

# Itérations : introduction

Il y a 3 structures de contrôle:

- les branchements conditionnels,
- les **itérations**, et
- les boucles conditionnelles.

## La boucle **for**

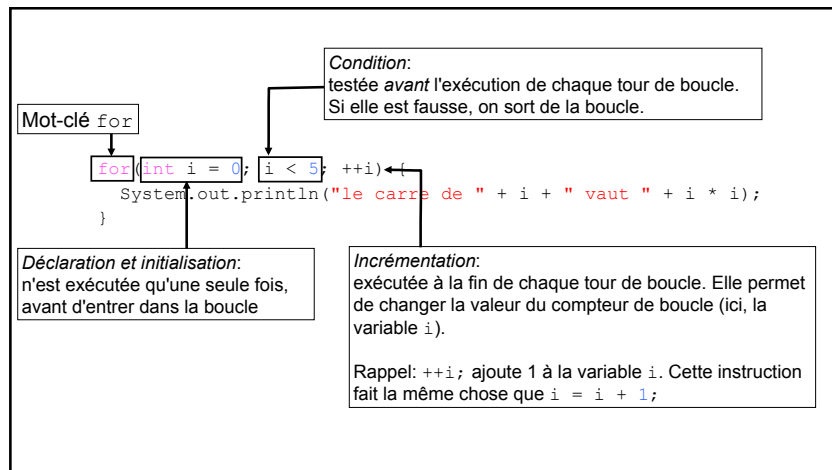
Une boucle **for** permet de répéter un nombre donné de fois la même série d'instructions.

Par exemple, si on fait:

```
for(int i = 0; i < 5; ++i) {  
    System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

le programme affichera les carrés des 5 premiers entiers:

```
le carre de 0 vaut 0  
le carre de 1 vaut 1  
le carre de 2 vaut 4  
le carre de 3 vaut 9  
le carre de 4 vaut 16
```



```
for(int i = 0; i < 5; ++i) {
    System.out.println("le carre de " + i + " vaut " + i * i);
}
```

**Corps de la boucle:**  
Bloc d'instructions qui seront exécutées à chaque tour de boucle.

Comme pour le `if`, les accolades ne sont obligatoires que si plusieurs instructions doivent être répétées.

Si il n'y a qu'une seule instruction, on peut ne pas utiliser d'accolades:

```
for(int i = 0; i < 5; ++i)
    System.out.println("i = " + i);
```

Mais, toujours comme pour le `if`, il est conseillé de garder les accolades:

```
for(int i = 0; i < 5; ++i) {
    System.out.println("i = " + i);
}
```

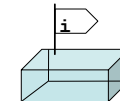
## Pas-à-pas

```
for(int i = 0; i < 5; ++i) {
    System.out.println("le carre de " + i + " vaut " + i * i);
}
```

Ce qui s'affiche dans la fenêtre Terminal:

```
|
```

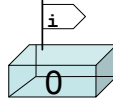
La variable `i` est déclarée et initialisée à 0



```
for(int i = 0; i < 5; ++i) {
    System.out.println("le carre de " + i + " vaut " + i * i);
}
```

Ce qui s'affiche dans la fenêtre Terminal:

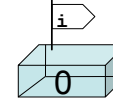
```
|
```



```
→ for(int i = 0; i < 5; ++i) {  
    System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

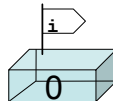
```
l
```



```
for(int i = 0; i < 5; ++i) {  
→ System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

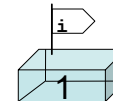
```
le carre de 0 vaut 0  
l
```



```
→ for(int i = 0; i < 5; ++i) {  
    System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

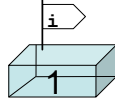
```
le carre de 0 vaut 0  
l
```



```
→ for(int i = 0; i < 5; ++i) {  
    System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

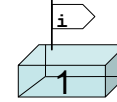
```
le carre de 0 vaut 0  
l
```



```
for(int i = 0; i < 5; ++i) {  
→ System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

