Introduction to Java

By Sanjana Bandara

Skill Lab

Wings Your Skill to find Your own path

Contents

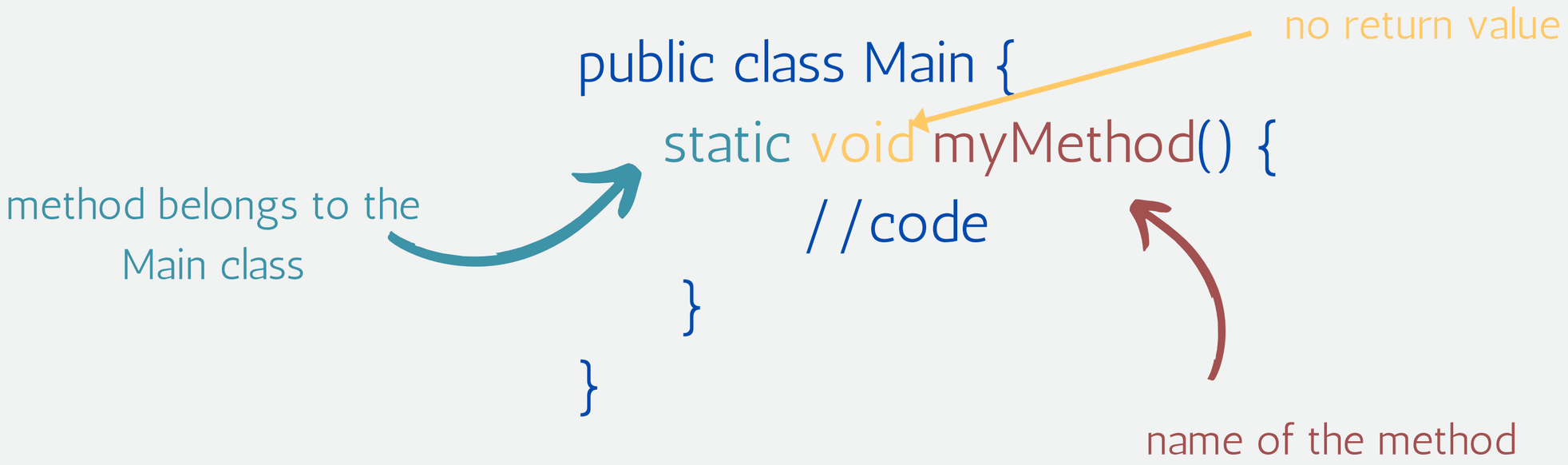
Java Methods

- Is a block of code
- Only runs when it is called
- Can pass data into a method - known as PArameters
- Used to perform actions - known as Functions
- To reuse code - Define the code once, and use it many times
- Declare within a class
- Java provides predefined functions too

  Example : System.out.println()

Java Methods

- Example



Skill Lab
Wings Your Skill to find Your own path

Java Methods

Call a Method

- write the method's name followed by two parentheses () and a semicolon;
- Example: myMethod();

```
public class Main {
    static void myMethod() {
        System.out.println("I just got executed!");
    }

    public static void main(String[] args) {
        myMethod();
    }
}

// Outputs "I just got executed!"
```

Skill Lab
Wings Your Skill to find Your own path

Java Methods

```
public class Main {
    static void myMethod() {
        System.out.println("I just got executed!");
    }

    public static void main(String[] args) {
        myMethod();
        myMethod();
        myMethod();
    }
}

// Outputs "I just got executed!"
```

Java Methods Parameters

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

```
public class Main {
        static void myMethod(String fname) {
                System.out.println(fname + "Refsnes");
        }
        public static void main(String[] args) {
myMethod("Liam");
myMethod("Jenny");
myMethod("Anja");
}
}
```

Skill Lab
Wings Your Skill to find Your own path

Java Methods Parameters

- void - indicates that the method should not return a value
- To return a value, use primitive data type instead of void and then return keyword inside the method.

```
public class Main {
static int myMethod(int x) {
return 5 + x;
}

public static void main(String[] args) {
System.out.println(myMethod(3));
}
}
```

Skill Lab
Wings Your Skill to find Your own path

Java Methods Parameters

```
public class Main {
static int myMethod(int x, int y) {
return x + y;
}

public static void main(String[] args) {
System.out.println(myMethod(5, 3));
}
}
```

Java Methods Parameters

- Store the resault in a variable

```
public class Main {
static int myMethod(int x, int y) {
return x + y;
}

public static void main(String[] args) {
int z = myMethod(5, 3);
System.out.println(z);
}

}// Outputs 8 (5 + 3)
```

Java Methods Parameters

- A method with if else

```
public class Main {
static void checkAge(int age) {
if (age < 18) {
System.out.println("Access denied - You are not old enough!");
} else {
System.out.println("Access granted - You are old enough!");}
}
public static void main(String[] args)
{
checkAge(20);
}
}
```

Java Method Overloadina

- 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)
```

```
public class Main {
  static int plusMethodInt(int x, int y) {
    return x + y;
  }
  static double plusMethodDouble(double x, double y) {
    return x + y;
  }
  public static void main(String[] args) {
    int myNum1 = plusMethodInt(8, 5);
    double myNum2 = plusMethodDouble(4.3, 6.26);
    System.out.println("int: " + myNum1);
    System.out.println("double: " + myNum2);
  }}
```

Skill Lab
Wings Your Skill to find Your own path

Java Method Overloadina

- 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)
```

```
public class Main {
  static int plusMethod(int x, int y) {
    return x + y;
  }
static double plusMethod(double x, double y) {
    return x + y;
  }
public static void main(String[] args) {
    int myNum1 = plusMethod(8, 5);
    double myNum2 = plusMethod(4.3, 6.26);
    System.out.println("int: " + myNum1);
    System.out.println("double: " + myNum2);
  }
}
```

Java Scope

- Variables are only accessible inside the region and are created. This is called Scope.

Method Scope

- Variables declared directly inside a method are available anywhere in the method.

```
public class Main {
public static void main(String[] args) {
    // Code here CANNOT use x
int x = 100;

// Code here can use x
    System.out.println(x);
  }
}
```

Skill Lab
Wings Your Skill to find Your own path

Java Scope

Block Scope

- A block of code refers to all of the code between curly braces {}.
- Variables declared inside blocks of code are only accessible by the code between the curly braces

```
public class Main {
public static void main(String[] args) {
// Code here CANNOT use x{
// This is a block
// Code here CANNOT use x

int x = 100;
// Code here CAN use x
      System.out.println(x);
}
// The block ends here// Code here
CANNOT use x}}
```

Java Recursion

Recursion is the technique of making a function call itself.

```
public class Main {
public static void main(String[] args) {
int result = sum(10);
System.out.println(result);
}
public static int sum(int k)
 {
if (k > 0) {
return k + sum(k - 1);
}
else {return 0;}
}
}
```

```
10 + sum(9)
10 + ( 9 + sum(8) )
10 + ( 9 + ( 8 + sum(7) ) )
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 +
sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
```

Skill Lab
Wings Your Skill to find Your own path