FICHE 3

LES TYPES - LES VARIABLES CORRESPONDANTES

(Fiches Java)

1 Introduction

En java, les variables, les objets qu'on manipule ont un **type**. Les types possibles sont les **types primitifs**, les **types construits**.

Les types primitifs sont les types qui définissent les caractères (***char***), les entiers (***int, long, short, byte***), les réels flottants (***float, double***), les booléens (***boolean***).

Les types construits sont **les tableaux** et les **classes**.

2 Les types primitifs

2.1 Les entiers signés

Il sont codés en **complément à 2**.

- Sur 1 octet pour les *byte*.
- Sur 2 octets pour les *short*.
- Sur 4 octets pour les *int*.
- Sur 8 octets pour les *long*.

Voici des exemples de déclarations :

```
int u ;
int n=-567 ;
long m = 46787 ;
short k=45 ;
byte v=-54 ;
```

2.2 Les réels flottants

Ils sont codés suivant la norme IEEE-754. Chaque flottant sauf 0 est écrit sous la forme normalisée

$$x = (-1)^s x_0.x_1 \cdots x_{p-1} 2^e,$$

où $s = 0$ *ou* 1 définit le signe, $x_0.x_1 \cdots x_{p-1}$ est la mantisse ($x_0 = 1$ les autres x_i valent 0 ou 1) et $-e_{max} + 1 \leq e \leq e_{max}$ l'exposant. On utilise l'exposant biaisé $E = e + e_{max}$. On les stocke de la façon suivante :

- Sur 32 bits pour les *short* (simple précision IEEE-754). On a alors le signe sur 1 bit, l'exposant biaisé sur 8 bits ($e_{max} = 127$), la mantisse de longueur 24 est elle stockée sur 23 bits car le bit x_0 valant toujours 1 n'est pas écrit. Le 0 est représenté par un exposant biaisé nul et par une mantisse nulle. On se reportera à la norme citée pour des informations complémentaires.
- Sur 64 bits pour les *long* (double précision IEEE-754. Le signe est sur 1 bit, l'exposant biaisé sur 11 bits ($e_{max} = 1023$), la mantisse de 53 bits est stockée sur 52 bits.

Voici des exemples de déclarations :

```
float x = 1.345f;
float y = 0.2E-6F;
float z;

double u = 1.567;
double v = 4.67d;
double w;
```

2.3 Les caractères

Ils sont stockés suivant la **norme Unicode** sur 2 octets. On écrit

```
char c='a';
```

2.4 Les booléens

Les booleens prennent deux valeurs *true* et *false* On peut déclarer

```
boolean b=false;
```

3 Les types construits

3.1 Les classes

Les classes sont des parties de codes qui définissent des familles d'objets ainsi que leurs comportements et les méthodes qui agissent sur eux. Un objet est typé par sa classe. Les classes sont écrites par l'utilsateur ou déjà définies dans des bibliothèques.
On peu déclarer un objet d'une classe MaClasse sous la forme :

```
MaClasse cl ;
```

On a alors déclaré la variable cl, mais on ne l'a pas créee. Nous verrons plus tard comment faire.

3.2 Les tableaux

Voici comment déclarer un tableau unidimensionnel dont les éléments on pour type MonType.

```
MonType[] tab ;
tab=new Montype[80] ;

MonType[] tab=new MonType[80] ;

int[] table={ 1, 34, -5, 22 } ;
```

On vient de définir un tableau pouvant contenir 80 éléments de type MonType, dans des cases indexée de 0 jusqu'à 79.
On peut récupérer la taille du tableau dans son champ *length* (on consulte tab.length, mais on ne peut pas modifier cette valeur). On verra plus loin en détail le comportement des tableaux.

4 Différences de comportement

Les variables d'un type primitifs sont stockées "par valeur". Les méthodes qui utilisent ces variables comme paramètres d'entrée font donc une copie de leur valeur, copie qu'elles modifient éventuellement et qui disparaît lorsque la méthode se termine. En conséquence, **les paramètres de type primitif ne sont jamais modifiés par les méthodes**.

En revanche, les types construits sont stockés "par référence". En conséquence les méthodes qui travaillent sur des paramètres de type construit **transmettent en sortie les modifications faites sur les valeurs de ces paramètres**.

Il est donc utile de disposer de classes qui **enrobent** certains types primitifs de manière à les stocker "par référence". Ce sont les classes ***Integer, Long, Boolean, Float, Double, Character***.