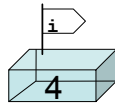
```
le carre de 0 vaut 0  
le carre de 1 vaut 1  
|
```



```
→ for(int i = 0; i < 5; ++i) {  
    System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

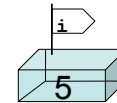
```
le carre de 0 vaut 0  
le carre de 1 vaut 1  
|
```



```
→ for(int i = 0; i < 5; ++i) {  
    System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

```
le carre de 0 vaut 0  
le carre de 1 vaut 1  
le carre de 2 vaut 4  
le carre de 3 vaut 9  
le carre de 4 vaut 16  
|
```



```
→ for(int i = 0; i < 5; ++i) {  
    System.out.println("le carre de " + i + " vaut " + i * i);  
}
```

Ce qui s'affiche dans la fenêtre Terminal:

```
le carre de 0 vaut 0  
le carre de 1 vaut 1  
le carre de 2 vaut 4  
le carre de 3 vaut 9  
le carre de 4 vaut 16  
|
```

```
for(int i = 0; i < 5; ++i) {
    System.out.println("le carre de " + i + " vaut " + i * i);
}
```

Ce qui s'affiche dans la console :

```
le carre de 0 vaut 0
le carre de 1 vaut 1
le carre de 2 vaut 4
le carre de 3 vaut 9
le carre de 4 vaut 16
|
```

Le programme continue en exécutant les instructions après la boucle.

La variable `i` n'est déclarée que pour l'intérieur de la boucle.

Elle ne peut pas être utilisée à l'extérieur de la boucle.

## Syntaxe de l'instruction `for`

```
for(déclaration_et_initialisation; condition; incrémentation) {
    bloc
}
```

- Si la condition ne devient jamais fausse, les instructions dans la boucle sont répétées indéfiniment !

## Affichage d'une table de multiplication

Dans le programme suivant, la même ligne ou presque est répétée 10 fois:  
Une constante prend les valeurs de 1 à 10.

```
System.out.println("Table de multiplication par 5:");

System.out.println("5 multiplie par 1 vaut " + 5 * 1);
System.out.println("5 multiplie par 2 vaut " + 5 * 2);
System.out.println("5 multiplie par 3 vaut " + 5 * 3);
System.out.println("5 multiplie par 4 vaut " + 5 * 4);
System.out.println("5 multiplie par 5 vaut " + 5 * 5);
...
```

→ il faut utiliser une boucle `for` pour éviter cette répétition.

## Affichage d'une table de multiplication

On peut remplacer:

```
System.out.println("5 multiplie par 1 vaut " + 5 * 1);
System.out.println("5 multiplie par 2 vaut " + 5 * 2);
System.out.println("5 multiplie par 3 vaut " + 5 * 3);
System.out.println("5 multiplie par 4 vaut " + 5 * 4);
System.out.println("5 multiplie par 5 vaut " + 5 * 5);
...
```

par

```
for(int i = 1; i <= 10; ++i) {
    System.out.println("5 multiplie par " + i + " vaut " + 5 * i);
}
```

La variable `i` prend ici les valeurs de 1 à 10.

Que s'affiche-t-il quand on exécute le code :

```
for(int i = 0; i < 5; ++i) {  
    System.out.print(i);  
    if (i % 2 == 0) {  
        System.out.print("p");  
    }  
    System.out.print(" ");  
}  
System.out.println();
```

A: 0p 1 2p 3 4p

B: 0p 1 2 3 4

C: 0 1 2p 3 4

D: 0p 1p 2p 3p 4p

?

## Itérations : approfondissements et exemples

### Exemples d'autres formes de boucles `for`

```
for (int p = 0; p < 10; p += 2) {  
    ...  
} la variable p prendra les valeurs de 0, 2, 4, 6, 8 (p += 2 est équivalent à p = p + 2);
```

```
for (int k = 10; k > 0; --k) {  
    ...  
} la variable k prendra les valeurs 10, 9, 8 ... jusqu'à 1;
```

```
for (int i = 0; i >= 0; ++i) {  
    ...  
} la condition est toujours vraie (du moins dans le principe).  
La boucle est répétée indéfiniment et la variable i prendra toutes les valeurs positives que le type int peut représenter
```

