**Types of Java methods**

Depending on whether a method is defined by the user, or available in standard library, there are two types of methods:

* Standard Library Methods
* User-defined Methods

**Standard Library Methods**

The standard library methods are built-in methods in Java that are readily available for use. These standard libraries come along with the Java Class Library (JCL) in a Java archive (\*.jar) file with JVM and JRE.

For example,

* print() is a method of java.io.PrintSteam. The print("...") prints the string inside quotation marks.
* sqrt() is a method of Math class. It returns square root of a number.

Here's an working example:

1. public class Numbers {
2. public static void main(String[] args) {
3. System.out.print("Square root of 4 is: " + Math.sqrt(4));
4. }
5. }

When you run the program, the output will be:

Square root of 4 is: 2.0

**User-defined Method**

You can also define methods inside a class as per your wish. Such methods are called user-defined methods.

**How to create a user-defined method?**

Before you can use (call a method), you need to define it.

Here is how you define methods in Java.

public static void myMethod() {

System.out.println(“My Function called”);

}

Here, a method named myMethod() is defined.

You can see three keywords public, static and void before the function name.

* The public keyword makes myMethod() method public. Public members can be accessed from outside of the class. To learn more, visit: Java public and private Modifiers.
* The static keyword denotes that the method can be accessed without creating the object of the class. To learn more, visit: *Static Keyword in Java*
* The void keyword signifies that the method doesn’t return any value. You will learn about returning value from the method later in this article.

In the above program, our method doesn’t accept any arguments. Hence the empty parenthesis (). You will learn about passing arguments to a method later in this article.

The complete syntax for defining a Java method is:

modifier static returnType nameOfMethod (Parameter List) {

// method body

}

Here,

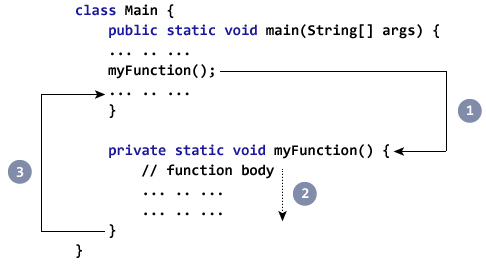
* **modifier** - defines access type whether the method is public, private and so on.
* **static** - If you use static keyword in a method then it becomes a static method. Static methods can be called without creating an instance of a class.  
    
  For example, the sqrt() method of standard Math class is static. Hence, we can directly call Math.sqrt() without creating an instance of Math class.
* **returnType** - A method can return a value.  
    
  It can return native data types (int, float, double etc.), native objects (String, Map, List etc.), or any other built-in and user defined objects.  
    
  If the method does not return a value, its return type is void.
* **nameOfMethod** - The name of the method is an identifier.  
    
  You can give any name to a method. However, it is more conventional to name it after the tasks it performs. For example, calculateInterest, calculateArea, and so on.
* **Parameters (arguments)** - Parameters are the values passed to a method. You can pass any number of arguments to a method.
* **Method body** - It defines what the method actually does, how the parameters are manipulated with programming statements and what values are returned. The codes inside curly braces { } is the body of the method.

**How to call a Java Method?**

Now you defined a method, you need to use it. For that, you have to call the method. Here's how:

myMethod();

This statement calls the myMethod() method that was declared earlier.



1. While Java is executing the program code, it encounters myMethod(); in the code.
2. The execution then branches to the myFunction() method, and executes code inside the body of the method.
3. After the codes execution inside the method body is completed, the program returns to the original state and executes the next statement.

**Example: Complete Program of Java Method**

Let's see a Java method in action by defining a Java class.

1. class Main {
2. public static void main(String[] args) {
3. System.out.println("About to encounter a method.");
4. // method call
5. myMethod();
6. System.out.println("Method was executed successfully!");
7. }
8. // method definition
9. private static void myMethod(){
10. System.out.println("Printing from inside myMethod()!");
11. }
12. }

When you run the program, the output will be:

About to encounter a method.

Printing from inside myMethod().

Method was executed successfully!