

**DEV1 – Laboratoires Java I****TD 2 – Alternatives**

Dans ce TD vous trouverez une introduction à la manipulation des nombres réels, aux expressions booléennes et à l'instruction `if/else`.

**Table des matières**

<b>1</b>	<b>Les réels</b>	<b>2</b>
<b>2</b>	<b>Les conditions</b>	<b>3</b>
<b>3</b>	<b>Les alternatives : <code>if/else</code></b>	<b>5</b>

# 1 Les réels

Au TD1 nous avons travaillé avec des entiers, nous allons maintenant manipuler des nombres réels.

En java les *réels* se représentent avec un point. Par exemple 12,3 s'écrira 12.3.

Les variables pour représenter un entier sont de type `int`. Pour les réels les variables sont de type `double` :

```
double taille = 1.92;
```

Pour lire un entier on utilise la méthode `nextInt()` du scanner. Pour lire un réel on utilise la méthode `nextDouble()`.

```
double y = clavier.nextDouble();
```

Les *opérateurs* permettant de manipuler les réels sont :

- ▷ l'addition : +
- ▷ la soustraction : -
- ▷ la multiplication : \*
- ▷ la division : /

```
1 package esi.dev1.td2;
2
3 import java.util.Scanner;
4
5 public class Calculs {
6
7     public static void main(String[] args) {
8         Scanner clavier = new Scanner(System.in);
9
10        int x = clavier.nextInt();
11        double y = clavier.nextDouble();
12
13        System.out.println(12.3 + 13.5);
14        System.out.println(x * 3.1415);
15        System.out.println(x + y);
16        System.out.println(x - y);
17        System.out.println(x * y);
18        System.out.println(x / y);
19        System.out.println(x*x + y*y);
20    }
21 }
```

code/Calculs.java

- ▷ À la ligne 3, on importe le Scanner afin de pouvoir lire les entrées de l'utilisateur.
- ▷ À la ligne 8, on initialise le Scanner, il scanne *l'entrée standard* : `System.in`.
- ▷ À la ligne 10, on déclare une variable nommée `x`, de type `int` (un nombre entier). On lui assigne la valeur entrée par l'utilisateur : `clavier.nextInt()`.
- ▷ À la ligne 11, on déclare une variable nommée `y`, de type `double` (un nombre réel). On lui assigne la valeur entrée par l'utilisateur : `clavier.nextDouble()`.
- ▷ Aux lignes 13 à 19 le programme affiche le résultat des différents calculs.

Le tableau suivant reprend les différences entre entiers et réels :

	type	opérateurs	lecture
Entier	<code>int</code>	<code>+, -, *, /, %</code>	<code>nextInt()</code>
Réel	<code>double</code>	<code>+, -, *, /</code>	<code>nextDouble()</code>

### Exercice 1

## Calculs avec des réels et entiers

Créez un package `g12345.dev1.td2` où vous remplacez `g12345` par votre identifiant. Dans ce package créez une classe `Exercice1`. Dans cette classe créez un programme qui affiche la valeur des expressions suivantes :

- ▷ `12.3+13.5`
- ▷ `12.3-13.5`
- ▷ `12.3*13.5`
- ▷ `2.0/3.0`
- ▷ `2/3.0`
- ▷ `2.0/3`
- ▷ `2/3`
- ▷ `2.0/0`
- ▷ `2/0`

### Exercice 2

## Périmètre et aire d'un cercle

Dans une classe `Cercle` écrivez un programme qui demande à l'utilisateur le rayon d'un cercle (un réel) et affiche son périmètre et son aire.

Rappel : Le périmètre se calcule par la formule  $2\pi r$  et l'aire par la formule  $2\pi r^2$  où  $\pi$  vaut environ 3.141593.

## 2 Les conditions

En informatique, on appelle *booléens* les deux valeurs 'vrai' et 'faux'. En java les booléens se représentent par `true` (vrai) et `false` (faux).

Une condition est une expression dont la valeur s'évalue à `true` ou `false`. Voici quelques exemples de conditions :

- ▷ `age < 18` : `age` a une valeur inférieure à 18 ;
- ▷ `nb >= 0` : `nb` est supérieur à 0 ;
- ▷ `b*b - 4*a*c < 0` : le delta  $b^2 - 4ac$  est négatif ;
- ▷ `nb >= 0 && nb <= 100` : le nombre `nb` est compris entre 0 et 100 ;
- ▷ `a < b` : la valeur de la variable `a` est inférieure à celle de `b`.

Pour construire une condition on utilisera les opérateurs de comparaison sur les entiers et sur les réels :

- ▷ plus petit : `<`
- ▷ plus petit ou égal : `<=`
- ▷ plus grand : `>`
- ▷ plus grand ou égal : `>=`
- ▷ égal : `==`

- ▷ différent : `!=`

Et on combinera des conditions avec les *opérateurs booléens* :

- ▷ Le *ET* logique s'écrit `&&`.