### Boucles infinies

Une boucle `for` peut ne pas s'arrêter, ce qui se produit quand la condition est toujours vraie. Plusieurs causes sont possibles:

1. On s'est trompé sur la condition:

Par exemple:

```
for (int i = 0; i > -1; ++i) { // !!!
```

### Boucles infinies

Une boucle `for` peut ne pas s'arrêter, ce qui se produit quand la condition est toujours vraie. Plusieurs causes sont possibles:

1. On s'est trompé sur la condition:

Par exemple:

```
for (int i = 0; i > -1; ++i) { // !!!
```

2. On s'est trompé sur l'incrémentation:

```
for (int i = 0; i < 10; ++j) { // !!!
```

j est incrémenté au lieu de i, i garde donc toujours la valeur 0, et la boucle ne s'arrête pas.

## Pas de point-virgule (;) à la fin de l'instruction **for**

Les instructions suivantes n'affichent qu'une seule fois la chaîne "bonjour":

```
for(int i = 0; i < 10; ++i);  
    System.out.println("bonjour");
```

Le point-virgule seul est considéré comme une instruction (qui ne fait rien).

Le corps de la boucle est donc constitué de cette instruction qui ne fait rien:

```
for(int i = 0; i < 10; ++i)  
{  
    ;  
    System.out.println("bonjour");  
}
```

i prendra les valeurs de 0 à 10, puis l'ordinateur sortira de la boucle, et exécutera l'instruction `System.out.println("bonjour");` une seule fois.

## Attention aux accolades

```
for(int i = 0; i < 5; ++i)  
{  
    System.out.println("i = " + i);  
    System.out.println("Bonjour");  
}
```

affiche:

```
i = 0  
i = 1  
i = 2  
i = 3  
i = 4  
Bonjour
```

Interprétation:

```
for(int i = 0; i < 5; ++i)  
{  
    System.out.println("i = " + i);  
    System.out.println("Bonjour");  
}
```

## Evitez de modifier une variable compteur à l'intérieur d'une boucle **for**

```
for(int i = 0; i < 10; ++i) {  
    ...  
    if (...)  
        --i; // !!!  
}
```

1. Ça ne fera sans doute pas ce que vous voulez: n'oubliez pas que la boucle `for`, de son côté, incrémente la variable `i`.
2. Un relecteur risque de ne pas s'apercevoir que la variable est modifiée également à l'intérieur de la boucle, et de ne pas comprendre le fonctionnement.

## Moyenne de 4 notes

Sans boucle `for`, en utilisant 5 variables:

```
Scanner clavier = new Scanner(System.in);  
  
double somme = 0;  
  
System.out.println("Entrez la note numero 1");  
double note1 = clavier.nextDouble();  
  
System.out.println("Entrez la note numero 2");  
double note2 = clavier.nextDouble();  
  
System.out.println("Entrez la note numero 3");  
double note3 = clavier.nextDouble();  
  
System.out.println("Entrez la note numero 4");  
double note4 = clavier.nextDouble();  
  
somme = note1 + note2 + note3 + note4;  
  
System.out.println("Moyenne = " + somme / 4);
```



Sans boucle `for`, en n'utilisant que 2 variables:

```
double somme = 0;

System.out.println("Entrez la note numero 1");
double note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 2");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 3");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 4");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Moyenne = " + somme / 4);
```

Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double somme = 0;

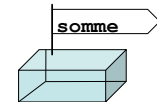
System.out.println("Entrez la note numero 1");
double note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 2");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 3");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 4");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Moyenne = " + somme / 4);
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double somme = 0;

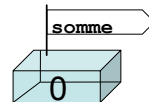
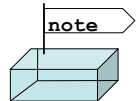
System.out.println("Entrez la note numero 1");
double note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 2");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 3");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 4");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Moyenne = " + somme / 4);
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double somme = 0;

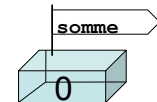
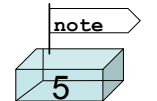
System.out.println("Entrez la note numero 1");
double note = clavier.nextDouble();
→ somme = somme + note;

System.out.println("Entrez la note numero 2");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 3");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Entrez la note numero 4");
note = clavier.nextDouble();
somme = somme + note;

