Methods in Java

- **What is method in JAVA?**

Method in Java is similar to a function defined in other programming languages. Method describes the behaviour of an object. A method is a collection of statements that are grouped together to perform an operation.

For example, if we have a class Human, then this class should have methods like eating(), walking(), talking() etc, which describes the behaviour of the object. Declaring method is similar to function.

Syntax to declare the method in Java.

```
return-type methodName(parameter-list)

{

 //body of method

}
```

return-type refers to the type of value returned by the method.

methodName is a valid meaningful name that represent name of a method.

parameter-list represents list of parameters accepted by this method.

Method may have an optional return statement that is used to return value to the caller function.

- **Example of a Method:**

Let's understand the method by simple example that takes a parameter and returns a string value.

```
public String getName(String st)

{

 String name="Welcome";

 name=name+st;

 return name;

}
```

- **method definition in java**

Modifier : Modifier are access type of method.

Return Type : A method may return value. Data type of value return by a method is declare in method heading.

Method name : Actual name of the method.

Parameter : Value passed to a method.

Method body : collection of statement that defines what method does.

- **Calling a Method**

Methods are called to perform the functionality implemented in it. We can call method by its name and store the returned value into a variable.

String val = GetName(" class")

It will return a value Welcome class after appending the argument passed during method call.

- **Returning Multiple values**

In Java, we can return multiple values from a method by using array. We store all the values into an array that want to return and then return back it to the caller method. We must specify return-type as an array while creating an array.

Example:

Below is an example in which we return an array that holds multiple values.

```
class MethodDemo2{
   static int[] total(int a, int b)
   {
int[] s = new int[2];
s[0] = a + b;
s[1] = a - b;
      return s;
   }
```

```
    public static void main(String[] args)
    {
int[] s = total(200, 70);
System.out.println("Addition = " + s[0]);
System.out.println("Subtraction = " + s[1]);
    }
}
```

- **Return Object from Method**

In some scenario there can be need to return object of a class to the caller function. In this case, we must specify class name in the method definition.

Below is an example in which we are getting an object from the method call. It can also be used to return collection of data.

Example:

In this example, we created a method get() that returns object of Demo class.

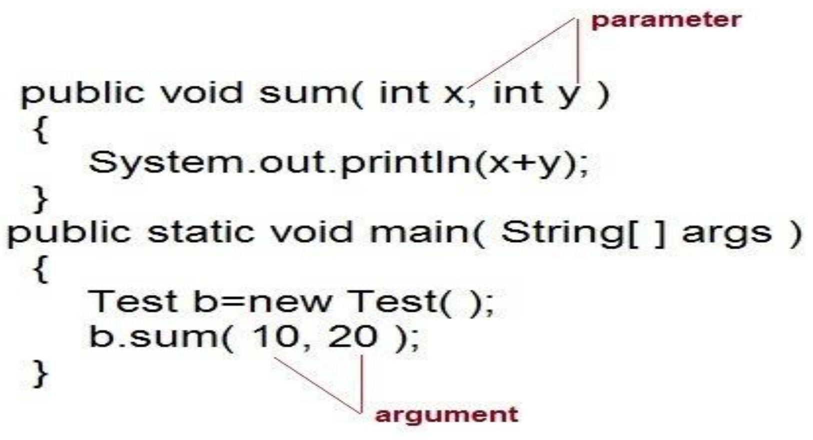
```
class Demo{
int mul;
double div;
int add;
Demo(int m, double d, int a)
    {
        mul = m;
        div = d;
        add = a;
    }
}
```

```
class MethodDemo4{
   static Demo get(int x, int y)
   {
      return new Demo(x * y, (double)x / y, (x + y));
   }
   public static void main(String[] args)
   {
      Demo ans = get(25, 5);
System.out.println("Multiplication = " + ans.mul);
System.out.println("Division = " + ans.div);
System.out.println("Addition = " + ans.add);
   }
}
```

- **Parameter Vs. Argument in a Method**

Parameter is variable defined by a method that receives value when the method is called. Parameter are always local to the method they dont have scope outside the method. While argument is a value that is passed to a method when it is called.

You can understand it by the below image that explain parameter and argument using a program example.



```
public void sum( int x, int y )
 {
     System.out.println(x+y);
 }
public static void main( String[ ] args )
 {
     Test b=new Test( );
     b.sum( 10, 20 );
 }
```

Java is Strictly Pass by Value

In Java, it is very confusing whether java is pass by value or pass by reference.

When the values of parameters are copied into another variable, it is known as pass by value and when a reference of a variable is passed to a method then it is known as pass by reference.

It is one of the features of C language where address of a variable is passed during function call and that reflect changes to original variable.

In case of Java, there is no concept of variable address, everything is based on object. So, either we can pass primitive value or object value during the method call.

Note: Java is strictly pass by value. It means during method call; values are passed not addresses.

Below is an example in which value is passed by value and changed them inside the function but outside the function changes are not reflected to the original variable values.

```
class Add
{
        int x, y;
        Add(int i, int j)
        {
                x = i;
                y = j;
        }
}
class Demo
{
        public static void main(String[] args)
        {
                Add obj = new Add(5, 10);
                // call by value
                change(obj.x, obj.y);
                System.out.println("x = "+obj.x);
                System.out.println("y = "+obj.y);
        }
        public static void change(int x, int y)
        {
                x++;
                y++;
        }
}
```

Let’s take another example in which an object is passed as value to the function, in which we change its variable value and the changes reflected to the original object. It seems like pass by reference but we differentiate it. Here member of an object is changed not the object.

Example:

Below is an example in which value is passed and changes are reflected.

```
class Add
{
        int x, y;
        Add(int i, int j)
        {
                x = i;
                y = j;
```

```
	}
}
class Demo
{
	public static void main(String[] args)
	{
		Add obj = new Add(5, 10);
		// call by value (object is passed)
		change(obj);
		System.out.println("x = "+obj.x);
		System.out.println("y = "+obj.y);

	}
	public static void change(Add new_obj)
	{
		new_obj.x++;
		new_obj.y++;
	}
}
```