***Département d'Informatique***
*DUT Informatique S1*
*Année 2014/2015*
*Algorithmique et programmation*

# Algorithmique et programmation

# TP 2 : Le langage Java, type des expressions de calcul, constantes et bibliothèques mathématiques

**NB :** *Finir le TP précédent si ce dernier n'a pas été terminé.*

## 1 Type des expressions de calcul

### 1.1 Priorité des opérateurs

La priorité des opérateurs dans les expressions Java est donnée dans le tableau suivant.

| Priorité | Type | Opérateurs |
|---|---|---|
| 1 | postfixé | *expr*`++` *expr*`--` |
| 2 | unaire | `++`*expr* `--`*expr* `+`*expr* `-`*expr* `~` `!` |
| 3 | multiplication | `*` `/` `%` |
| 4 | addition | `+` `-` |
| 5 | décalage | `<<` `>>` `>>>` |
| 6 | relationnel | `<` `>` `<=` `>=` `instanceof` |
| 7 | égalité | `==` `!=` |
| 8 | ET bit-à-bit | `&` |
| 9 | OU exclusif bit-à-bit | `^` |
| 10 | OU inclusif bit-à-bit | `\|` |
| 11 | ET logique | `&&` |
| 12 | OU logique | `\|\|` |
| 13 | ternaire | `?` `:` |
| 14 | affectation | `=` `+=` `-=` `*=` `/=` `%=` `^=` `\|=` `<<=` `>>=` `>>>=` |

Dans chaque classe de priorité les opérateurs ont la même priorité. Si plusieurs opérateurs binaires de la même classe se suivent, l'évaluation se fait en passant de la gauche vers la droite dans l'expression, sauf pour les opérateurs d'affectation où l'évaluation se fait de droite à gauche.

### 1.2 Type des expressions de calcul

Lors de l'évaluation des expressions, le type du résultat est déterminé à partir du type des opérandes. En général [1], une opération entre deux opérandes de même type (entier ou flottant) donnera un résultat du même type que les opérandes. Si les opérandes sont de types différents, le résultat sera du type « le plus général ». Par exemple, une opération entre un entier et un flottant donnera normalement un résultat flottant.

Il est important de connaître l'existence de ces règles, comme on a pu s'en rendre compte lors de l'exercice sur la conversion de degrés Fahrenheit en degrés Celsius du TP 1.

1. En réalité les règles sont un peu plus complexes. Se référer à une documentation de référence sur le langage pour les détails.

**Exercice** Reprendre l'exercice de la section 4 du TP 1 et essayer maintenant d'expliquer la différence de résultat entre :

```
double fahr;
double celsius;
...
celsius = 5/9 * (fahr - 32);
```

et

```
double fahr;
double celsius;
...
celsius = 5 * (fahr - 32) / 9;
```

**Explications** Dans les deux variantes, les variables sont des réels et les nombres des entiers. Seule la deuxième variante donne le bon résultat, la première donnant toujours 0. Cela peut s'expliquer de la manière suivante :

`celsius = 5/9 * (fahr - 32)`

- La première opération effectuée est `(fahr - 32)`, le résultat est donc un réel.
- Ensuite c'est la division qui est réalisée, ainsi on obtient la division *entière* de 5 par 9, ce qui donne zéro.
- Finalement, zéro est multiplié avec le résultat de la première opération. Comme il y a multiplication d'un entier et d'un réel on obtient un réel : 0.

`celsius = 5 * (fahr - 32) / 9`

- La première opération effectuée est `(fahr - 32)`, le résultat est donc un réel.
- Ensuite c'est la multiplication qui est réalisée, ainsi on obtient un réel correspondant à la multiplication du résultat de l'opération précédente par 5.
- Finalement, le résultat de la multiplication est divisé par l'entier 9. Comme la première opérande de la division est de type réel, c'est une division réelle qui est mise en œuvre.