Exemple : `nb >= 0 && nb <= 100` vérifie si `nb` est compris entre 0 et 100. Cette condition sera vraie si `nb >= 0` ET si `nb <= 0`.

- ▷ Le *OU* logique s'écrit `||`.

Exemple : `a < b || a < c` sera vraie si `a < b` ou bien si `a < c` (ou les deux).

- ▷ La négation s'écrit `!`.

Exemple : `! (a < b)` sera vraie si `a < b` est faux, c'est-à-dire si `a >= b`.

### Exercice 3

## Conditions

Dans une classe `Conditions` écrivez un programme qui affiche la valeur des expressions suivantes :

- ▷ `10 < 20` (écrivez simplement : `System.out.println(10 < 20);` )
- ▷ `10 > 20`
- ▷ `10 == 10.0`
- ▷ `20.0/2 != 10.0`
- ▷ `true` (écrivez simplement : `System.out.println(true);` )
- ▷ `false`
- ▷ `!true`
- ▷ `!false`
- ▷ `true && true`
- ▷ `true && false`
- ▷ `false && true`
- ▷ `false && false`
- ▷ `true || true`
- ▷ `true || false`
- ▷ `false || true`
- ▷ `false || false`

### Exercice 4

## Conditions

Dans une classe `Conditions2` écrivez un programme qui demande à l'utilisateur 3 nombres entiers `a`, `b` et `c` et affiche la valeurs des expressions suivantes :

- ▷ `a%2 == 0` (`a` est divisible par 2 et donc pair)
- ▷ `a%2 == 1` (`a` est impair)
- ▷ `a%b == 0` (`a` est divisible par `b`)
- ▷ `a < b`
- ▷ `a <= b && a <= c` (`a` est le minimum)
- ▷ `(a < b && b < c) || (a > b && b > c)` (`b` est strictement compris entre `a` et `c`)

### 3 Les alternatives : if/else

L'instruction `if` permet d'exécuter des instructions si une certaine condition est vérifiée.

Le programme suivant affichera "ce nombre est positif" si `nb` est plus grand ou égal à 0 et n'affichera rien dans le cas contraire.

```
1 package esi.dev1.td2;
2
3 import java.util.Scanner;
4
5 public class Positif {
6
7     public static void main(String[] args) {
8         Scanner clavier = new Scanner();
9
10        System.out.print("Entrez un nombre entier: ");
11        int nb = clavier.nextInt();
12
13        if(nb>=0) {
14            System.out.print("ce nombre est positif.");
15        }
16    }
17 }
```

[code/Positif.java](#)

L'instruction `if/else` permet d'exécuter des instructions si une certaine condition est vérifiée et d'autres instructions si la condition n'est pas vérifiée. On peut traduire 'else' par 'sinon'.

On peut remplacer le `if` du programme précédent par le `if/else` suivant :

```
if(nb >= 0) {
    System.out.println("ce nombre est positif.");
} else {
    System.out.println("ce nombre est négatif.");
}
```

Le programme affichera "ce nombre est positif" si `nb` est plus grand ou égal à 0 et "ce nombre est négatif" sinon.

Il est aussi possible d'utiliser une succession de `if/else` :

```
if(nb > 0) {
    System.out.println("ce nombre est positif.");
} else if(nb < 0) {
    System.out.println("ce nombre est négatif.");
} else {
    System.out.println("ce nombre est nul.");
}
```

#### Exercice 5

#### Majeur - if

Dans une classe `Majeur` écrivez un programme qui demande à l'utilisateur son age et affiche s'il est majeur (s'il a plus de 18 ans). S'il n'est pas majeur le programme n'affiche rien.

Exemple : si l'utilisateur entre 19 le programme affiche "vous êtes majeur".

#### Exercice 6

#### Pair ou impair - if/else

Dans une classe `Parité` écrivez un programme qui demande à l'utilisateur un nombre entier et affiche "ce nombre est pair" ou "ce nombre est impair" selon le cas.

Exemple : si l'utilisateur entre -23 le programme affiche "ce nombre est impair".

Astuce : un nombre est pair si le reste de la division par 2 vaut 0.

#### Exercice 7

### Maximum de 2 nombres

Dans une classe `Maximum` écrivez un programme qui demande à l'utilisateur deux nombres réels et affiche le plus grand des deux.

Exemple : si l'utilisateur entre 7,5 et 2,3 le programme affiche 7,5.

#### Exercice 8

### Maximum de 3 nombres

Dans une classe `Maximum3` écrivez un programme qui demande à l'utilisateur trois nombres réels et affiche le maximum des trois.

Exemple : si l'utilisateur entre 7,5, 17,9 et 2,3 le programme affiche 17,9.

#### Exercice 9

### Le type de triangle

Dans une classe `TypeTriangle` écrivez un programme qui demande à l'utilisateur la longueur des 3 côtés d'un triangle et affiche s'il est : équilatéral (tous égaux), isocèle (2 égaux) ou quelconque.

Exemple : si l'utilisateur entre 2,5, 5 et 5 le programme affiche "le triangle est isocèle".