



Java Programming

Chapter 6

Method

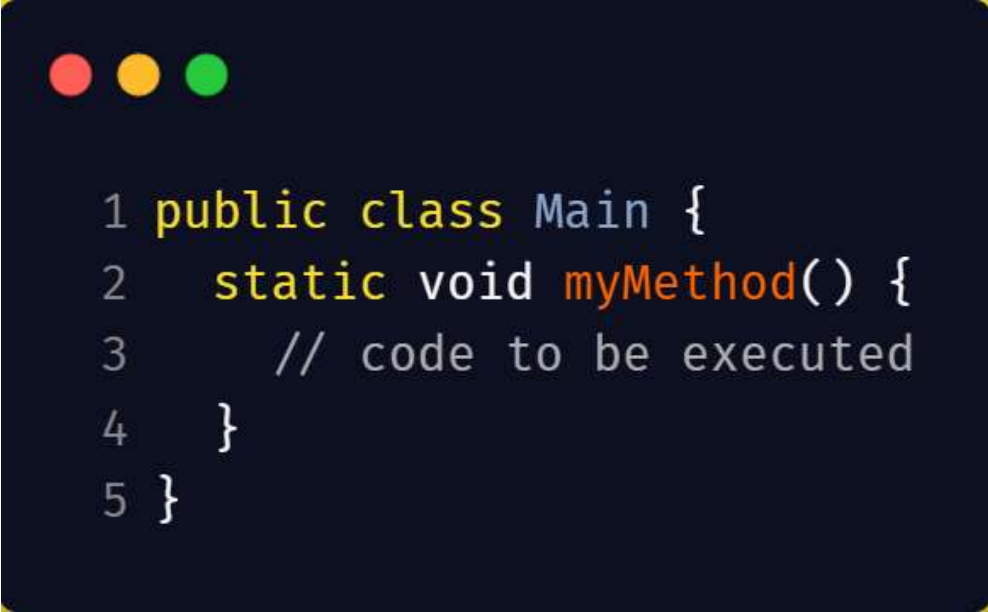
Method

- A method is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a method.
- Methods are used to perform certain actions, and they are also known as functions.
- Why use methods? To reuse code: define the code once, and use it many times.

Method

- Create a Method
 - A method must be declared within a class.
 - It is defined with the name of the method, followed by parentheses ().
 - Java provides some pre-defined methods,
 - `System.out.println()`

Method

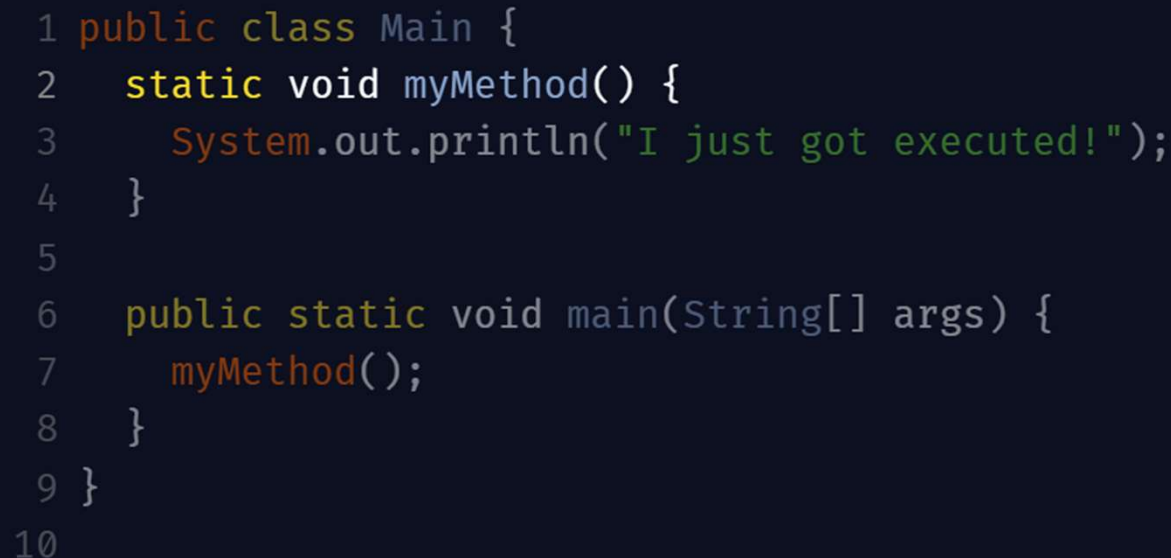


```
1 public class Main {  
2     static void myMethod() {  
3         // code to be executed  
4     }  
5 }
```

