

## Programme

- . Introduction à la programmation informatique et aux différents langages de programmation.
- . Présentation des concepts de base de la programmation de haut niveau. Aspect algorithmique. Introduction au langage Java.
- . Syntaxe de base du langage Java, la sémantique et le concept de la compilation.
- . Les variables, types et opérations mathématiques
- . Opérateurs, affectation, ....
- . **conditionnelles et boucles (2)**
- . Tableaux ( accès aux composants, ...).
- . Procédures et Fonctions.
- . Présentations de quelques classes clés.
- . Des algorithmes fondamentaux.
- . Des exemples applicatifs de l'ensemble des notions de Java présentées.

## Aujourd'hui conditionnelles et boucles

- Introduction
- Définition : Boucle
- entre parenthèse : la classe Scanner.
- for
- while
- do ... while

On commence par

un exemple (1/2)

Écrivons un programme qui affiche à l'écran 5 lignes de 4 étoiles.

On commence par  
un exemple (2/2)

Une des solutions.

```
public class Dessin {  
    public static void main (String[] args) {  
        System.out.println("****");  
        System.out.println("****");  
        System.out.println("****");  
        System.out.println("****");  
        System.out.println("****");  
    }  
}
```

Sinon, pour une solution plus élégante, on utilise les boucles: Elles s'appellent les boucles ou instructions d'itérations.

## Les boucles: définition

- Une boucle est une structure qui permet de répéter les mêmes instructions plusieurs fois.
- Tout comme pour les conditions, il y a plusieurs façons de réaliser des boucles :
  - while
  - do ... while
  - for

## Les boucles: Définition

Définition:

- Le rôle des boucles est de répéter un certain nombre de fois les mêmes opérations.
- Les boucles permettent à un programme de recommencer depuis le début, pour attendre une action précise de l'utilisateur, parcourir une série de données, etc.
- Une boucle s'exécute tant qu'une condition est remplie.
- Une boucle commence par une déclaration. Cela veut dire, à peu de chose près, « tant que ». Puis nous avons une condition : c'est elle qui permet à la boucle de s'arrêter.
- Une boucle n'est utile que lorsque nous pouvons la contrôler, et donc lui faire répéter une instruction un certain nombre de fois.

## La boucle for (1/5)

### Syntaxe

```
for (initialisation; test; incrémantation)
```

```
{ corps de la boucle; }
```

- la partie initialisation est exécutée une fois, avant de commencer la boucle.

Il s'agit généralement de l'initialisation d'une variable et éventuellement sa déclaration.

- la partie test est évaluée avant de rentrer dans la boucle, à chaque répétition. Si le test a pour résultat la valeur true (vrai), alors le corps de la boucle est exécuté. Si le résultat est false (faux), la boucle est terminée. Le programme passe à l'expression suivante, après la boucle.

## La boucle for (2/5)

### Syntaxe

```
for (initialisation; test; incrémentation)  
{ corps de la boucle; }
```

- la partie incrémentation est exécutée après le corps de la boucle, avant de refaire le test pour un tour de plus. Il s'agit généralement de l'incrémentation d'une variable.
- le corps de la boucle comporte n'importe quelle suite d'instructions.
- Appliquons la boucle for au premier exemple.

## La boucle for (3/5)

La solution en utilisant la boucle for:

**zackage** boucles;

```
public class Rectangle {  
    public static void main (String[] args) {  
        for (int i=0;i<5;i=i+1){  
            System.out.println("****");  
        }  
    }  
}
```

En effet:

- Répète  $i$  fois, pour  $i$  allant de 0 à 4 les instructions qui sont dans les accolades, c'est à dire dans notre cas : afficher une ligne de 4 \*.
- Cela revient bien à dire : répète 5 fois "afficher une ligne de 4 étoiles".

## La boucle for (4/5)

```
for (int i=0;i<5;i=i+1){  
    System.out.println("****");  
}
```

Détaillons cela :

- La boucle for fonctionne avec un compteur du nombre de répétition qui est géré dans les 3 expressions entre les parenthèses.
- La première : int i=0 donne un nom à ce compteur (i) et lui donne une valeur initiale 0.
- Ce compteur ne sera connu qu'à l'intérieur de la boucle for. (il a été déclaré dans la boucle for int i)
- La troisième : i=i+1 dit que les valeurs successives de i seront 0 puis 0+1 puis 1+1, puis 2+1 etc.
- La seconde (i<5) dit quand s'arrête l'énumération des valeurs de i : la première fois que i<5 est faux.

Grâce à ces 3 informations nous savons que i va prendre les valeurs successives 0, 1, 2, 3, 4. Pour chacune de ces valeurs successives, on répètera les instructions dans les accolades.

