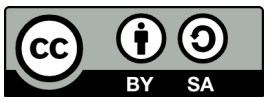
# Methods

Java Fundamentals

Libre Education



# Methods

A method is a block of a code that can be called to perform a specific function, it can have values passed to it and return a value. The advantage of using methods is that it allows you to execute the same block of code any number of times from anywhere in your program by calling it with just one line, thus eliminating the need to rewrite the same code multiple times.

## Structure of a Methods

A method has a number of different components that are required to it, although some are optional:

1. Modifiers - includes the access and non-access modifiers
2. Return Type - the type of data that the method will return. If the method will not return any data the void keyword is used
3. Method Name - the name that will be used when calling the method
4. Parameters - the parameters are the values that need to be given or passed to the method when it is called, parameters are optional
5. Method Body - the code that will be executed when the method is called
6. Return Statement - this statement returns a value from the method when it is called

### Access Modifiers

Access Modifiers set the access level for a method. The access level essentially describes where the method can be called from. There are three types of access modifier:

1. Public (public) - accessible to the whole program
2. Protected (protected) - accessible to the package and all sub classes
3. Private (private) - accessible to the class only

### Non-Access Modifiers

The non-access modifiers provide a set of miscellaneous functionality, which will be discussed at a later point:

1. Static (static)
2. Final (final)
3. Abstract (abstract)
4. Synchronized (synchronized)
5. Volatile (volatile)

### Syntax

```
modifiers return_type method_name(data_type parameter_name){
    // Code to execute when the method is called
    return return_value;
}
```

### Example

```
public static void printSomething(String stringToPrint){
    System.out.println(stringToPrint);
}
```

or

```
public static int addSomething(int integer){
    integer++;
    return integer;
}
// An int that is one more than what is given public static void printSomething
     (String stringToPrint){
    System.out.println(stringToPrint);
}

public static int addSomething(int integer){
    integer++;
    return integer;
}as a parameter is returned
```

## Calling Methods

To use a method it needs to be called. The process of calling a method itself is simple but often has many variations in context depending on number of factors such as if it returns a value or if it is located in the same class or package that is being called from. The following example make use of these two methods:

```
public static void printSomething(String stringToPrint){
    System.out.println(stringToPrint);
}

public static int addSomething(int integer){
    integer++;
    return integer;
}
```

### Calling (void) methods

When calling a void method that is declared in the same class that it is being called from, just the method name has to be stated.

### Syntax

```
method_name(parameter);
```

### Example

```
printSomething("Hello World");
```

## Calling methods with return statements

When calling a method the value that is returned can be used in multiple ways, such as assigning it to a variable or checking the value with an if statement.

### Syntax

```
data_type variable_name = method_name(parameter);
```

or

```
if(method_name(parameter) condition){
   // Code to execute
}
```

### Example

```
int i = addSomething(5);
System.out.println(i);
// Output: 6
```

or

```
if(addSomething(5) > 4){
   System.out.println("The value returned is greater than 4");
}
```

## Calling methods from other classes

When calling a method from another class, how it is called depends on whether the method is static or non-static (i.e. it has no static modifier). If the method is static it can simply be called by using the class name and the method name, if the method is non-static an object of the class needs to be created. The method can then be called using the object. The following examples use the printSomething method located in the Printer class.

### Syntax

From a static context

```
class_name.method_name(parameters);
```

or

From a non-static context

```
class_name object_name = new class_name();
object_name.method_name(parameters);
```

### Example

From a static context

```
Printer.printSomething("Hello World");
```

or

From a non-static context

```
Printer printerObject = new Printer();
printerObject.printSomething("Hello World");
```

## Calling methods from other packages

When calling a method from a class located in a different package, relative to where the method is being called, it can either be called by using the fully qualified name of the method, which includes the package and class or by importing the class or the entire package using the import keyword. The import statement is located above the class declaration. The following examples use the printSomething method of the Printer class which in located in the utils package. In this case printSomething is a static method.

### Syntax

Using fully qualified name

```
package_name.class_name.method_name(parameters);
```

or

Using import statement

```
import package_name.class_name;
class_name.method_name(parameters);
```

### Example

Using fully qualified name

```
utils.Printer.printSomething("Hello World");
```

or

Using import statement

```
import utils.Printer;
...
Printer.printSomething("Hello World");
```

## Other Resources

1. The Java Tutorials (docs.oracle.com/javase/tutorial/java/javaOO/methods.html)
2. TutorialsPoint (www.tutorialspoint.com/java/java_methods.htm)
3. Home & Learn (www.homeandlearn.co.uk/java/java_methods.html)
4. How To Program With Java (howtoprogramwithjava.com/what-is-a-method-in-java/)