System.out.println("Moyenne = " + somme / 4);
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double somme = 0;
```

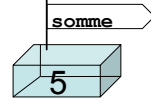
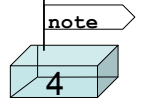
```
System.out.println("Entrez la note numero 1");  
double note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Entrez la note numero 2");  
→ note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Entrez la note numero 3");  
note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Entrez la note numero 4");  
note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Moyenne = " + somme / 4);
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double somme = 0;
```

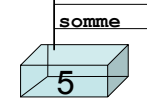
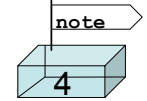
```
System.out.println("Entrez la note numero 1");  
double note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Entrez la note numero 2");  
note = clavier.nextDouble();  
→ somme = somme + note;
```

```
System.out.println("Entrez la note numero 3");  
note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Entrez la note numero 4");  
note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Moyenne = " + somme / 4);
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double somme = 0;
```

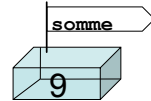
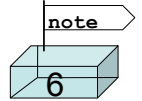
```
System.out.println("Entrez la note numero 1");  
double note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Entrez la note numero 2");  
note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Entrez la note numero 3");  
note = clavier.nextDouble();  
→ somme = somme + note;
```

```
System.out.println("Entrez la note numero 4");  
note = clavier.nextDouble();  
somme = somme + note;
```

```
System.out.println("Moyenne = " + somme / 4);
```



Même programme en utilisant une boucle for.

```
double somme = 0;
```

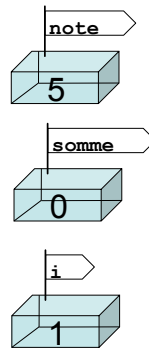
```
for(int i = 1; i <= 4; ++i) {  
    System.out.println("Entrez la note numero " + i);  
    double note = clavier.nextDouble();  
    somme = somme + note;  
}
```

```
System.out.println("Moyenne = " + somme / 4);
```

```
double somme = 0;

for(int i = 1; i <= 4; ++i) {
    System.out.println("Entrez la note numero " + i);
    → double note = clavier.nextDouble();
    somme = somme + note;
}

System.out.println("Moyenne = " + somme / 4);
```



Comment modifier le code pour laisser l'utilisateur choisir le nombre de notes ?

```
double somme = 0;

for(int i = 1; i <= 4; ++i) {
    System.out.println("Entrez la note numero " + i);
    double note = clavier.nextDouble();
    somme = somme + note;
}

System.out.println("Moyenne = " + somme / 4);
```

```
System.out.println("Entrez le nombre de notes");
int nombre_de_notes = clavier.nextInt();

double somme = 0;

for(int i = 1; i <= nombre_de_notes; ++i) {
    System.out.println("Entrez la note numero " + i);
    double note = clavier.nextDouble();
    somme = somme + note;
}

System.out.println("Moyenne = " + somme / nombre_de_notes);
```

Il y a un **bug**!

```
System.out.println("Entrez le nombre de notes");
int nombre_de_notes = clavier.nextInt();

double somme = 0;

for(int i = 1; i <= nombre_de_notes; ++i) {
    System.out.println("Entrez la note numero " + i);
    double note = clavier.nextDouble();
    somme = somme + note;
}

System.out.println("Moyenne = " + somme / nombre_de_notes);
```

Une **solution**:

```
System.out.println("Entrez le nombre de notes");
int nombre_de_notes = clavier.nextInt();

double somme = 0;

if (nombre_de_notes > 0) {
    for(int i = 1; i <= nombre_de_notes; ++i) {
        System.out.println("Entrez la note numero " + i);
        double note = clavier.nextDouble();
        somme = somme + note;
    }

    System.out.println("Moyenne = " + somme / nombre_de_notes);
}
```

## Boucles imbriquées

Reprenons l'exemple précédent de la table de multiplication par 5:

