

Cours Introduction à la Programmation Java III (IP1 Java)

Arnaud Sangnier
sangnier@irif.fr

Mercredi 11 Octobre 2017
INFO et MATHS-INFO

Dans les épisodes précédents

- Présentation de certains aspects de **Java**
- Ce que nous avons vu :
 - Les données et leur type : **int** et **String**
 - Les variables : affectation, lecture, modification
 - Les fonctions : **public static int f(int z) {...}**
 - Les tests : **if (){ ...} else {...}**
 - Les boucles : **for (int i=0;i<10;i=i+1) {...}**

Construction de programmes - I

- Comment construire vos programmes
 - Rappel : un programme est une suite d'instructions
 - Quelles instructions avons nous vu :
 - Affectation de variables $x = 5$ (la première fois)
 - Modification de variables $x = x + 1$
 - Appel de fonctions $y = f(x)$ ou $g(x)$ (si g ne renvoie pas de valeur)
 - Tests :

```
if (x > 5) {  
    [suite d'instructions]  
} else {  
    [suite d'instructions]  
}
```

- Boucles :

```
for(int i=1;i<5;i=i+1){  
    [suite d'instructions]  
}
```

Exemple

- Écrire un programme qui affiche 15 lignes de la forme *****...***** avec 60 étoiles par ligne
 - Il faut faire une boucle par ligne (qui va donc faire 15 tours)
 - À la fin de cette boucle il faut aller à la ligne
 - Avant il faut afficher la ligne, c'est-à-dire les 60 étoiles
 - On fait une boucle qui fait 60 tours

```
for (int i=0;i<15;i=i+1){  
    for (int j=0;j<60;j=j+1){  
        System.out.print('*');  
    }  
    System.out.println("");  
}
```

Cette instruction appartient à la boucle interne

Cette instruction appartient à la boucle externe

Construction de programmes - II

- On peut ainsi utiliser un **if** dans un **if**, un **for** dans un **for** (boucles imbriquées), un **if** dans un **for** etc
- Il faut aussi définir les fonctions que l'on utilise dans le programme. Typiquement on les met avant dans le fichier.

```
public static int f (int x, int y) {  
    [suite d'instructions]  
}
```

- Comme un appel de fonctions est une instruction, **on peut aussi appeler une fonction dans une fonction!**

Exemple

- Écrire une fonction qui renvoie la somme des carrés de 1 à n où n est donné en paramètres
 - Combien de paramètres prend la fonction : un de type **int**
 - La fonction retourne-t-elle une valeur : oui de type **int**
 - Quelle est le nom de la fonction : **somCarres**

```
public static int somCarres (int n) {  
    int x = 0;  
    for (int i=0;i<n;i=i+1){  
        x = x + (i * i) ;  
    }  
    return x ;  
}
```

Exemple

- Écrire une fonction qui pour un n donné calcule le produit $S_1 * S_2 * \dots * S_N$ où S_i est la somme des carrés des entiers de 1 à i
 - Combien de paramètres prend la fonction : un de type **int**
 - La fonction retourne-t-elle une valeur : oui de type **int**
 - Quelle est le nom de la fonction : **prodSomCarres**
 - On va utiliser la fonction précédente

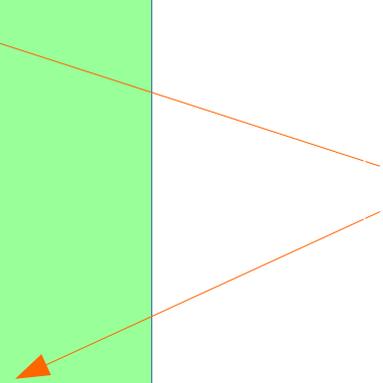
Exemple

```
public static int somCarres (int n) {  
    int x = 0;  
    for (int i=0;i<n;i=i+1){  
        x = x + (i * i) ;  
    }  
    return x ;  
}
```

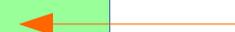
```
public static int prodSomCarres (int n) {  
    int r = 1;  
    for (int i=0;i<n;i=i+1){  
        x = r * somCarres(i);  
    }  
    return r ;  
}
```

```
.... main(...){  
    int z = prodSomCarres (3) ;  
    System.out.println(z) ;  
}
```

