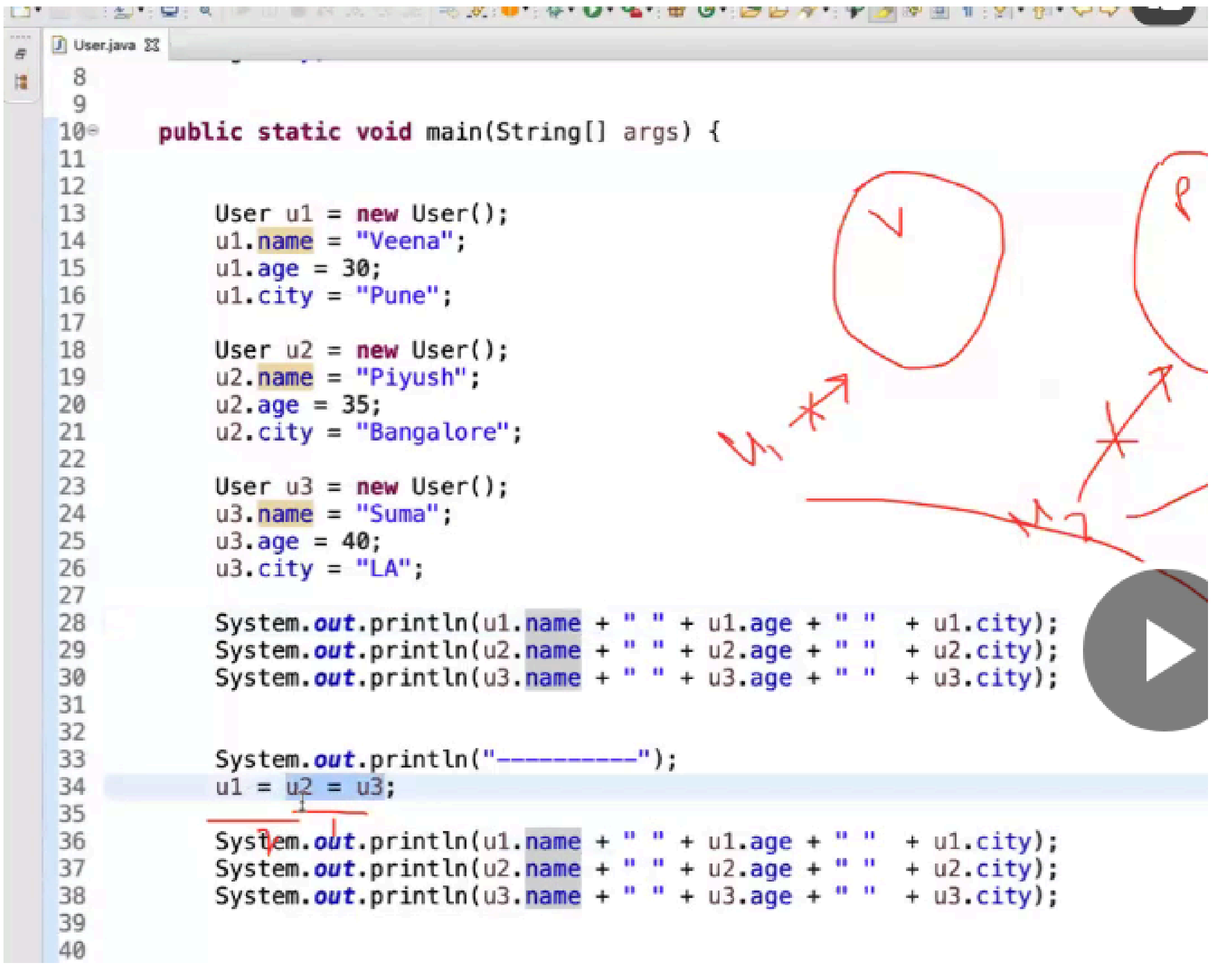
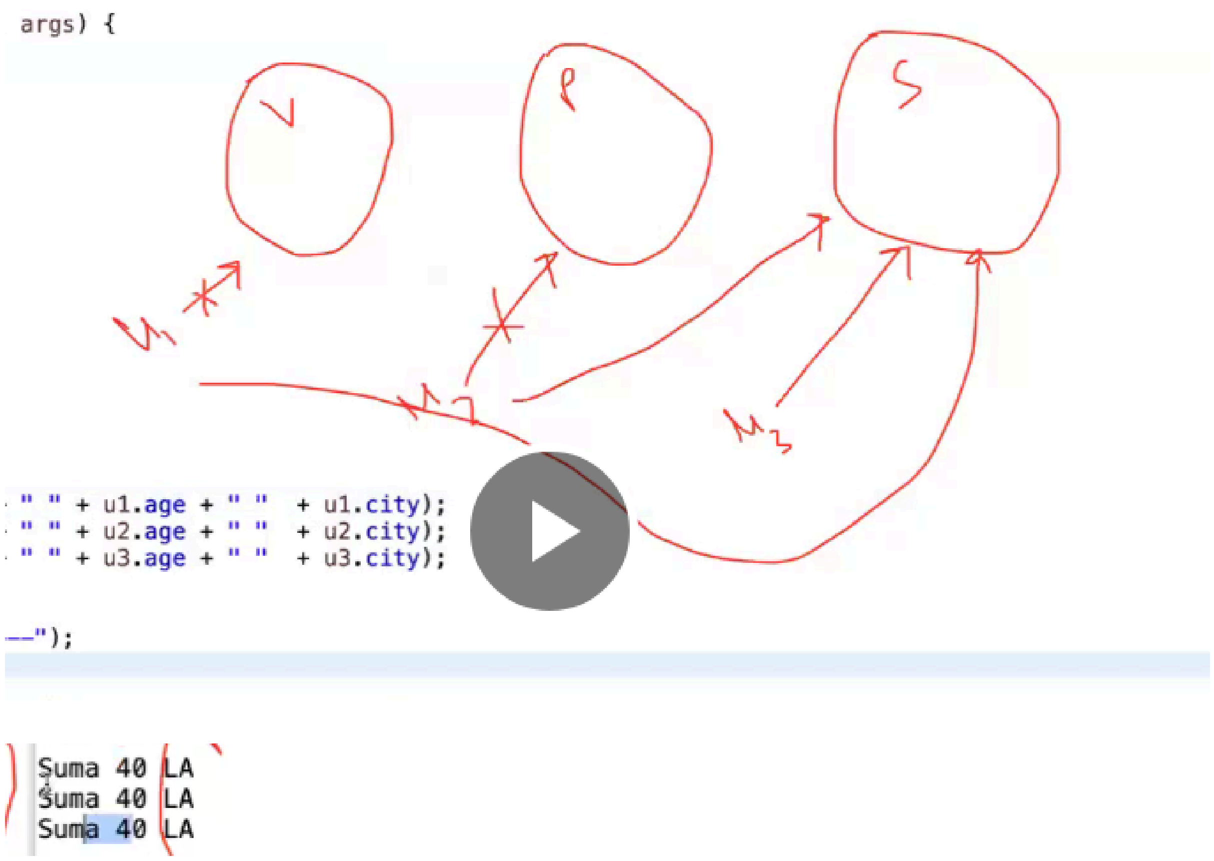


```java
	public static void main(String[] args) {


		User u1 = new User();
		u1.name = "Veena";
		u1.age = 30;
		u1.city = "Pune";

		User u2 = new User();
		u2.name = "Piyush";
		u2.age = 35;
		u2.city = "Bangalore";

		User u3 = new User();
		u3.name = "Suma";
		u3.age = 40;
		u3.city = "LA";

		System.out.println(u1.name + " " + u1.age + " "  + u1.city);
		System.out.println(u2.name + " " + u2.age + " "  + u2.city);
		System.out.println(u3.name + " " + u3.age + " "  + u3.city);


		System.out.println("----------");
		u1 = u2 = u3;

		System.out.println(u1.name + " " + u1.age + " "  + u1.city);
		System.out.println(u2.name + " " + u2.age + " "  + u2.city);
		System.out.println(u3.name + " " + u3.age + " "  + u3.city);
```

