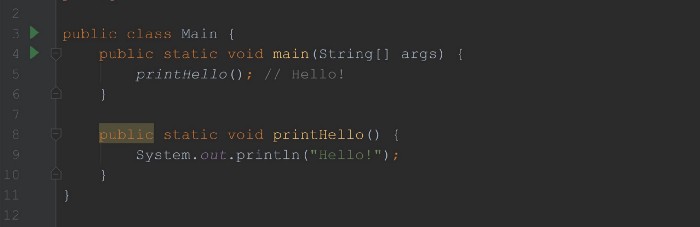
1. Method overview.
2. Method arguments.
3. Method return type.

Method overview

A method is a block of code that can be executed when it’s called by its name. The methods are actions in the program. Also, it’s useful to think that method is a separate program that has input, process, and output.

The core idea of creating methods is reusability. We can create a method once and reuse it many times. However, be careful to create extremely generic methods. These kinds of methods are messy and hard to read. The method should solve only one problem.

In OOP methods are behaviors of the objects and they usually manipulate with object properties.



in Main.java

In the above example, we created our method printHello and used it in the main method. The output of this code is Hello!.

Let’s break down our method

public static void printHello() {  
 System.out.println("Hello!");  
}

public — access modifier. it will define from where we can use this method. Public methods can be used from anywhere in the project.

static — it means this method belongs to the class and not to a specific object. In this case, I made it static to use directly in the main method.

void — it’s the return type of our method. Methods can return some data back to the place where it was called. void means method doesn’t return anything and just performs a specific action. While creating, we need to decide what data type method will return.

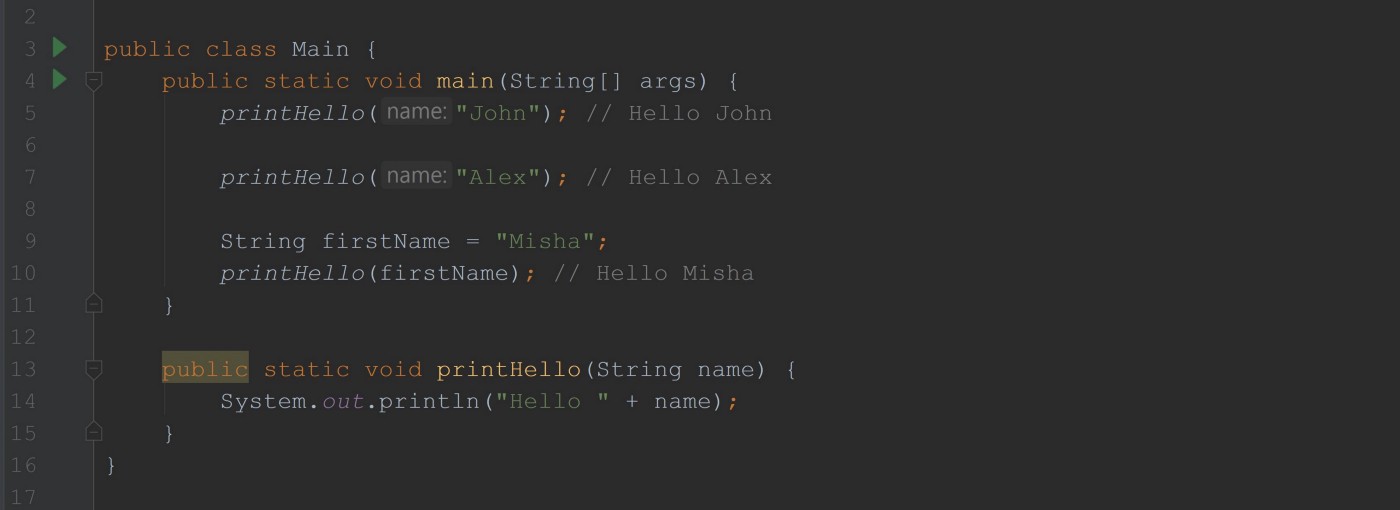
printHello — the name of the method. We call our method by its name to use it.