```
for(int i = 1; i <= 10; ++i) {
    System.out.println("5 multiplie par " + i + " vaut " + 5 * i);
}
```

Supposons qu'on veuille maintenant afficher toutes les tables de multiplication, de 2 à 10.

Il suffit de mettre la boucle précédente dans une autre boucle, et de remplacer le 5 par...ce qu'il faut.

## Boucles imbriquées

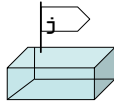
```
for(int j = 2; j <= 10; ++j) {
    for(int i = 1; i <= 10; ++i) {
        System.out.println("5 multiplie par " + i + " vaut " + 5 * i);
    }
}
```

affiche 9 fois la table de multiplication par 5

## Boucles imbriquées

```
for(int j = 2; j <= 10; ++j) {
    for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}
```

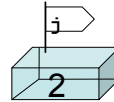
affiche la table de multiplication par 2, puis par 3, jusque 10.



```

→ for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
    for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

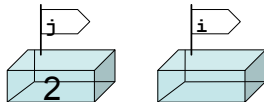


```

for(int j = 2; j <= 10; ++j) {
→ System.out.println("Table de multiplication par " + j + ": ");
    for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

Table de multiplication par 2:

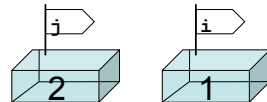


```

for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
→ for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

Table de multiplication par 2:

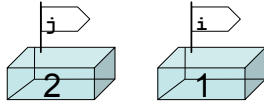


```

for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
→ for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

Table de multiplication par 2:  
 2 multiplie par 1 vaut 2



```

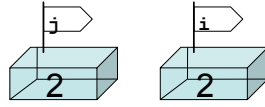
for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
    → for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
|

```



```

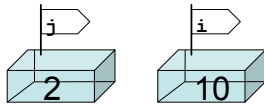
for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
    for(int i = 1; i <= 10; ++i) {
        → System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
|

```



```

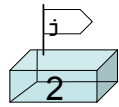
for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
    → for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
|

```



```

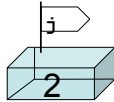
for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
    for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
    → }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
|

```



```

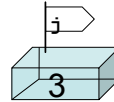
→ for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
    for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
|

```



```

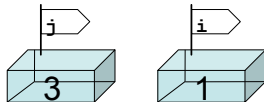
for(int j = 2; j <= 10; ++j) {
→ System.out.println("Table de multiplication par " + j + ": ");
    for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
Table de multiplication par 3:
|

```



```

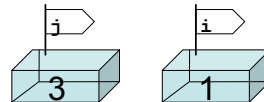
for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
→ for(int i = 1; i <= 10; ++i) {
        System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
Table de multiplication par 3:
|

```



```

for(int j = 2; j <= 10; ++j) {
    System.out.println("Table de multiplication par " + j + ": ");
    for(int i = 1; i <= 10; ++i) {
→ System.out.println(j + " multiplie par " + i + " vaut " + j * i);
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
Table de multiplication par 3:
3 multiplie par 1 vaut 3
|

```

# Itérations

Que s'affiche-t-il quand on exécute le code :

```
for(int i = 0; i < 3; ++i) {  
    for(int j = 0; j < 4; ++j) {  
        if (i == j) {  
            System.out.print("*");  
        } else {  
            System.out.print(j);  
        }  
    }  
    System.out.println("");  
}
```

A:	C:
*123	****
*123	****
*123	****

B: D:

012*	*123
012*	0*23
012*	01*3

?

Que s'affiche-t-il quand on exécute le code :

```
for(int i = 0; i < 3; ++i) {  
    for(int j = 0; j < i; ++j) {  
        System.out.print(j);  
    }  
    System.out.println("");  
}
```

A:	C:
	rien
0	
01	

B: D:

0	0123
01	0123
012	0123

?



# Les boucles conditionnelles

Il y a 3 structures de contrôle:

- les branchements conditionnels,
- les itérations, et
- les **boucles conditionnelles**.

Les itérations, ou boucles **for**, permettent de répéter une partie du programme.

Elles sont utilisées quand le nombre de répétitions est connu *avant* d'entrer dans la boucle.

Selon le problème à résoudre, il arrive qu'on ne connaisse pas combien de fois la boucle devra être exécutée.

On utilise alors une boucle conditionnelle, ou boucle **do..while** / **while**.