When we have multiple assignments, start from the right hand side.

U2=u3 first.

Then u1=u2.

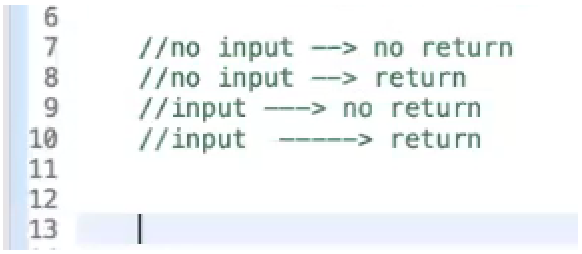


```
args) {
. " " + u1.age + " "  + u1.city);
. " " + u2.age + " "  + u2.city);
. " " + u3.age + " "  + u3.city);

--");
```

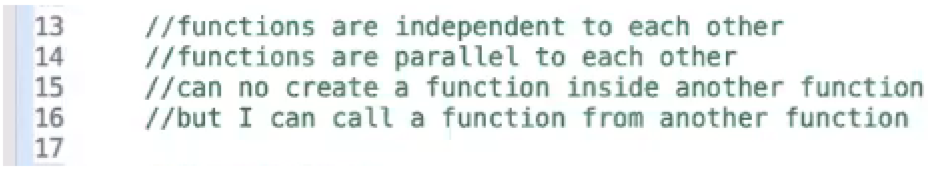
```
Suma 40 LA
Suma 40 LA
Suma 40 LA
```

Function also known as method in java.

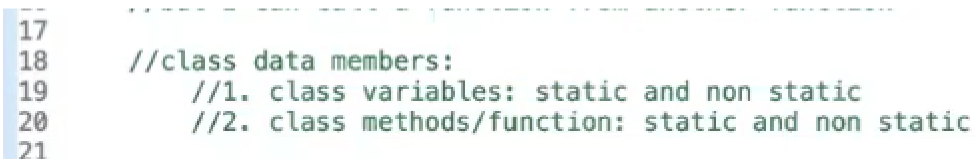
Reusability.



```
    //no input --> no return
    //no input --> return
    //input ---> no return
    //input  -----> return
```



```
    //functions are independent to each other
    //functions are parallel to each other
    //can no create a function inside another function
    //but I can call a function from another function
```

