

# Methods in Java

**Method** describe behavior of an object. A method is a collection of *statements* that are group together to perform an operation.

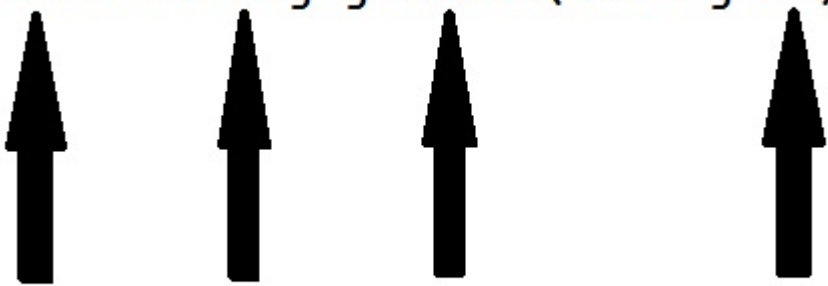
## Syntax:

```
return-type methodName(parameter-list)
{
    //body of method
}
```

## Example of a Method:

```
public String getName(String st)
{
    String name="StudyTonight";
    name=name+st;
    return name;
}
```

**public String getName(String st)**



modifier    return-type    method-name    parameter

**Modifier:** Modifier are access type of method. We will discuss it in detail later.

**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.

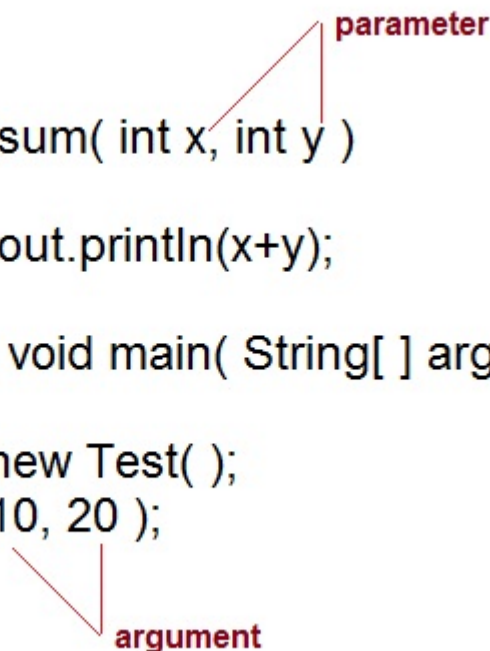
**Method body:** collection of statement that defines what method does.

---

## Parameter Vs. Argument

While talking about method, it is important to know the difference between two terms **parameter** and **argument**.

```
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 );  
}
```



## call-by-value and call-by-reference

There are two ways to pass an argument to a method

1. **call-by-value:** In this approach copy of an argument value is pass to a method. Changes made to the argument value inside the method will have no effect on the arguments.
2. **call-by-reference:** In this reference of an argument is pass to a method. Any changes made inside the method will affect the argument value.

### Example of call-by-value:

```
public class Test
{
    public void callByValue(int x)
    {
        x=100;
    }
    public static void main(String[] args)
    {
        int x=50;
        Test t = new Test();
        t.callByValue(x);    //function call
        System.out.println(x);
    }
}
```

### Output:

50

## Method overloading

If two or more method in a class have same name but different parameters, it is known as method overloading. Overloading always occur in the same class(unlike method overriding).

Method overloading is one of the ways through which java supports polymorphism. Method overloading can be done by changing number of arguments or by changing the data type of arguments. If two or more method have same name and same parameter list **but differs in return type are not** said to be overloaded method.

**Note:** Overloaded method can have different access modifiers.

## Method overloading by changing data type of Arguments

### Example:

```
class Calculate {  
    void sum(int a, int b) {  
        System.out.println("sum is" + (a + b));  
    }  
  
    void sum(float a, float b) {  
        System.out.println("sum is" + (a + b));  
    }  
  
    public static void main(String[] args) {  
        Calculate cal = new Calculate();  
        cal.sum(8, 5); // sum(int a, int b) is method is called.  
        cal.sum(4.6f, 3.8f); // sum(float a, float b) is called.  
    }  
}
```

**Output:**

Sum is 13  
Sum is 8.4

You can see that sum() method is overloaded two times. The first takes two integer arguments, the second takes two float arguments.

## Method overloading by changing number of argument

**Example:**

```
class Area {  
    void find(int l, int b) {  
        System.out.println("Area is" + (l * b));  
    }  
  
    void find(int l, int b, int h) {  
        System.out.println("Area is" + (l * b * h));  
    }  
  
    public static void main(String[] args) {  
        Area ar = new Area();  
        ar.find(8, 5); // find(int l, int b) is method is called.  
        ar.find(4, 6, 2); // find(int l, int b, int h) is called.  
    }  
}
```

```
}  
}
```

**Example:**

Area is 40

Area is 48

In this example the find() method is overloaded twice. The first takes two arguments to calculate area, and the second takes three arguments to calculate area.

When an overloaded method is called java look for match between the arguments to call the method and the method's parameters. This match need not always be exact, sometime when exact match is not found, Java automatic type conversion plays a vital role.

## Method overloading with type promotion

**Example:**

```
class Area {  
    void find(long l, long b) {  
        System.out.println("Area is " + (l * b));  
    }  
  
    void find(int l, int b, int h) {  
        System.out.println("Area is " + (l * b * h));  
    }  
  
    public static void main(String[] args) {  
        Area ar = new Area();  
        ar.find(8, 5); // automatic type conversion from find(int,int) to  
        find(long,long) .  
        ar.find(2, 4, 6); // find(int l, int b,int h) is called.  
    }  
}
```

**Output:**

Area is 40

Area is 48

# Assignment

Write a program to print out a “triangle made of numbers” and print out the “bottom right number”. If given number Of Rows == 5 , then it should look like:

```
01
02 03
04 05 06
07 08 09 10
11 12 13 14 15
bottom right number is: 15
```

## Further reading

What is a method in Java?

What is the difference between method overloading and overriding?

Function overloading (wikipedia)



