Methods :

**Java Methods**

A method is a block of code that performs a specific task.

Suppose you need to create a program to create a circle and color it. You can create two methods to solve this problem:

* a method to draw the circle
* a method to color the circle

Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.

In Java, there are two types of methods:

* **User-defined Methods**: We can create our own method based on our requirements.
* **Standard Library Methods**: These are built-in methods in Java that are available to use.

Let's first learn about user-defined methods.

This is the simple syntax of declaring a method. However, the complete syntax of declaring a method is

modifier static returnType nameOfMethod (parameter1, parameter2, ...) {

// method body

}

## Declaring a Java Method

The syntax to declare a method is:

returnType methodName() {

// method body

}

Here,

* **returnType** - It specifies what type of value a method returns For example if a method has an int return type then it returns an integer value.  
    
  If the method does not return a value, its return type is void.
* **methodName** - It is an [identifier](https://www.programiz.com/java-programming/keywords-identifiers#identifiers) that is used to refer to the particular method in a program.
* **method body** - It includes the programming statements that are used to perform some tasks. The method body is enclosed inside the curly braces { }.

## What is a method in Java?

A **method** is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. It is used to achieve the **reusability** of code. We write a method once and use it many times. We do not require to write code again and again. It also provides the **easy modification** and **readability** of code, just by adding or removing a chunk of code. The method is executed only when we call or invoke it.

.

### **Method Declaration**

The method declaration provides information about method attributes, such as visibility, return-type, name, and arguments. It has six components that are known as **method header**, as we have shown in the following figure.



**Method Signature:** Every method has a method signature. It is a part of the method declaration. It includes the **method name** and **parameter list**.

**Return Type:** Return type is a data type that the method returns. It may have a primitive data type, object, collection, void, etc. If the method does not return anything, we use void keyword.

**Method Name:** It is a unique name that is used to define the name of a method. It must be corresponding to the functionality of the method. Suppose, if we are creating a method for subtraction of two numbers, the method name must be **subtraction().** A method is invoked by its name.

**Parameter List:** It is the list of parameters separated by a comma and enclosed in the pair of parentheses. It contains the data type and variable name. If the method has no parameter, left the parentheses blank.

**Method Body:** It is a part of the method declaration. It contains all the actions to be performed. It is enclosed within the pair of curly braces.

Naming a Method

While defining a method, remember that the method name must be a **verb** and start with a **lowercase** letter. If the method name has more than two words, the first name must be a verb followed by adjective or noun. In the multi-word method name, the first letter of each word must be in **uppercase** except the first word. For example:

**Single-word method name:** sum(), area()

## Types of Method

There are two types of methods in Java:

* Predefined Method
* User-defined Method

### **Predefined Method**

In Java, predefined methods are the method that is already defined in the Java class libraries is known as predefined methods. It is also known as the **standard library method** or **built-in method**. We can directly use these methods just by calling them in the program at any point. Some pre-defined methods are **length(), equals(), compareTo(), sqrt(),** etc. When we call any of the predefined methods in our program, a series of codes related to the corresponding method runs in the background that is already stored in the library.

Each and every predefined method is defined inside a class. Such as **print()** method is defined in the **java.io.PrintStream** class. It prints the statement that we write inside the method. For example, **print("Java")**, it prints Java on the console.

**Demo.java**

1. **public** **class** Demo
2. {
3. **public** **static** **void** main(String[] args)
4. {
5. // using the max() method of Math class
6. System.out.print("The maximum number is: " + Math.max(9,7));
7. }
8. }

### **User-defined Method**

The method written by the user or programmer is known as **a user-defined** method. These methods are modified according to the requirement.

### **How to Create a User-defined Method**

Let's create a user defined method that checks the number is even or odd. First, we will define the method.

1. //user defined method
2. **public** **static** **void** findEvenOdd(**int** num)
3. {
4. //method body
5. **if**(num%2==0)
6. System.out.println(num+" is even");
7. **else**
8. System.out.println(num+" is odd");
9. }

**package** com.corejava;

**public** **class** Main {

**static** **int** *s*= 10;

//create a method

**public** **int** addNumb(**int** a,**int** b) {

**int** Sum =a+b;

**return** Sum;

}

**public** **void** run() {

System.***out***.println("i am running ");

}

**public** **static** **int** getSquare(**int** x) {

**return** x\*3;

}

**public** **int** mulTwo(**int** c,**int** d) {

**int** mul= c\*d;

**return** mul;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

//System.out.println("square root of 16 is:" + Math.sqrt(16));

System.***out***.println(Main.*s*);

Main m = **new** Main();

**int** result = m.addNumb(10, 20);

System.***out***.println(result);

**int** r= m.mulTwo(10, 30);

System.***out***.println(r);

m.run();

**for**(**int** i=1;i<=10;i++) {

**int** v= *getSquare*(i);

System.***out***.println(v);

}

}

}

Naming Conventions:

Mvp –method,variable,package -----------Lower case

Cic-class,interface,constant------------UpparCase