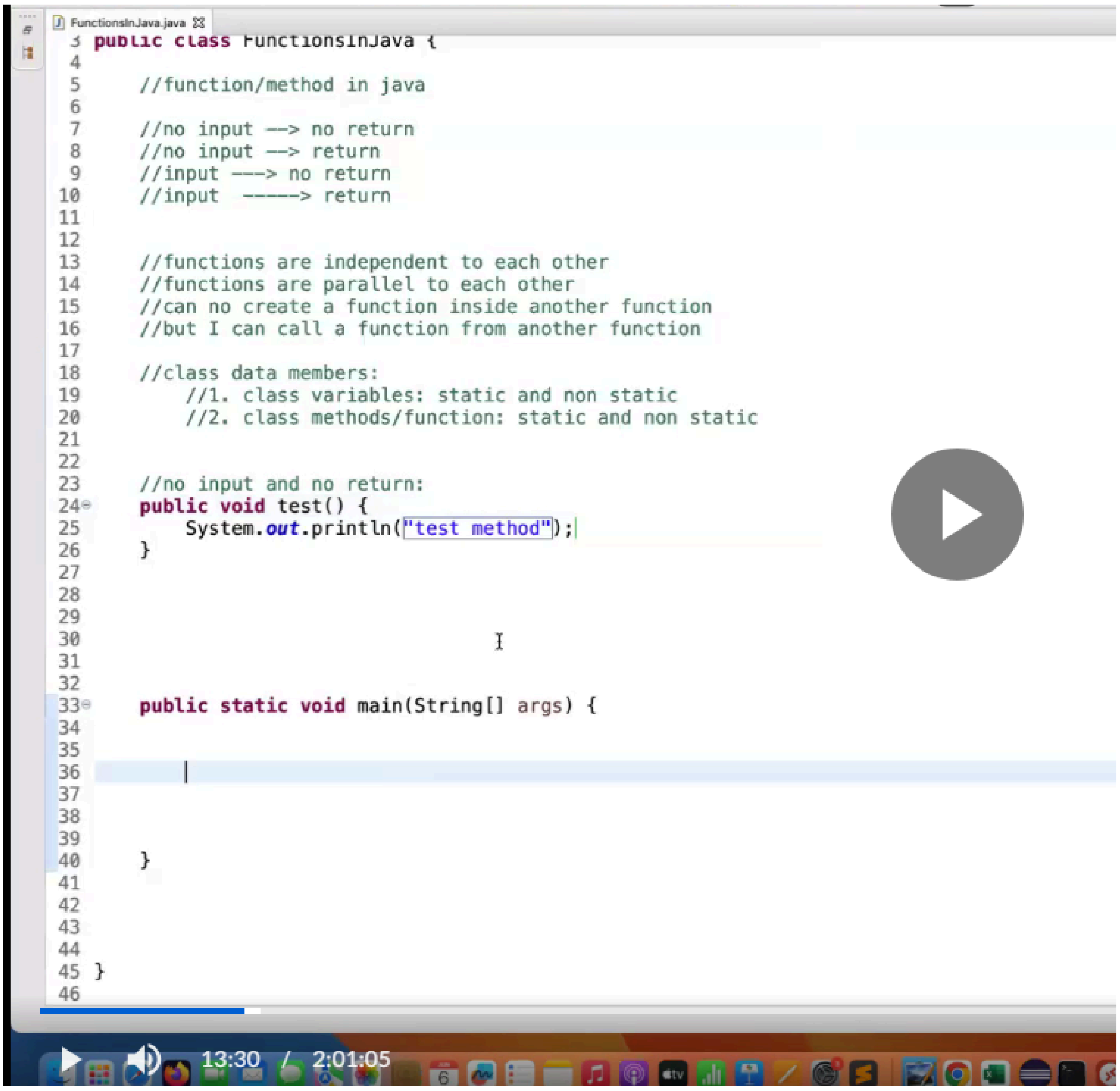


```
    //class data members:
        //1. class variables: static and non static
        //2. class methods/function: static and non static
```