## La boucle for (5/5)

Analysons les deux codes suivants

```
package boucles;

public class Rectangle {
public static void main (String[] args) {
for (int i=0;i<5;i=i+2){
System.out.println("****");
}
}
}
```

```
public class Rectangle {

public static void main (String[] args) {
for (int i=1;i<5;i=i+2){
System.out.println("****");
}
}
}
```

Quel sera le résultat dans chaque cas?

Quel valeur prendra i dans chaque cas?

## Entre parenthèse: lire les entrées clavier (1/3)

- Pour que Java puisse lire ce que vous tapez au clavier, vous allez devoir utiliser un objet de type Scanner.
- Cet objet peut prendre différents paramètres, mais ici nous n'en utiliserons qu'un : celui qui correspond à l'entrée standard en Java. Lorsque vous faites `System.out.println();` vous appliquez la méthode `println()` sur la sortie standard ;
- Pour lire les entrées clavier, nous allons utiliser l'entrée standard `System.in`.

## Entre parenthèse: lire les entrées clavier (2/3)

Pour indiquer à Java ce que nous allons taper au clavier :

- il faut importer la classe Scanner:

  - import java.util.Scanner;

- il faut, instancier un objet Scanner

  - Scanner sc = new Scanner(System.in);

Ecrire le code suivant dans une classe Main :

```
Scanner sc = new Scanner(System.in);
```

```
System.out.println("Veuillez saisir un mot :");
```

```
String str = sc.nextLine();
```

```
System.out.println("Vous avez saisi : " + str);
```

Et compiler

Entre parenthèse:

lire les entrées clavier (3/3)

```
package boucles;

import java.util.Scanner;
public class Rectangle {
public static void main (String[] args) {

Scanner sc = new Scanner(System.in);
System.out.println("Veuillez saisir un mot :");
String str = sc.nextLine();
System.out.println("Vous avez saisi : " + str);

}
}
```

## La boucle While (1/3)

Syntaxe:

```
while /* Condition */  
{ // Instructions à répéter}
```

Exemple :

```
package boucles;  
  
import java.util.Scanner;  
public class Rectangle {  
public static void main (String[] args) {  
  
    int a = 1, b = 15;  
    while (a < b)  
    {  
        System.out.println("coucou " +a+ " fois !!");  
        a++;  
    }  
}  
}
```

## La boucle While (2/3)

### Exemple applicatif

- Nous allons afficher « Bonjour, <un prénom> », prénom qu'il faudra taper au clavier ; puis nous demanderons si l'on veut recommencer
- Pour cela, il nous faut une variable qui va recevoir le prénom, donc dont le type sera String ainsi que définir le nombre de fois que la boucle soit exécutée ( exemple 5 fois).

## Exemple de solution:

```
package boucles;

import java.util.Scanner;
public class Rectangle {
public static void main (String[] args) {
    //Une variable vide
    String prenom;
    //On initialise celle-ci à 0 pour oui
    char reponse = '0';
    //Notre objet Scanner, n'oubliez pas l'import de java.util.Scanner !
    Scanner sc = new Scanner(System.in);
    //Tant que la réponse donnée est égale à oui...
    while (reponse == '0')
    {
        //On affiche une instruction
        System.out.println("Donnez un prénom : ");
        //On récupère le prénom saisi
        prenom = sc.nextLine();
        //On affiche notre phrase avec le prénom
        System.out.println("Bonjour " +prenom+ ", comment vas-tu ?");
        //On demande si la personne veut faire un autre essai
        System.out.println("Voulez-vous réessayer ? (O/N)");
        //On récupère la réponse de l'utilisateur
        reponse = sc.nextLine().charAt(0);
    }
    System.out.println("Au revoir...");
    //Fin de la boucle}}
```

## La boucle do ... While (1/3)

Syntaxe:

```
do{ //Instructions  
}while(acondition);
```

La boucle do ... while et la boucle while ne sont pas cousines, mais plutôt sœurs. Leur fonctionnement est identique à deux détails près :

- **Première différence:** la boucle do... while s'exécutera au moins une fois, contrairement à sa sœur. C'est-à-dire que la phase de test de la condition se fait à la fin, car la condition se met après le while.

**Deuxième différence:** c'est une différence de syntaxe, qui se situe après la condition du while. Il y a un « ;» après le while.

## La boucle do ... While (2/3)

Refaire le programme précédent avec une boucle do... while.

## La boucle do ... While (3/3)

Exemple de solution

```
import java.util.Scanner;  
public class Rectangle {  
public static void main (String[] args) {  
  
    String prenom = new String();  
    //Pas besoin d'initialiser : on entre au moins une fois dans la boucle !  
    int a = 1;  
    int b= 3;  
  
    Scanner sc = new Scanner(System.in);  
  
    do{  
        System.out.println("Donnez un prénom : ");  
        prenom = sc.nextLine();  
        System.out.println("Bonjour " +prenom+ ", comment vas-tu ?");  
        a++;  
    }while (a<b);  
  
    System.out.println("Au revoir...");  
}  
}
```