INF1256 Informatique pour les sciences de la gestion

– *Variables et opérateurs* –

Johnny TSHEKE, Ing. Jr.

Université du Québec à Montréal
Département d'Informatique
tsheke_shele.johnny@uqam.ca

Séance 03

Déclaration Variables en Java

- Déclaration simple : `TypeDonnée nomVariable;`
  Ex : `int j;`
- Avec Initialisation : `TypeDonnée nomVariable = VALEUR;`
  Ex : `int j = 0;`
- Plusieurs Variables : `TypeDonnée nomVariable1, nomVariableN;`
  Ex : `int i, j, k;`
  `int i = 12 , j = 4 , k = 0;`
  → Il est recommandé d'éviter les déclarations multiples

Types Variables en Java

Variable d'instance	: hors méthode sans le mot `static` (chaque instance a sa variable)
Variable de classe	: hors méthode avec le mot `static` (partagée par toutes les instances)
Constante classe	: hors méthode avec les mots `static` et `final` (partagée)
Variable locale	: dans une méthode ou bloc
Constante locale	: dans une méthode ou bloc et avec le mot `final`
Paramètre/argument	: en argument d'une méthode

⇒ on peut ajouter `public` ou `private` devant les variables d'instances ou de classe.

⇒ On peut aussi ajouter `potected` devant une variable d'instance

Exemples type variables Java Exemple.java

```
package inf1256s03;

public class Exemple {
        static final int CONSTANTE_DE_CLASSE = 10; // avec final static
        static int varClasse = 0; // avec le mot static -- tres rare
        int varInstance = 0; // sans le mot static

        public static void main(String[] args) {
                // la variable args est un paramètre d'une methode
                final int CONSTANTE_LOCALE = 5;
                int varLocale = 0; // dans un bloc
                System.out.println("valeur initale = " + varLocale);
                System.out.println("Une constante vaut: " + CONSTANTE_DE_CLASSE);
                System.out.println("Constante locale vaut: " + CONSTANTE_LOCALE);
        }
}
```

Types primitifs

Ces types sont prédéfinis dans le langage

byte	entier compris entre -128 et 127
short	entier compris entre -32,768 et 32,767
int	entier entre -2^{31} et $2^{31} - 1$. Sur Java 8, on peut utiliser la classe `Integer` pour les valeurs comprises entre 0 et $2^{32} - 1$
long	entier long -2^{63} et $2^{63} - 1$. En Java 8, `Long` entre 0 et $2^{64} - 1$. Valeur littérale avec suffixe l
float	flotant (simple précisions) Pas pour applications de grande précision – financières. suffixe littérale f ou F
double	double précisions. (pour les applications financières, utiliser `java.math.BigDecimal`). suffixe littérale d ou D
boolean	2 valeurs possibles : `true` et `false`
char	un seul caractère unicode compris entre `u0000` (0) et `uffff` (65,535)
String	chaine des caractères (Pas un type primitif mais une classe `java.lang.String`)

⇒ `https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html`

Type chaine des caractères (String)

- `import java.lang.*;`
- Déclaration : `String s;`
- valeure litérale : `"Chaine"`
- Concaténation : `+`
- comparaison

```
String s1="Merci";
String s2="merci";
System.out.println(s1.equals(s2));// false
System.out.println(s1.equalsIgnoreCase(s2)); //true
```

Opérateurs d'affectation et opérateur unaire

= affectation. Ex : `i = 4;`
Affectation multiple : `i = j = 4;`

\+ nonbre positif. Ex : `int i = +2;`

\- nonbre négatif. Ex : `int i = -2;`

++ incrémenter .(ATTENTION) : après i = 5 ; j =5 ;

```
k = i++; // k devient 5 et i devient 6
m = ++j ; // m devient 6 et j devient 6
```

-- décrémenter

! négation

Opérateurs de comparaison

<	
>	
<=	
>=	supérieur ou égal
==	égalité (pas pour les chaines de caractères)
! =	différence

Opérateurs mathématiques

$+$

$-$

$*$

$/$ division $(5/2.0 = 2.5) \rightarrow (5/2 = 2)$

$\%$ modulo (reste de la division entière, $5\%2 = 1$)

$+=$ incrémentation. $x+=1$ revient à $x = x + 1$

Opérateurs logiques

|| ou (vrai si au moins une opérande est vraie. Sinon, faux)

&& et (vrai si toutes les opérandes sont vraies. Sinon, faux)

! négation (vrai si l'opérande est fausse. Sinon, faux)

$\wedge$ opérateur logique XOR

Si a et b sont des expressions booléennes, alors, on peut écrite la table de vérité suivante

a	**b**	**a || b**	**a && b**	**!(a)**
True	True	True	True	False
True	False	True	False	False
False	True	True	False	True
False	False	False	False	True

Conversion de type et d'évaluation d'une expression

- Mettre des parenthèses pour regroupes les expressions à évaluer en priorité :
  i = (12-3) * 4 ;
- String → int : `nb= Integer.ParseInt(s);`
- String → double : `nb= Double.ParseDouble(s);`
- nombre → String : `s =""+nb;`
- int → double : il suffit de faire l'affectation.

```
int  nombre = 10; // 10
double montant = nombre; //10.0
```

http://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html