# TP 4 : instructions conditionnelles et fonctions

Informatique Fondamentale (IF1)

Semaine du 22 octobre 2007

Comme dans les TP précédents, vous devez *tester* vos programmes avec différentes entrées qui recouvrent tous les comportements possibles de vos programmes.

## Les fonctions en Java

Jusqu'ici, tous les programmes Java que nous avons écrits avaient la forme suivante :

```
import fr.jussieu.script.Deug;

public class MaClasse {
    public static void main(String[] args) {
        // faire differents calculs
        Deug.println(la reponse calculee);
    }
}
```

Après quelques calculs, ils *affichent* des messages à l'écran (le type du résultat est `void`).

Dans ce TP, nous allons au contraire écrire des *fonctions* qui *calculent* des résultats d'un certain *type*, à partir d'*arguments*. Le mot clé qui permet de retourner un résultat en Java est `return`.

Dans l'exemple ci-dessous, la fonction `somme` de signature

```
public static int somme(int x, int y)
```

prend en argument deux entiers `x` et `y` (de type `int`), et calcule leur somme : le résultat retourné par cette fonction est donc de type `int`. Les fonctions ainsi définies peuvent ensuite être utilisées dans le programme principal de la classe (`main`) pour afficher des résultats.

**Exemple :**

```
import fr.jussieu.script.Deug;

public class Operation{
    public static void main(String[] args){
        int a, b, c;
        Deug.println("Entrer trois entiers");
        a=Deug.readInt();
        b=Deug.readInt();
        c=Deug.readInt();
        Deug.println("En multipliant par le premier la somme des
                      deux autres on obtient " + produit(a,somme(b,c)));
    }

    public static int somme(int x, int y) {
        return (x+y);
    }
    public static int produit(int x, int y) {
        return (x*y);
    }
}
```

Après la compilation, c'est toujours le programme principal de la classe (`main`) qui est executé. Les autres fonctions définies dans la classe sont destinées à être utilisées par ce programme principal (ou par d'autres fonctions).

# 1 Opérateurs booléens

**1.** Sans utiliser les opérateurs logiques `||`, `&&`, `!` etc., écrivez les quatre fonctions suivantes dans une même classe `OperateursBooleens` :

- `not` de signature `public static boolean not(boolean b)`, qui prend un booléen en argument, et qui calcule sa négation ;
- `and` de signature `public static boolean and(boolean b, boolean c)`, qui prend deux booléens en argument, et qui calcule leur conjonction ;
- `or` de signature `public static boolean or(boolean b, boolean c)`, qui prend deux booléens en argument, et qui calcule leur disjonction ; et
- `xor` de signature `public static boolean xor(boolean b, boolean c)`, qui prend deux booléens en argument , et qui calcule leur disjonction exclusive.

Testez vos quatre fonctions (sur toutes les valeurs possibles de leurs arguments) dans le programme principal de la classe, en affichant leurs résultats.

## 2 Tri

**2.** Dans une classe `Trie`, écrivez trois fonctions (`premier`, `deuxieme` et `troisieme`), qui prennent chacune en argument trois entiers, et qui calculent respectivement le plus grand, le deuxième et le plus petit des trois entiers.

Par exemple, `premier` a la signature

```
public static int premier(int a, int b, int c).
```

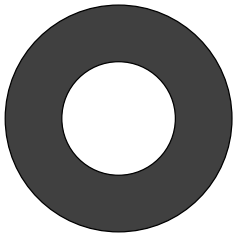
Dans le programme principal de la classe, écrivez un programme qui lit trois entiers sur l'entrée standard et les affiche dans l'ordre croissant. Pour cela, utilisez les fonctions `premier`, `deuxieme` et `troisieme`.

## 3 Couronne

**3.** Écrivez une fonction `couronne` de signature

```
public static boolean couronne(double r1, double r2, double x, double y)
```

qui prend en argument quatre flottants (double précision) $r_1$, $r_2$, $x$ et $y$, et qui retourne un résultat de type `boolean` indiquant si le point du plan de coordonnées $(x, y)$ se trouve sur la couronne de centre $(0, 0)$, de rayon intérieur $r_1$, et de rayon extérieur $r_2$, c'est-à-dire si le point $(x, y)$ se trouve dans la zone grisée de la figure suivante :



Écrivez une fonction `main` qui permet de tester votre fonction avec des valeurs entrées par l'utilisateur.

**4.** Dans la même classe, écrivez aussi une fonction `aire` de signature

```
public static double aire(double r1, double r2)
```

qui calcule l'aire de la couronne de centre $(0, 0)$, de rayon intérieur $r_1$, et de rayon extérieur $r_2$.

Modifiez votre fonction `main` pour qu'elle permette de tester les deux fonctions.

## 4 S'il vous reste du temps : dates et jour de la semaine

**5.** Écrivez une fonction de signature

```
public static int jour (int j, int m, int a, int p)
```

qui prend en entrée quatre entiers tels que :

- les trois premiers définissent une date (jour `j`, mois `m`, année `a`)
- le dernier (`p`) indique le jour de la semaine qui correspond au 1er janvier de l'année `a` (0 pour dimanche, 1 pour lundi, etc.).

et qui retourne l'entier qui correspond au jour de la semaine à la date `j/m/a`.

En utilisant cette fonction, écrivez une fonction `main` dans la même classe qui demande à l'utilisateur d'entrer trois entiers correspondant à une date, et un jour de la semaine (celui du 1er janvier de l'année en question), et qui affiche le jour de la semaine de la date proposée. Testez votre programme.

*Le petit plus :* Vous pouvez aussi intégrer dans cette classe une fonction `date` de signature

```
public static boolean date (int j, int m, int a)
```

inspirée de la question 8 du tp2, qui vérifie que les trois entiers entrés par l'utilisateur définissent bien une date.

**6.** Modifiez la fonction `jour` de la question précédente pour qu'elle ne prenne plus en argument que les trois entiers définissant une date, sachant que le 1er janvier 2000 était un samedi. Cette nouvelle fonction a la signature

```
public static int jour2 (int j, int m, int a)
```

Modifiez la fonction `main` en conséquence et testez votre programme.