```
System.out.println("Entrez le nombre de notes");
int nombreDeNotes = clavier.nextInt();

double somme = 0;

if (nombreDeNotes > 0) {
    for(int i = 1; i <= nombreDeNotes; ++i) {
        System.out.println("Entrez la note numero " + i);
        double note = clavier.nextDouble();
        somme = somme + note;
    }

    System.out.println("Moyenne = " + somme / nombreDeNotes);
}
```

```

System.out.println("Entrez le nombre de notes");
int nombreDeNotes = clavier.nextInt();

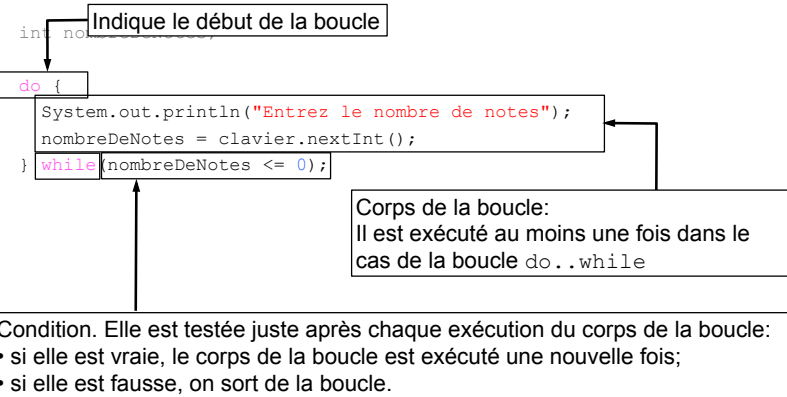
double somme = 0;

if (nombreDeNotes > 0) {
    for(int i = 1; i <= nombreDeNotes; ++i) {
        System.out.println("Entrez la note numero " + i);
        double note = clavier.nextDouble();
        somme = somme + note;
    }

    System.out.println("Moyenne = " + somme / nombreDeNotes);
}

```

Comment forcer l'utilisateur à entrer une note supérieure à 0 ?



## Syntaxe de l'instruction do...while

```

do {
    bloc
} while (condition);

```

- Comme pour l'instruction **if**:
  - La condition peut utiliser des opérateurs logiques.
  - Les parenthèses autour de la condition sont obligatoires.
- Les instructions à l'intérieur de la boucle **do...while** sont toujours exécutées au moins une fois.
- Si la condition ne devient jamais fausse, les instructions dans la boucle sont répétées indéfiniment !

## L'instruction while...

Il existe également la forme suivante:

```

while (condition) {
    bloc
}

```

Le principe est similaire à celui de la boucle **do...while** que nous venons de voir.

La différence est que la condition est testée **avant** d'entrer dans la boucle. Si la condition est fausse, les instructions dans la boucle ne sont donc pas exécutées.

## Exemple

```
int i = 100;
do {
    System.out.println("bonjour");
} while (i < 10);
// affichera une fois bonjour.
```

Dans les 2 cas,  
la condition `i < 10` est fausse.

```
int i = 100;
while (i < 10) {
    System.out.println("bonjour");
}
// n'affichera rien.
```

## Erreurs classiques

Il n'y a pas de ; à la fin de la condition du `while`...

```
while (i < 10); // !!
    ++i;
```

sera interprété comme

```
while (i < 10)
;
    ++i;
```

Le point-virgule est considéré comme le corps de la boucle,  
et l'instruction `++i` est **après la boucle**.

Si `i` est inférieur à 10, on entre dans la boucle pour ne jamais  
en ressortir puisque la valeur de `i` ne sera jamais modifiée.

En revanche, il y a un point-virgule à la fin du `do...while`:

```
do {
    ++i;
} while (i < 10);
```

## Quand utiliser la boucle `while` ? Quand utiliser la boucle `for` ?

Quand le nombre d'itérations (de répétitions) est connu avant d'entrer dans la boucle,  
utiliser `for`:

```
for(int i = 0; i < nombre_d_iterations; ++i) {
```

Sinon, utiliser `while`:

– quand les instructions doivent être effectuées au moins une fois, utiliser `do...while`:

```
do {
    instructions;
} while (condition);
```

– Sinon, utiliser la forme `while`...

