

Méthodes static

```
public class Bonjour {  
    public static void main(String[] args) {  
        System.out.print("Quel est ton nom ? ");  
        String nom = Console.lireString();  
        direBonjour(nom);  
  
        int premierNombre = 40;  
        int deuxiemeNombre = 2;  
        int troisiemeNombre = somme(premierNombre, deuxiemeNombre); // 42  
    }  
  
    public static void direBonjour(String nom) {  
        System.out.printf("Bonjour %s !\n", nom);  
    }  
  
    private static int somme(int a, int b) {  
        int c = a + b;  
        return c;  
    }  
}
```

Une méthode (ou fonction) est un bloc de code qui peut être réutiliser plusieurs fois. En fonction de comment elle est définie, une méthode peut renvoyer une certaine valeur ou rien, et peut prendre des paramètres ou pas.

Pour mieux comprendre voici comment on peut décomposer notre méthode `direBonjour` au dessus :

- `public` indique l'accessibilité. Elle signifie que d'autres classes peuvent aussi utiliser cette méthode. Au contraire, si cela serait `private` alors seul la classe `Bonjour` pourrait y accéder
- `static` indique que ce n'est pas un "objet" de la classe `Bonjour`, on verra plus tard ce que cela signifie
- `void` signifie que la méthode ne retourne aucune valeur
- `direBonjour` est le nom de notre méthode
- `(String nom)` indique que la méthode ne prends qu'un argument de type String, qui va être appelé "nom" dans le bloc de la méthode.

Dans le deuxième exemple :

- `private` indique que la méthode ne peut être utilisée que dans la classe `Bonjour`
- `static` même chose que pour `direBonjour`
- `int` indique que la méthode retourne un type "int"
- `somme` est le nom de la méthode
- `(int a, int b)` indique que la méthode prends 2 arguments, tout deux de type "int" qui seront appelé dans la méthode "a" et "b".
- `return` indique une valeur à retourner de la fonction. Une fois que la fonction a retourné quelque chose, tout code écrit après ce `return` ne sera pas exécuté.

Mais il y a aussi la fonction main ! On a l'a tout le temps utilisé mais c'est aussi une méthode. La seule différence est que quand on exécute une classe, Java va chercher cette fonction "main" dans notre classe pour l'exécuter. Si il n'y a pas de fonction main, la classe n'est pas exécutable à elle seule.

Plusieurs méthodes avec le même nom

```
public static void main(String[] args) {  
    System.out.printf("12 + 15 = %d%n", somme(12, 15));  
    System.out.printf("12.5 + 13.0 = %f%n", somme(12.5, 13.0));  
}  
  
public static int somme(int a, int b) {  
    return a + b;  
}  
  
public static double somme(double a, double b) {  
    return a + b;  
}
```

Dans java, contrairement à d'autres langage il peut y avoir plusieurs méthodes qui ont le même nom tant qu'elle ne prennent pas les mêmes arguments (paramètres).

Dans ce cas ci, on a deux variables "somme" mais une prends des types "int" et l'autre prends des types "double". Donc elles peuvent toutes les deux exister dans la même classe sans problème.

Appeler une méthode

Comme vu dans les exemples ci dessus, on peut appeler une méthode quand on est dans la même classe qu'elle. Mais on peut aussi appeler une méthode qui est dans une autre classe.

```
public static void main(String[] args) {  
    int input = io.Console.lireInt();  
}
```

Dans l'exemple ici on appelle la méthode `lireInt` qui est dans la classe `Console` du package `io` par exemple.

En savoir plus

- [Wikiversity FR - Méthodes](#)

Revision #1

Created 27 April 2023 04:31:29 by SnowCode

Updated 27 April 2023 04:35:12 by SnowCode