Java
Les Types Primitifs

Ahcène Bounceur

UBO
université de bretagne occidentale

La notion de type

- La mémoire centrale :
  - Un ensemble de "positions binaires" nommées **bits**
  - Chaque groupe de 8 bits représente un **octet** (byte)
  - Chaque octet est repéré par son **adresse**

La notion de type

- Chaque nombre est codé en binaire
- Additionner deux nombres nécessite avant tout de savoir comment les coder
- Un type sert à décider de la manière dont il faut coder un nombre

La notion de type

- En java, il existe 4 grandes catégories de types primitifs :
  - Les nombres entiers
  - Les nombres flottants
  - Les caractères
  - Les booléens
- Il existe aussi le type énuméré, mais comme il se repose sur la notion de classe, il ne constitue donc pas un type primitif

Les types entiers

- Il servent à représenter les nombres entiers relatifs
- Représentation mémoire :
  - Un bit pour le signe (0 : positif et 1 : négatif)
  - Les autres bits servent à représenter :
    - La valeur absolue du nombre pour les positifs
    - Le complément à deux du nombre pour les négatifs

Les types entiers

- Exemple :
  - type **short** (entier courts)
  - Représenté sur 16 bits
  - Cas d'un nombre **positif** :
    - 1 0000000000000001 0001
    - 2 0000000000000010 0002
    - 3 0000000000000011 0003
    - 16 0000000000010000 0010
    - 127 0000000001111111 007F
    - 255 0000000011111111 00FF

Les types entiers

- Exemple :
  - type **short** (entier courts)
  - Représenté sur 16 bits
  - Cas d'un nombre **négatif** :
    - –1 1111111111111111 FFFF
    - –2 1111111111111110 FFFE
    - –3 1111111111111101 FFFD
    - –4 1111111111111100 FFFC
    - –16 1111111111110000 FFF0
    - –256 1111111100000000 FF00

Les types entiers

- Remarque (type short) :
  - Le plus grand nombre positif : 32767
    - 0111111111111111 (7FFF)
  - Le plus petit nombre négatif : –32768
    - 1000000000000000 (8000)
- **Attention ! Le dépassement de capacité n'est jamais détecté**

32767 + 1 = –32768

Les différents types entiers

- Java dispose de 4 types d'entiers

Type	Taille (octets)	Valeur minimale	Valeur maximale
byte	1	-128 (Byte.MIN_VALUE)	127 (Byte.MAX_VALUE)
short	2	-32768 (Short.MIN_VALUE)	32767 (Short.MAX_VALUE)
int	3	-2 147 483 648 (Int.MIN_VALUE)	2 147 483 647 (Int.MAX_VALUE)
long	4	-9 223 372 036 854 775 808 (Long.MIN_VALUE)	9 223 372 036 854 775 807 (Long.MAX_VALUE)

Les différents types entiers

- **Attention** !
  - Pour les short, int et long, java ne gère pas entièrement leur taille. Cette dernière peut varier d'une machine à une autre
  - Exemple : le type entier int peut être codé sur 2 octets sur une machine, 4 octets sur une autre

Affecter une valeur entière

- Il existe plusieurs manières d'affecter une valeur entière :
  - Utilisation des constantes :
    - x = +9465
    - x = 38
    - x = -654
  - Utilisation des héxadécimales (0x… ou 0X…) :
    - x = +0x24F9 // x=9465
    - x = +0X26 // x = +0x0026 ou x = 38
    - x = -0xFD72 // x = -0x0000FD72 ou x = -654
    - x = 0xFFFFFD72 // x = -654

Les types flottants

- Ils servent à représenter, de manière approchée, une partie des nombres réels
- Les différents types flottants

Type	Taille (octets)	Précision (chiffres significatifs)	Valeur absolue minimale	Valeur absolue maximale
float	4	7	1.4E-45 Float.MIN_VALUE	3.4028235E38 Double.MIN_VALUE
double	8	15	4.9E-324 Float.MAX_VALUE	1.79769313486231 57E308 Double.MAX_VALUE

Affecter une valeur flottante

- x = 12.43
- x = -0.38
- x = -.38
- x = 4.
- x = .27
- x = 4.25E4
- x = 4.25e+4
- x = 42.5E3
- x = 42.57E-32
- x = 42E4
- x = 42.e12
- x = 42.57E-32
- x = -.570e-2

Affecter une valeur flottante

- Attention !
  - float x ;
  - x = 12.5 ; // **ERREUR !!!**
  - 12.5 affecter directement est considéré come étant un double. Un double n'est pas convertir implicitement en float
  - **Solutions :**
  - x = (float) 12.5 ; // cast
  - x = 12.5**f** ;

Le type caractère

- Ils sert à manipuler des caractères
- Codés sur 2 octets (Unicode)
- Parmi les 65536 combinaisons qu'offre unicode, on retrouve les 128 possibilités du code ASCII (7 bits) ainsi que les 256 possibilités du code ASCII/ANSI (latin 1) de windows
- Déclaration :

```
char c1, c2 ;
```

Affecter un caractère

- Affectation classique (constantes) :
  - Caractère graphiques
    - c = 'a' ;
    - c = 'E' ;
    - c = 'é' ;
    - c = '+' ;

Affecter un caractère

- Affectation classique (constantes) :
  - Caractères non graphiques

Notation	Code Unicode (hex)	Abréviation usuelle	Signification
\b	0008	BS (Backspace)	Retour arrière
\t	0009	HT (Horizontal tabulation)	Tabulation horizontale
\n	000a	LF (Line feed)	Saut de ligne
\f	000c	FF (Form Feed)	Saut de page
\r	000d	CR (Cariage return)	Retour charriot
\"	0022		
\’	0027		
\\	005c		

Affecter un caractère

- Affectation héxadécimale ('\uxxxx ');
  - c = '**\u0041**' ; // c=65 ou c='A'

Le type booléen

- Il sert à représenter une valeur logique du type vrai/faux (true/false)
  - if (n<p) …
  - boolean verif ;
  - verif = n<p ;
  - verif = true ;

Initialisation et constantes

- On peut écrire :
  - int n = 15 ;
- Ou encore :
  - int n ;
  - n = 15 ;
- Et aussi :
  - int n ;
  - n = 15 ;
  - int p = n*2;

Initialisation et constantes

- On ne peut écrire ! (erreur de compilation)
  - int n ;
  - System.out.println("n = "+n) ;
- Ou encore :
  - int n ;
  - int p = n+4 ;
- Et aussi :
  - int n ;
  - if (…) n=30;
  - int p = n*2;

Initialisation et constantes

- Mais on peut écrire
  - int n ;
  - if (…) n=30; else n=10;
  - int p = n*2;

Les constantes

- **final** int N = 20 ;
- N = N + 1 ; // **ERREUR !**
- On peut écrire :
  - final int N ;
  - N = 20 ;
- On peut écrire :
  - final int N ;
  - int p = 15 ;
  - N = p*2 ;

Les constantes

- On ne peut pas écrire :
  - final int N ;
  - int p = 15 ;
  - N = p*2 ;
  - …
  - N++ ; // ERREUR, seule la première affection est autorisée

Les constantes

- On peut pas écrire :
  - final int NOMBRE = 20 ;
  - …
  - final int LIMITE = 2 * NOMBRE + 3 ;

- Merci de votre attention