```
while (condition) {
    instructions;
}
```

```
int nombreDeNotes;
```

```
do {
    System.out.println("Entrez le nombre de notes");
    nombreDeNotes = clavier.nextInt();
} while (nombreDeNotes <= 0);
```

Entrez le nombre de notes:

-2

il faut entrer un nombre supérieur à 0

Entrez le nombre de notes:

5

```
int nombreDeNotes;

do {
    System.out.println("Entrez le nombre de notes");
    nombreDeNotes = clavier.nextInt();
    if (nombreDeNotes <= 0) {
        System.out.println("il faut entrer un nombre supérieur a 0");
    }
} while(nombreDeNotes <= 0);
```

```
Entrez le nombre de notes:
-2
il faut entrer un nombre superieur a 0
Entrez le nombre de notes:
5
```

## Comment trouver la condition ?

On veut répéter la boucle **tant que** le nombre de notes est incorrect, le nombre de notes est incorrect si il est inférieur ou égal à 0, ce qui donne la condition précédente:

```
while (nombreDeNotes <= 0);
```

## Comment trouver la condition ?

Supposons maintenant qu'on veuille limiter le nombre de notes à 10.

On veut toujours qu'il soit supérieur à 0.

Comment trouver la nouvelle condition ?

On veut répéter la boucle **tant que** le nombre de notes est incorrect, le nombre de notes est incorrect si il est inférieur ou égal à 0 **ou** si il est supérieur à 10,

ce qui donne la nouvelle condition:

```
while (nombreDeNotes <= 0 || nombreDeNotes > 10);
```

Supposons qu'on veuille écrire un programme qui demande à l'utilisateur de deviner un nombre. Pour simplifier, nous supposons que le nombre à deviner est toujours 5.

Le programme peut s'écrire ainsi:

```
int nombreADeviner = 5;
int nombreEntre;

do {
    System.out.println("Entrez un nombre entre 1 et 10");
    nombreEntre = clavier.nextInt();
} while( condition ? );

System.out.println("Trouve");
```

la boucle doit être répétée  
tant que l'utilisateur n'a pas trouvé le nombre à deviner, c'est-à-dire  
tant que `nombreEntre` est différent de `nombreADeviner`,  
la condition est donc:

```
nombreEntre = clavier.nextInt();
} while(nombreEntre != nombreADeviner);

System.out.println("Trouve");
```

Supposons qu'on veuille en plus limiter le nombre d'essais à 3.  
On peut ajouter une variable qui va compter le nombre d'essais utilisés:

```
int nombreADeviner = 5;
int nombreEntre;
int nombreEssais = 0;

do {
    System.out.println("Entrez un nombre entre 1 et 10");
    nombreEntre = clavier.nextInt();
    ++nombreEssais;
} while ( condition ? );

System.out.println("Trouve");
```

Comment modifier la condition pour que  
la boucle s'arrête quand le nombre  
d'essais dépasse 3 ?

la boucle doit être répétée  
tant que l'utilisateur n'a pas trouvé le nombre à deviner **et** qu'il reste des essais,  
c'est-à-dire  
tant que `nombreEntre` est différent de `nombreADeviner` **et** que  
`nombreEssais` est inférieur à 3,  
la condition est donc:

```
} while(nombreEntre != nombreADeviner && nombreEssais < 3);
```

```
int nombreADeviner = 5;
int nombreEntre;
int nombreEssais = 0;

do {
    System.out.println("Entrez un nombre entre 1 et 10");
    nombreEntre = clavier.nextInt();
    ++nombreEssais;
} while(nombreEntre != nombreADeviner && nombreEssais < 3);
```

Si on veut afficher un message pour indiquer à l'utilisateur s'il a trouvé le nombre ou  
si il a épuisé ses essais, on peut ajouter après la boucle:

```
if (nombreEntre == nombreADeviner) {
    System.out.println("Trouve");
} else {
    System.out.println("Perdu. Le nombre etait " + nombreADeviner);
}
```

