Chapitre 2

Les variables

1. Variables

1.1 Les noms de variables

- Le premier caractère ne doit pas être un chiffre
  (ex : **`1DeuxTrois`** n'est pas correct).

- Aucun espace dans le nom

- Les majuscules sont différentes des minuscules :
  `Toto` ≠ **`toto`**

- Caractères ne devant pas être utilisés :

  & ~ " # ‘ { } ( ) [ ] - | ` \ ^ @ = % * ? : / § ! < > £ , ;

- Tout autre caractère peut être utilisé, y compris les caractères accentués et le caractère de soulignement "_"

- Le nombre de lettres composant le nom d’une variable est indéfini.

1.2 Les types simples de Java

4 catégories :

- Catégorie **logique**
- Catégorie **caractère**
- Catégorie **entier**
- Catégorie **réel** (flottant)

Ces types sont dits simples, car, à un instant donné, une variable de type simple ne peut contenir qu'une et une seule valeur.

Remarque

Nous rencontrerons par la suite des **types structurés** qui permettent le stockage, sous un même nom de variable, de plusieurs valeurs de même type ou non.

Il s'agit des tableaux, des classes, des vecteurs ou encore des dictionnaires.

Catégorie logique

Il s'agit du type **boolean**. Seulement deux états possibles : **`true`** (vrai) et **`false`** (faux).

Catégorie caractère

Deux types définissent cette catégorie :

- **String** : chaîne de caractères.

- **char** : caractère isolé. Codé en UNICODE sur deux octets → permet de représenter jusqu'à 2^{16} caractères.

Catégorie entier

Quatre types d'entiers :

- **byte** : 1 octet, de –128 à 127
- **short** : 2 octets, de –32768 à 32767
- **int** : 4 octets de –2 147 483 648 à 2 147 483 647
- **long** : 8 octets de – 9 223 372 036 854 775 808
  à 9 223 372 036 854 775 807.

Catégorie réel (flottant) :

Nombres à virgules, appelés nombres réels ou encore flottants.

- **float** : 4 octets, de 1.40239846 e-45F
  à 3.40239846 e38F
- **double** : 8 octets, de 4.9406564584124654 4 e-324D
  à 1.7976931348623157 0 e308D

En langage Java, toute valeur numérique réelle est définie par défaut en double précision (type **double**).

Par conséquent, la lettre **d** (ou **D**) placée en fin de valeur n'est pas nécessaire.

Par contre, dès que l'on utilise une variable **float**, la lettre **f** (ou **F**) est indispensable, sous peine d'erreur de compilation.

1.3 Déclarer une variable

En déclarant une variable, le programmeur donne le type et le nom de la variable.

Syntaxe :

```
type  nomdevariable;
```

ou

```
type  nomdevariable1, nomdevariable2;
```

Exemples :

```
int   test;

char  choix, temp;
```

1.4 Déclarer une constante

On place le mot clé final devant la déclaration de la variable :

Syntaxe :

```
final type  nomdeconstante;
```

Exemples :

```
final int   test = 10;
test = 2;              // ERREUR
```

Remarque

Il est possible d'initialiser les variables pendant la déclaration. Cela permet d'éviter des erreurs de compilation (le compilateur vérifiant que les variables sont bien initialisées).

Exemples :

```
float f1=0.0f,f2=1.2f; // initialisation de deux float
int test = 0; // initialisation d'une variable de type int
boolean OK = true; // initialisation d'un boolean
```

2. Les opérations arithmétiques

2.1 Les opérateurs

Opérateurs utilisés par le langage Java :

Symbole	Opération	Exemple
+	Addition	a = b+2
-	Soustraction	a = b-7
*	Multiplication	a = b*5
/	Division	a = b/5
%	Modulo	a = b%3

2.2 Le type d'une expression arithmétique

Attention au type des variables employées dans les expressions arithmétiques.

Terme	Opération	Terme	Résultat
Entier	+ - * / %	Entier	Entier
Réel	+ - * / %	Réel	Réel

Exemple :

Diviser deux entiers → résultat entier :

```
int x=7, y=2 ;
float z;
z=x/y;              // → z = 3, et pas 3.5
```

Casting

Java est un langage **fortement typé**.

Pour affecter un résultat d'un certain type dans une variable d'un autre type, il peut être nécessaire de faire un « **cast** » ou « **casting** », en plaçant entre parenthèse le nouveau type devant la variable à convertir.

Ex: Affectation :

```
int   x;
float y;

x = y;
```

Produit une erreur lors de la compilation, car il n'est pas possible de mettre un **`float`** dans un **`int`** (perte de précision). L'inverse est possible :

```
y = x;
```

Solution : faire un cast de **`y`** en **`int`** avant de l'affecter à **`x`** :

```
x = (int)y;
```

Ex: Diviser deux entiers :

```
int   x=7, y=2;
float res1, res2;

res1 = x/y;          // → res1 = 3
res2 =(float)x/y; // → res2 = 3.5

res2 =(float)(x/y);      // → res2 = 3
```

x est convertit en **float**, ce qui permet de faire une division réelle.

Ex: Résultat d'une fonction :

```
float res;

res = Math.sqrt(4);
```

→Produit une erreur lors de la compilation, car **`Math.sqrt()`** est une fonction retournant un **`double`**, et non pas un **`float`**. Il est possible de mettre un **`float`** dans un **`double`**, mais pas l'inverse (perte de précision).

→Solution : on cast le résultat de **`Math.sqrt()`** en **`float`** :

```
float res;

res = (float)Math.sqrt(4);
```