- **myMethod()** is the name of the method
- **static** means that the method belongs to the Main class and not an object of the Main class. You will learn more about objects and how to access methods through objects later in this tutorial.
- **void** means that this method does not have a return value. You will learn more about return values later in this chapter

Method

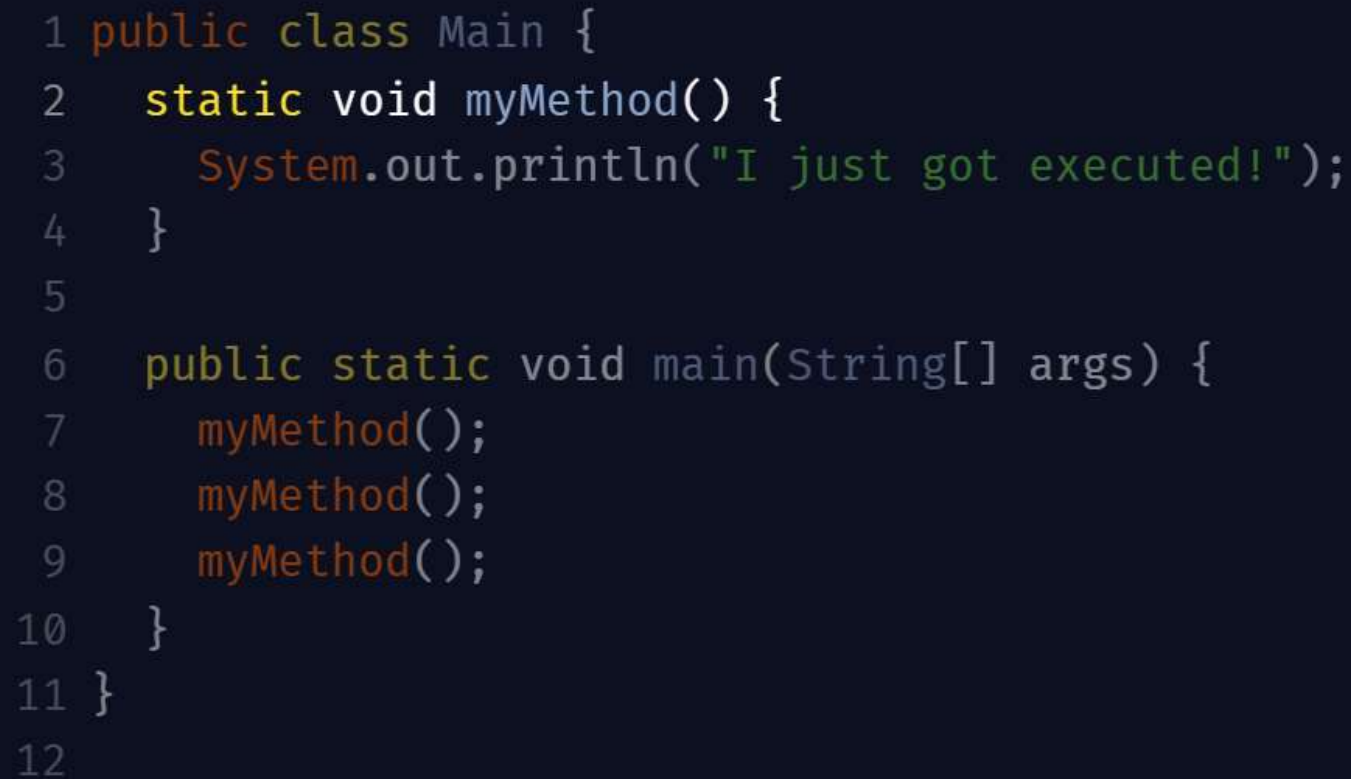
- Call a Method
 - to call a method in Java, write the method's name followed by two parentheses () and a semicolon;
 - In the following example, myMethod() is used to print a text (the action), when it is called:



```
1 public class Main {  
2     static void myMethod() {  
3         System.out.println("I just got executed!");  
4     }  
5  
6     public static void main(String[] args) {  
7         myMethod();  
8     }  
9 }  
10
```