() — method arguments. Here we have no arguments for our method. Arguments are used to pass data as input to our methods.

{} — the body of the method. The actual code that will be executed goes into the body.

Method arguments

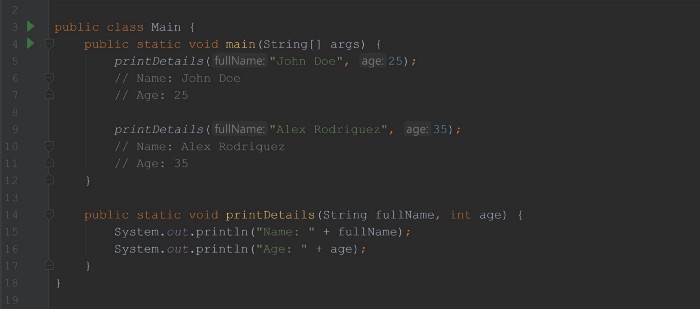
The method argument is the way we pass our input data to our method. In our first example, the printHello() method didn’t have arguments. So when using it, we didn’t have to pass any data.



in Main.java

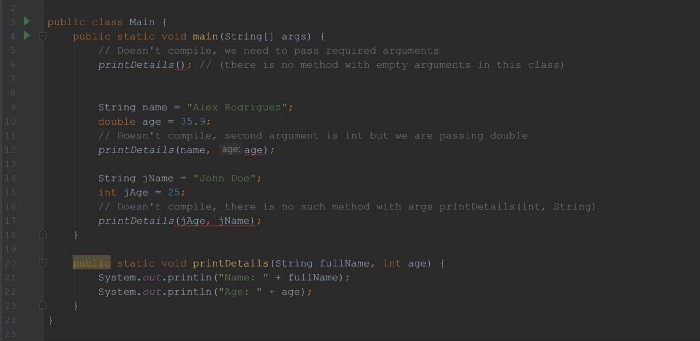
In the above example, we have a method that has one String argument. We declare method arguments between parenthesis after the name.

* We have to predefine our argument data type. In our case, we have one String argument. As you can see, when we use our method we actually pass the String value. It will not compile if we pass different data types, for example int.
* We used our method three times in the above program. And we have three different outputs. When used our method as printHello("John"); program printed Hello John so the argument name will take a value "John". When it’s called as printHello("Alex") on line 7 it will print Hello Alex because we gave a different value to our method. So that’s how we pass data into our methods via method arguments.
* There is no limit to the number of arguments. Good practice to keep it up to 3 or max 4 arguments for a method. If you need to pass more data you can use Array or pass them as Object. We will learn about them the future articles.



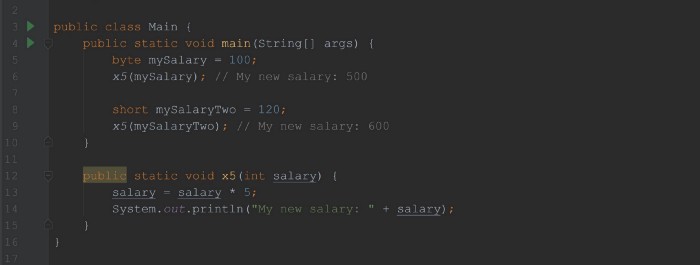
in Main.java

In the above example, we have a method printDetails(String fullName, int age)that accepts two arguments.



Does not compile

The above code does not compile. The number of arguments and their type should match when calling our method.



