une déclaration de variable se fait comme ceci :

```
<Type de la variable> <Nom de la variable> ;
```

Cette opération se termine toujours par un point-virgule « ; » (comme toutes les instructions de ce langage). Ensuite, on l'initialise en entrant une valeur.

En Java, nous avons deux types de variables :

1. des variables de type simple ou « primitif » ;
2. des variables de type complexe (de référence )ou des « objets ».

Ce qu'on appelle des *types simples* ou *types primitifs*, en Java, sont des nombres entiers, des nombres réels, des booléens ou encore des caractères.

**Les variables de type numérique**

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

```
byte temperature;
temperature = 64;
```

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

```
short vitesseMax;
vitesseMax = 32000;
```

Le type int (4 octets) va de -2*109 à 2*109 (2 et 9 zéros derrière… ce qui fait déjà un joli nombre).

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

Le type long (8 octets) peut aller de $-9*10^{18}$ à $9*10^{18}$ (encore plus gros…).

```
long anneeLumiere;
anneeLumiere = 9460700000000000L;
```

Afin d'informer la JVM que le type utilisé est **long**, vous **DEVEZ** ajouter un "**L**" à la fin de votre nombre, sinon le compilateur essaiera d'allouer ce dernier dans une taille d'espace mémoire de type entier et votre code ne compilera pas si votre nombre est trop grand…

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

```
float pi;
pi = 3.141592653f;
```

Ou encore :

```
float nombre;
nombre = 2.0f;
```

Vous remarquerez que nous ne mettons pas une virgule, mais un point ! même si le nombre en question est rond, on écrit « .0 » derrière celui-ci, le tout suivi de « f ».

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

```
double division;
division = 0.33333333333333333333333333333333333333333333334d;
```

Ici encore, vous devez utiliser une lettre - le « d » - pour parfaire la déclaration de votre variable.

**Des variables stockant un caractère**

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

```
char caractere;
caractere = 'A';
```

**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;
```

**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*.

```
//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 ! »);
```

Attention : String commence par une majuscule ! Car il s'agit d'une convention de nommage. En fait, c'est une façon d'appeler les classes, les variables, etc. Et lors de l'initialisation, on utilise des guillemets doubles (« " " »).

Les opérateurs arithmétiques

Les opérateurs arithmétiques sont :

- « + » : permet d'additionner deux variables numériques (mais aussi de concaténer des chaînes de caractères).
- « - » : permet de soustraire deux variables numériques.
- « * » : permet de multiplier deux variables numériques.
- « / » : permet de diviser deux variables numériques.
- « % » : permet de renvoyer le reste de la division entière de deux variables de type numérique ; cet opérateur s'appelle le **modulo**.

Pour afficher le contenu d'une variable dans la console, il suffit d'utiliser l'opérateur « + » qui sert aussi d'opérateur de concaténation, c'est-à-dire qu'il permet de mélanger du texte brut et des variables. Voici un exemple d'affichage avec une perte de précision :

```
int nbre1 = 10, nbre2 = 3;
int resultat = nbre1 / nbre2;
System.out.println("Le résultat est = " + resultat);
```

## Les conversions, ou « cast »

D'un type int en type float :

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

D'un type int en double :

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

Et inversement :

```
double i = 1.23;
double j = 2.9999999;
int k = (int)i;       //k vaut 1
k = (int)j;           //k vaut 2
```

Ce type de conversion s'appelle une « conversion d'ajustement », ou *cast* de variable. Ce qui permet de passer directement d'un type int à un type double.

Il est aussi possible de caster le résultat d'une opération mathématique en la mettant entre « ( ) » et en la précédant du type de *cast* souhaité. Donc :

```
double nbre1 = 10, nbre2 = 3;
int resultat = (int)(nbre1 / nbre2);
System.out.println("Le résultat est = " + resultat);
```

Voilà qui fonctionne parfaitement. Pour bien faire, vous devriez mettre le résultat de l'opération en type double. Et si on fait l'inverse : si nous déclarons deux entiers et que nous mettons le résultat dans un double ? Voici une possibilité :

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

On obtient 1.0. On ne caste pas ici, car un double peut contenir un int.

```
int nbre1 = 3, nbre2 = 2;
double resultat = (double) nbre1 / nbre2 ;
System.out.println("Le résultat est = " + resultat);
//affiche : Le résultat est = 1.5
```

### Le formatage des nombres

On a la possibilité de formater les 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 :

```
double nombre = 1000000000000d; // 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
```

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.

### Remarque

- 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 !

### En résumé

- Les variables sont essentielles dans la construction de programmes informatiques.
- On affecte une valeur dans une variable avec l'opérateur égal (« = »).
- Après avoir affecté une valeur à une variable, l'instruction doit se terminer par un point-virgule (« ; »).
- Vos noms de variables ne doivent contenir ni caractères accentués ni espaces et doivent, dans la mesure du possible, respecter la convention de nommage Java.
- Vous pouvez *caster* un résultat en ajoutant un type devant celui-ci : (int), (double), etc.