Method

- A method can also be called multiple times:



```
1 public class Main {  
2     static void myMethod() {  
3         System.out.println("I just got executed!");  
4     }  
5  
6     public static void main(String[] args) {  
7         myMethod();  
8         myMethod();  
9         myMethod();  
10    }  
11 }  
12
```

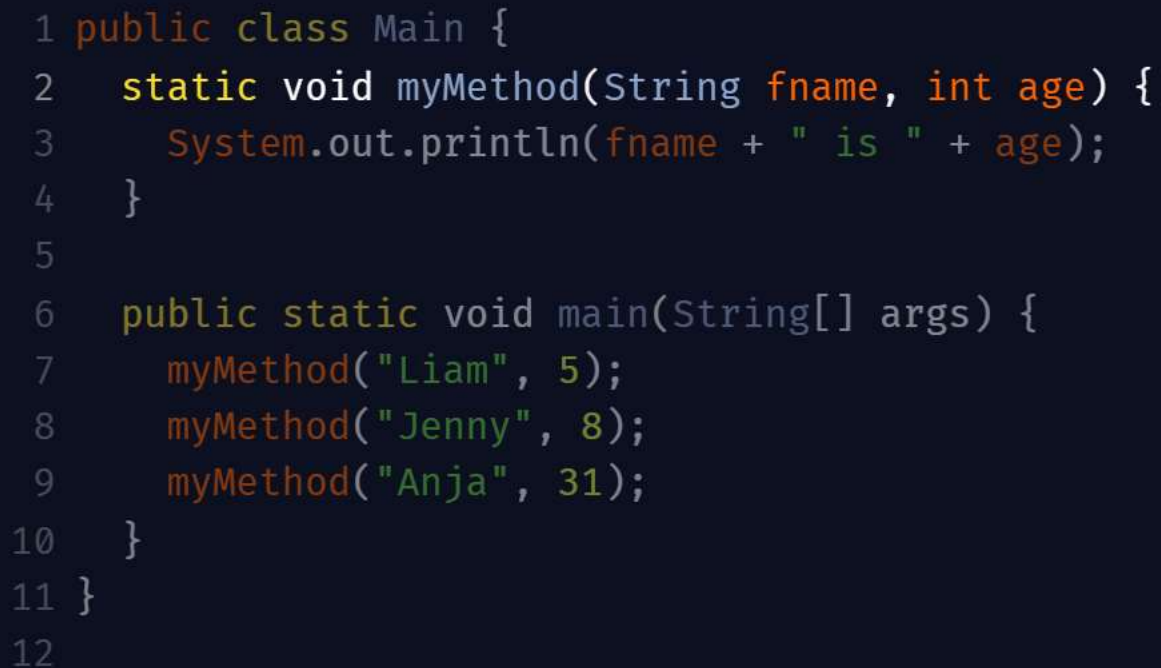
Method Parameters

- Parameters and Arguments
 - Information can be passed to methods as a parameter. Parameters act as variables inside the method.
 - Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

```
1 public class Main {  
2     static void myMethod(String fname) {  
3         System.out.println(fname + " Refsnes");  
4     }  
5  
6     public static void main(String[] args) {  
7         myMethod("Liam");  
8         myMethod("Jenny");  
9         myMethod("Anja");  
10    }  
11 }  
12
```