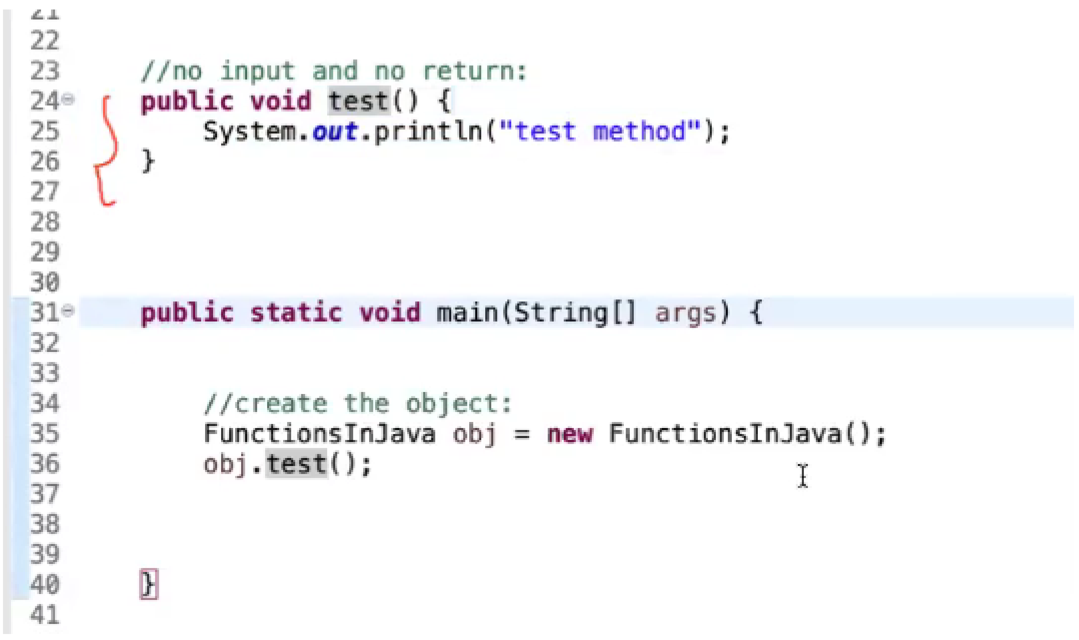
Note-

Public void test();

This is not allowed for functions. There should be open and closing braces.

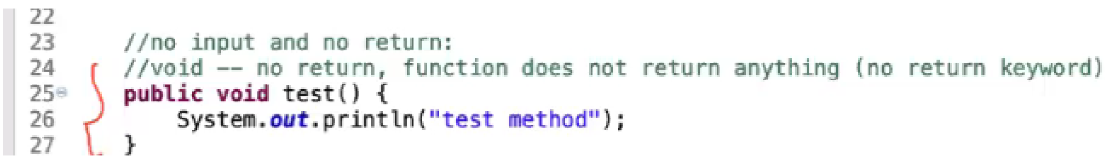


```
public class FunctionsInJava {

    //function/method in java

    //no input --> no return
    //no input --> return
    //input ---> no return
    //input  -----> return


    //functions are independent to each other
    //functions are parallel to each other
    //can no create a function inside another function
    //but I can call a function from another function

    //class data members:
        //1. class variables: static and non static
        //2. class methods/function: static and non static


    //no input and no return:
    public void test() {
        System.out.println("test method");
    }






    public static void main(String[] args) {







    }




}
```

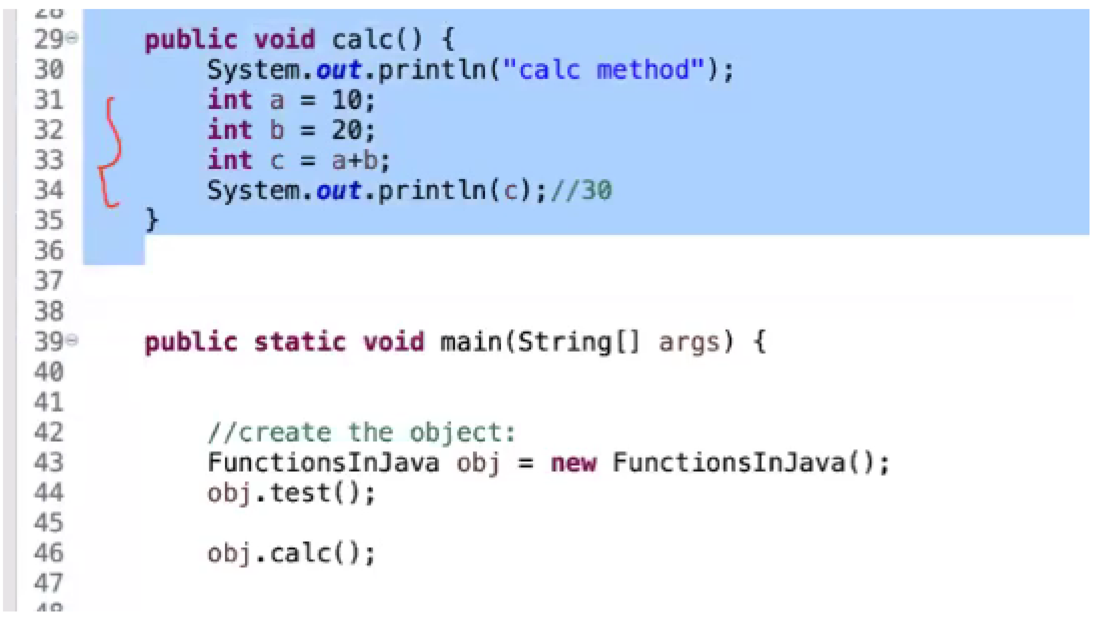


```
    //no input and no return:
    public void test() {
        System.out.println("test method");
    }



    public static void main(String[] args) {


        //create the object:
        FunctionsInJava obj = new FunctionsInJava();
        obj.test();



    }
```