**Ces deux n
sont différents**



Bien tester sa fonction



Exemple

```
public static int somCarres (int n) {
    int x = 0;
    for (int i=0;i<n;i=i+1){
        x = x + (i * i) ;
    }
    return x ;
}

public static int prodSomCarres (int m) {
    int r = 1;
    for (int i=0;i<m;i=i+1){
        x = r + somCarres(i);
    }
    return r ;
}

.... main(...){
    int z = prodSomCarres (3) ;
    System.out.println(z) ;
}
```

Exemple

- Écrire une fonction qui prend deux paramètres n et l et qui affiche n lignes de l étoiles
 - Combien de paramètres prend la fonction : deux de type **int**
 - La fonction retourne-t-elle une valeur : non
 - Quelle est le nom de la fonction : **ligneEtoiles**
 - On a ce code qui affiche 15 lignes de 60 étoiles

```
for (int i=0;i<15;i=i+1){  
    for (int j=0;j<60;j=j+1){  
        System.out.print("*");  
    }  
    System.out.println("");  
}
```

Exemple

```
public static void ligneEtoiles(int n,int l){  
    for (int i=0;i<n;i=i+1){  
        for (int j=0;j<l;j=j+1){  
            System.out.print('*');  
        }  
        System.out.println("");  
    }  
}
```

Appel de fonctions dans les fonctions

- Lorsque l'on appelle une fonction dans une autre fonction, il faut faire attention à ne pas créer de comportements infinis
- Par exemple :

```
public static int f (int x) {  
    int y = 2 * x ;  
    int z = g(y) ;  
    return z ;  
}  
  
public static int g (int u) {  
    int t = u * u ;  
    int w = f(t) ;  
    return w ;  
}  
  
... main(...){  
    int a = f(5) ;  
    System.out.println(a) ;  
}
```

Qu'affiche ce programme ?
RIEN
Il ne termine jamais et finit par
provoquer une erreur

Un nouveau type

- Le type **int** représente les valeurs entières : -2,1,-100,5,0,...
- Le type **String** représente les chaînes de caractères : "aaa", "Hello", "Bob",...
- Le type **boolean** représente les booléens.
- Les booléens n'ont que deux valeurs possibles : **true** et **false**
- Les opérateurs booléens
 - opérateur unaire : **!** (non)
 - opérateurs binaires : **||** (ou) et **&&** (non)
 - pour savoir comment interpréter ces opérateurs il faut connaître leur table de vérité

Les tables de vérité - I

- `a && b`

a	b	a && b
True	True	True
False	True	False
True	False	False
False	False	False

- `a || b`

a	b	a b
True	True	True
False	True	True
True	False	True
False	False	False

Les tables de vérité - II

- Les tables de vérité peuvent exister pour des formules plus complexes
- Par exemple : $a \ \&\& \ (b \ || \ (! \ c))$

a	b	c	! c	B (!c)	a && (b (! c))
True	True	True	False	True	True
False	True	True	False	True	False
True	False	True	False	False	False
False	False	True	False	False	False
True	True	False	True	True	True
False	True	False	True	True	True
True	False	False	True	True	True
False	False	False	True	True	False

Où trouve-t-on les booléens ?

- En fait, vous les utilisez déjà
- Quand on fait un test, par exemple $3 < 2$, il correspond à une valeur booléenne, ici `false`
- Ainsi si le type de $(3 < 2)$ on obtient le type `boolean`
- Donc dans un `if (...)` ce que l'on met entre parenthèses est une valeur de type `boolean`
- Ainsi toutes les expressions suivantes sont des booléens
 - $2 == 3$, $2 <= 3$, `"aaaa".equals("bbbb")` etc

Booléens dans des fonctions - I

- On peut ainsi faire des fonctions qui renvoient un booléen
- Par exemple, une fonction qui dit si un nombre n est premier ou non
- Pour cela on teste tous les entiers entre 2 et $n-1$, pour voir si il existe un diviseur de n

```
public static boolean estPremier (int n) {  
    boolean r=true ;  
    for(int i=2;i<n;i=i+1){  
        if (r % i == 0){  
            r= false ;  
        }  
    }  
    return r ;  
}
```

Booléens dans des fonctions - II

- On peut aussi mettre un booléen en paramètre
- Écrire une fonction prenant en paramètre un booléen b et un entier n et qui affiche n lignes avec n étoiles si b est vrai et n lignes avec n dièses si n est faux.

```
public static void sharpOrStar(boolean b,int n) {
    String s = "" ;
    String c = " " ;
    if (b) :
        c="*"
    else :
        c= "#"
    for(int i=0;i<n;i=i+1){
        s =s + c
    }
    for(int i=0;i<n;i=i+1){
        System.out.println(s) ;
    }
}
```