Method Parameters

Multiple Parameters



```
1 public class Main {  
2     static void myMethod(String fname, int age) {  
3         System.out.println(fname + " is " + age);  
4     }  
5  
6     public static void main(String[] args) {  
7         myMethod("Liam", 5);  
8         myMethod("Jenny", 8);  
9         myMethod("Anja", 31);  
10    }  
11 }  
12
```


Method Parameters

- Example
 - It is common to use if...else statements inside methods:

```
1 public class Main {  
2  
3     // Create a checkAge() method with an integer parameter called age  
4     static void checkAge(int age) {  
5  
6         // If age is less than 18, print "access denied"  
7         if (age < 18) {  
8             System.out.println("Access denied - You are not old enough!");  
9  
10            // If age is greater than, or equal to, 18, print "access granted"  
11            } else {  
12                System.out.println("Access granted - You are old enough!");  
13            }  
14  
15        }  
16  
17        public static void main(String[] args) {  
18            checkAge(20); // Call the checkAge method and pass along an age of 20  
19        }  
20    }
```

Java Return

- Return Values
 - In the previous page, we used the void keyword in all examples (like static void myMethod(int x)), which indicates that the method should not return a value.
 - If you want the method to return a value, you can use a primitive data type (such as int, char, etc.) instead of void, and use the return keyword inside the method:

```
1 public class Main {  
2     static int myMethod(int x) {  
3         return 5 + x;  
4     }  
5  
6     public static void main(String[] args) {  
7         System.out.println(myMethod(3));  
8     }  
9 }  
10
```

Java Return

- This example returns the sum of a method's **two parameters**:

```
1 public class Main {  
2     static int myMethod(int x, int y) {  
3         return x + y;  
4     }  
5  
6     public static void main(String[] args) {  
7         System.out.println(myMethod(5, 3));  
8     }  
9 }  
10  
11
```

Method Overloading

- With **method overloading**, multiple methods can have the same name with different parameters:
 - `int myMethod(int x)`
 - `float myMethod(float x)`
 - `double myMethod(double x, double y)`

Method Overloading

Consider the following example, which has two methods that add numbers of different type:

```
1 public class Main {
2     static int plusMethodInt(int x, int y) {
3         return x + y;
4     }
5
6     static double plusMethodDouble(double x, double y) {
7         return x + y;
8     }
9
10    public static void main(String[] args) {
11        int myNum1 = plusMethodInt(8, 5);
12        double myNum2 = plusMethodDouble(4.3, 6.26);
13        System.out.println("int: " + myNum1);
14        System.out.println("double: " + myNum2);
15    }
16 }
17
```

Method Overloading

- Instead of defining two methods that should do the same thing, it is better to overload one.
- In the example below, we overload the plusMethod method to work for both int and double:

```
1 public class Main {
2     static int plusMethodInt(int x, int y) {
3         return x + y;
4     }
5
6     static double plusMethodDouble(double x, double y) {
7         return x + y;
8     }
9
10    public static void main(String[] args) {
11        int myNum1 = plusMethodInt(8, 5);
12        double myNum2 = plusMethodDouble(4.3, 6.26);
13        System.out.println("int: " + myNum1);
14        System.out.println("double: " + myNum2);
15    }
16 }
17
```

```
1 public class Main {
2     static int plusMethod(int x, int y) {
3         return x + y;
4     }
5
6     static double plusMethod(double x, double y) {
7         return x + y;
8     }
9
10    public static void main(String[] args) {
11        int myNum1 = plusMethod(8, 5);
12        double myNum2 = plusMethod(4.3, 6.26);
13        System.out.println("int: " + myNum1);
14        System.out.println("double: " + myNum2);
15    }
16 }
17
```

Method Overloading