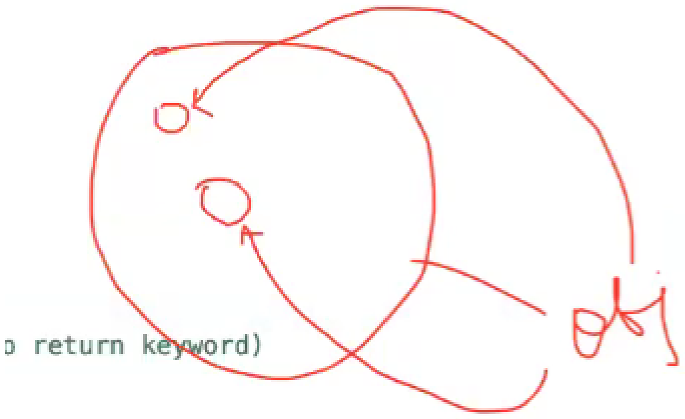
Test method.

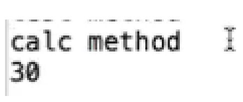


```
    //no input and no return:
    //void -- no return, function does not return anything (no return keyword)
    public void test() {
        System.out.println("test method");
    }
```

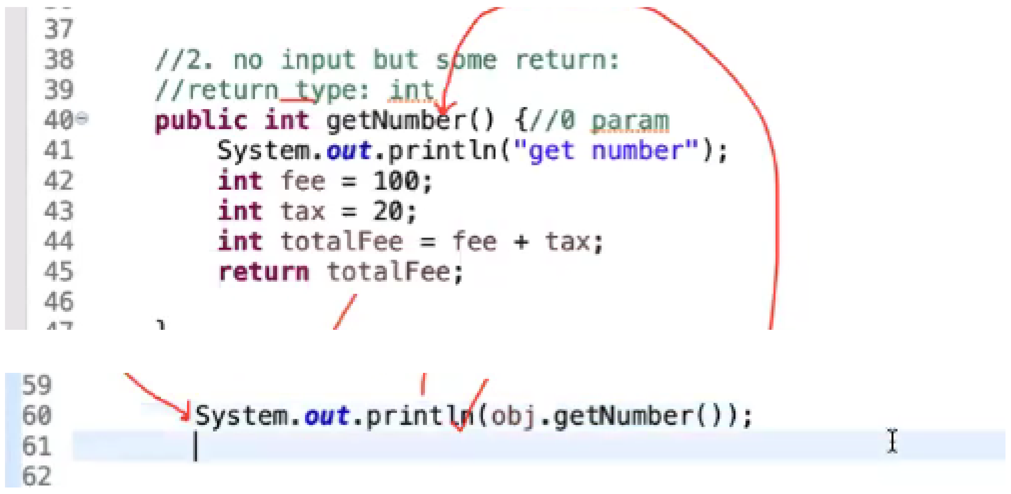


```
    public void calc() {
        System.out.println("calc method");
        int a = 10;
        int b = 20;
        int c = a+b;
        System.out.println(c);//30
    }



    public static void main(String[] args) {


        //create the object:
        FunctionsInJava obj = new FunctionsInJava();
        obj.test();

        obj.calc();