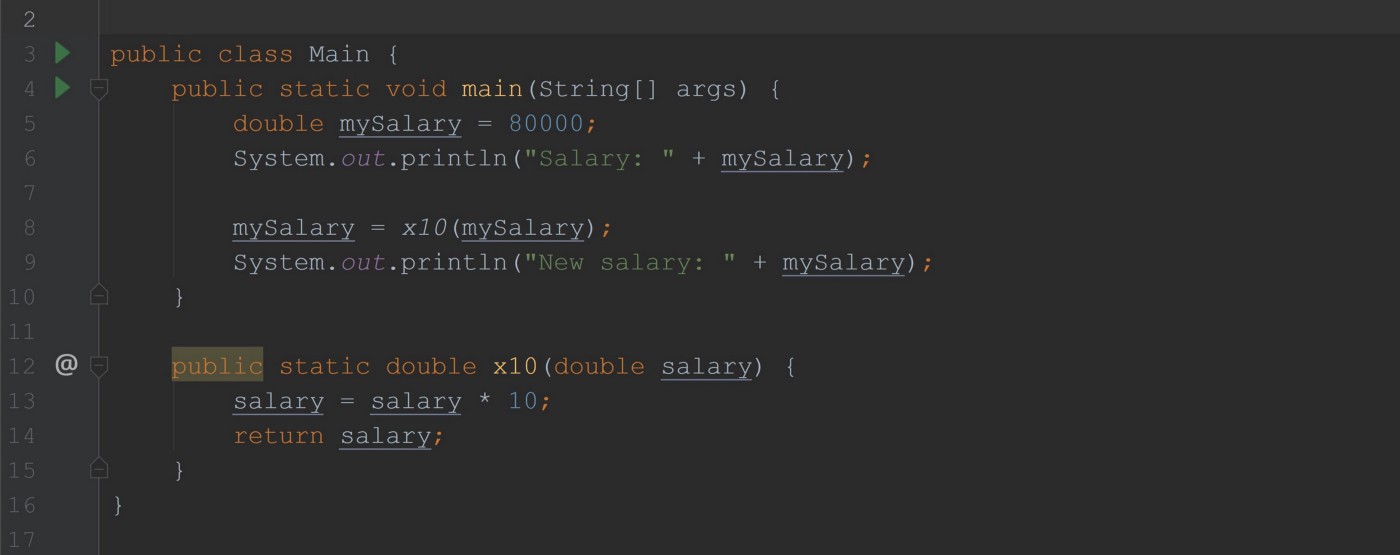
in Main.java

In the above example, our method has one int argument but we are passing byte and then short in the main method. Everything works fine because byte and short can be autoboxed to an int automatically and Java allows us to pass them to int argument. We will have the same behaviors for auto casting when working with objects which have a parent-child relationship.

Method return type

As we discussed a bit, methods can return some data back. So far we have been using only void methods. They just perform actions and do not return any data back.

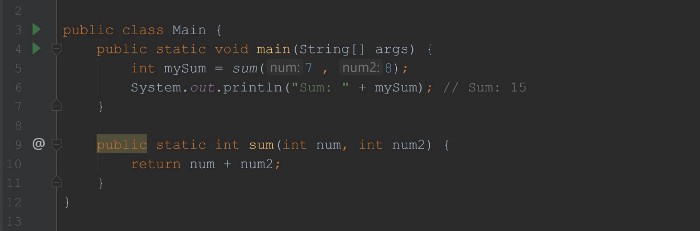
In order to create a method that returns something, we need to use the method return type.



in Main.java

Now, we are increasing our salary by 10! This time instead of printing, we are increasing the method argument salary by 10 and returning it.

* If you specify that method returns something then it must return. Code will not compile if you are not returning a promising data type.
* To return something we use return keyword. It’s important to remember that after returning, our program will exit from the method. You cannot have other statements after the return statement unless the return statement is under some condition.
* We need to return the exact type that we specified as the method return type.



in Main.java

* We can use our method that returns something and assign the return value to the same variable type.

Let’s answer some questions:

How do I know when I need to create a separate method?

When we write a duplicate code. If you catch yourself typing something similar twice it’s might be a time to create a method. Also, methods are used as object behaviors so while designing the object you will create them.

Should I always create static methods?

No, I just used examples with the static method to simplify the code. If a method is a behavior of a specific object, it shouldn’t be static. Usually, static methods are helper methods of util classes.