```
int nombreADeviner = 5;
int nombreEntre;
int nombreEssais = 0;

do {
    System.out.println("Entrez un nombre entre 1 et 10");
    nombreEntre = clavier.nextInt();
    ++nombreEssais;
} while(nombreEntre != nombreADeviner && nombreEssais < 3);
```

Attention, si on avait utilisé `nombreEssais < 3` comme condition:

```
if (nombreEssais < 3) {
    System.out.println("Trouve");
} else {
    System.out.println("Perdu. Le nombre etait " + nombreADeviner);
}
```

le programme afficherait "Perdu. ..." quand l'utilisateur trouve au troisième essai.

# Les blocs d'instructions

## Les blocs

En Java, les instructions peuvent être regroupées en **blocs**.

Les blocs sont identifiés par des délimiteurs de début et de fin : { et }

Exemple:

```
{
    Scanner keyb = new Scanner(System.in);
    int i;
    double x;

    System.out.println("Valeurs pour i et x : ") ;
    i = keyb.nextInt();
    x = keyb.nextDouble();
    System.out.println("Vous avez saisi : i = " + i +
                      ", x = " + x);
}
```

## Les blocs

Les blocs ont en Java une grande autonomie.

Ils peuvent contenir leurs propres déclarations et initialisation de variables:

```
if (i != 0) {
    int j = 0;

    ...
    j = 2 * i;
    ...
}
// A partir d'ici, on ne peut plus utiliser j
```

## Notion de portée

- Les variables déclarées à l'intérieur d'un bloc sont appelées **variables locales** (au bloc). Elles ne sont accessibles qu'à l'intérieur du bloc.

```
if (i != 0) {
    int j = 0;

    ...
    j = 2 * i;
    ...
}
// A partir d'ici, on ne peut plus utiliser j
```

## Notion de portée

- Les variables déclarées à l'intérieur d'un bloc sont appelées **variables locales** (au bloc). Elles ne sont accessibles qu'à l'intérieur du bloc.
- Les variables déclarées en dehors de `main` sont de portée **globales** (à la classe). Elles sont accessibles dans toute la classe.

Pour ces variables, on distingue en Java des variables de classes et des variables d'instance.

## Notion de portée

- Les variables déclarées à l'intérieur d'un bloc sont appelées **variables locales** (au bloc). Elles ne sont accessibles qu'à l'intérieur du bloc.
- Les variables déclarées en dehors de `main` sont de portée **globales** (à la classe). Elles sont accessibles dans toute la classe.

Bonne pratique: Déclarer les variables au plus près de leur utilisation.

## Notion de portée

*Déclarer les variables au plus près de leur utilisation*

Par exemple, si la variable `j` n'est pas utilisée après la condition,

écrivez:

```
if (i != 0) {  
    int j = 0;  
  
    ...  
    j = 2 * i;  
    ...  
}
```

plutôt que:

```
int j = 0;  
if (i != 0) {  
  
    ...  
    j = 2 * i;  
    ...  
}
```

## Notion de portée

La **portée** d'une variable, c'est l'ensemble des lignes de code où cette variable est accessible, autrement dit où elle est définie, existe, a un sens.

```
if (i != 0) {  
    int j = 0;  
  
    ...  
    j = 2 * i;  
    ...  
    if (j != 2) {  
        int k = 0;  
  
        ...  
        k = 3 * i;  
        ...  
    }  
    ...  
}
```



## Portée : règle

```
if (i != 0) {  
    int j = 0;
```

```
    ...  
    j = 2 * i;
```

En Java, on ne peut pas utiliser le nom d'une variable déclarée plus globalement pour déclarer une autre variable.

```
    ...  
    if (j != 2) {  
        int j = 0; // interdit
```

```
        ...  
        j = 3 * i;  
        ...  
    }  
    ...  
}
```

Cela permet d'éviter des ambiguïtés entre noms de variables.

## Portée : cas des itérations

La déclaration d'une variable à l'intérieur d'une itération est une déclaration **locale au bloc de la boucle**, et aux deux instructions de test et d'incrément:

```
for(int i = 0; i < 5; ++i) {  
    System.out.println(i);  
}  
// A partir d'ici, on ne peut plus utiliser ce i
```