```



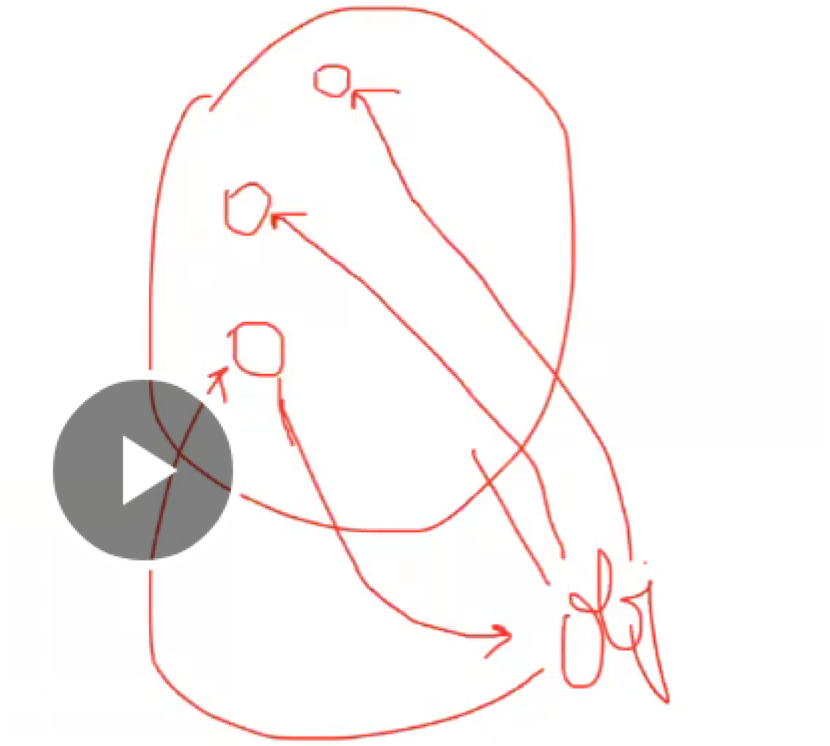


```
calc method
30
```



```
    //2. no input but some return:
    //return_type: int
    public int getNumber() {//0 param
        System.out.println("get number");
        int fee = 100;
        int tax = 20;
        int totalFee = fee + tax;
        return totalFee;

    }
```

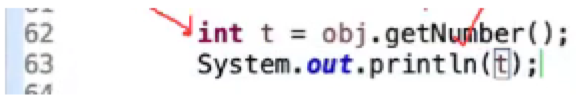
```
        System.out.println(obj.getNumber());
```



120

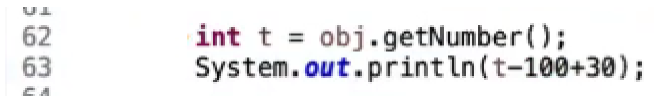
Not good practice to use the print the values directly.

Good practice to store in variables-



```
int t = obj.getNumber();
System.out.println(t);
```

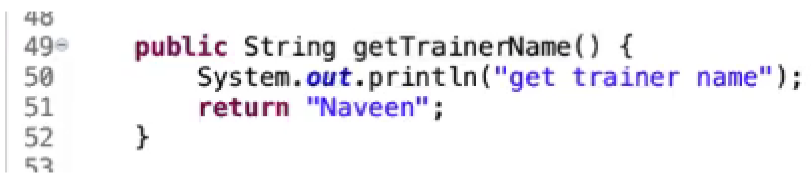
Useful for manipulating –



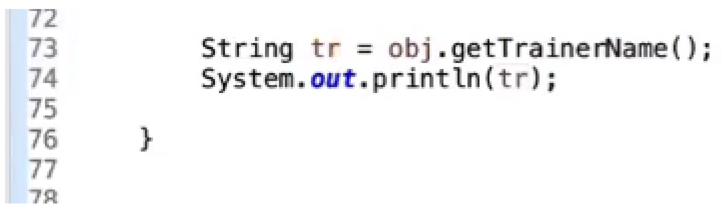
```
int t = obj.getNumber();
System.out.println(t-100+30);
```

Note-

Function variables are local variables.