En conclusion, il faut donc s'assurer du bon type des calculs. Ceci peut être fait de deux manières :

**implicitement** en s'assurant que les constantes sont du bon type ; ou

**explicitement** en utilisant une opération de *transtypage* (ou *cast*).

Par exemple, pour obtenir le bon résultat avec la première expression de calcul on écrirait :

- en utilisant un typage implicite :
  `celsius = 5.0/9 * (fahr - 32)` ou `celsius = 5/9.0 * (fahr - 32)`
- en utilisant l'opération de transtypage :
  `celsius = (double )5/9 * (fahr - 32)`

*On préférera cependant utiliser un typage implicite de la manière suivante :*

`celsius = 5.0/9.0 * (fahr - 32.0).`

**Remarques sur le transtypage** Le transtypage est une *opération déconseillée*. On tâchera de limiter son usage aux situations où il n'est pas possible de faire autrement. Les principaux inconvénients de cette opération sont que :

- la lecture des expressions est rendue plus difficile ;
- la maintenance des programmes peut être plus complexe (si on décide de changer de type par exemple) ;
- le programmeur se prive de l'aide que pourrait lui fournir le compilateur lors de certaines erreurs de programmation.

# 2 Constantes

Pour définir une constante en Java, il faut la déclarer avec l'attribut **final**. Par exemple :

```
final double G = 9.81;
final int ANNEE = 1984;
```

# 3 Fonctions mathématiques

La classe Math définit un certain nombre de fonctions mathématiques, opérant en général sur des nombres flottants. On peut citer par exemple :

- les fonctions trigonométriques : Math.sin, Math.cos, Math.tan, Math.asin, Math.acos, Math.atan, etc. ;
- les fonctions exponentielle et logarithmiques : Math.exp, Math.log, Math.log10, etc. ;
- les fonctions de puissance : Math.pow et Math.sqrt ;
- les fonctions d'arrondi : Math.abs, Math.ceil, Math.floor, etc.

Cf. `http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html` pour une documentation plus complète.

# 4 Exercices

**Calcul du nombre de radiateurs** En se basant sur l'exercice similaire effectué en TD, écrire un programme permettant de calculer le nombre de radiateurs dont on a besoin pour chauffer une pièce. On sait qu'un radiateur est capable de chauffer 8 $m^3$. L'utilisateur donnera la longueur, la largeur et la hauteur de la pièce en mètres.

**Le nombre de paquets de céréales** On sait que une tonne vaut 35 273,96 onces. Écrire un programme qui lit le poids en onces d'un paquet de céréales pour petit déjeuner, et écrit le poids du paquet en système métrique, ainsi que le nombre de paquets de céréales que l'on peut faire avec une tonne de céréales.

## Solutions possibles

```
import java. util .*;

class Radiateurs {

    static final Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        final double capacite = 8;

        int nombre;
        double longueur;
        double largeur;
        double hauteur;
        double volume;

        System.out. println ("Entrer longueur, largeur et hauteur de la pièce");
        longueur = input.nextDouble();
        largeur = input.nextDouble();
        hauteur = input.nextDouble();

        volume = longueur * largeur * hauteur;
        nombre = (int )Math.ceil(volume / capacite);

        System.out. println ("Nombre = " + nombre);
    }

}
```

```
import java. util .*;

class Cereales {

    static final Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        final double convTonneOnces = 35273.96;
        double poidsPaquetOnces;
        double poidsPaquetKilos;
        int kilos ;
        int grammes;
        int nbPaquetsTonne;

        System.out. println ("Entrer le poids d'un paquet en onces :");
        poidsPaquetOnces = input.nextDouble();

        // Calcul du poids d'un paquet en kilos
        poidsPaquetKilos = (poidsPaquetOnces * 1000.0) / convTonneOnces;
        kilos = (int )Math.floor(poidsPaquetKilos);
        grammes = (int )Math.round(1000 * (poidsPaquetKilos -
                                    Math.floor(poidsPaquetKilos )));
        System.out. println ("Un paquet pèse " +
                        kilos + " kilos " + grammes + " grammes");

        // Calcul du nombre de paquets obtenus a partir d'une tonne
        nbPaquetsTonne = (int )Math.floor(convTonneOnces / poidsPaquetOnces);
        System.out. println ("1 tonne <=> " + nbPaquetsTonne + " paquets");
    }

}
```