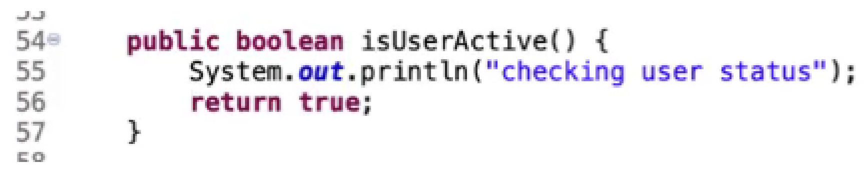


```
    public String getTrainerName() {
        System.out.println("get trainer name");
        return "Naveen";
    }
```

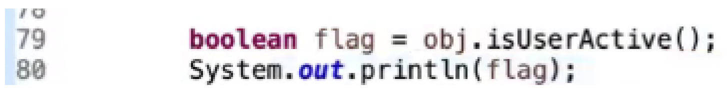


```
        String tr = obj.getTrainerName();
        System.out.println(tr);

    }
```

Naveen



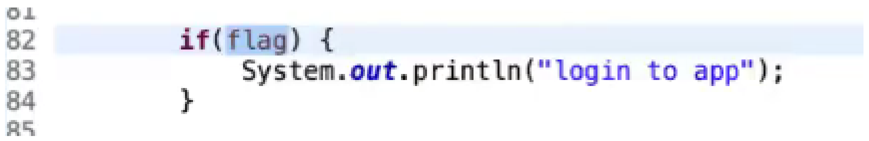
```
    public boolean isUserActive() {
        System.out.println("checking user status");
        return true;
    }
```



```
        boolean flag = obj.isUserActive();
        System.out.println(flag);
```

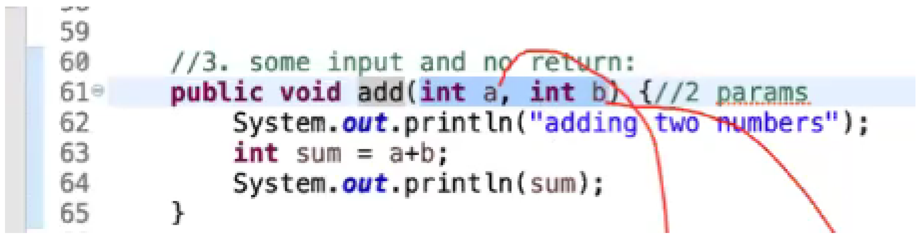
True

Use the variable for further operations-



```
        if(flag) {
            System.out.println("login to app");
        }
```

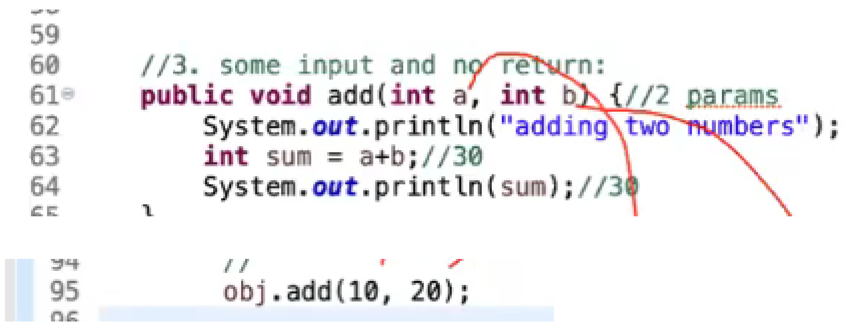
Login to app.



```
    //3. some input and no return:
    public void add(int a, int b) {//2 params
        System.out.println("adding two numbers");
        int sum = a+b;
        System.out.println(sum);
    }
```

Here a and b are known as parameters.

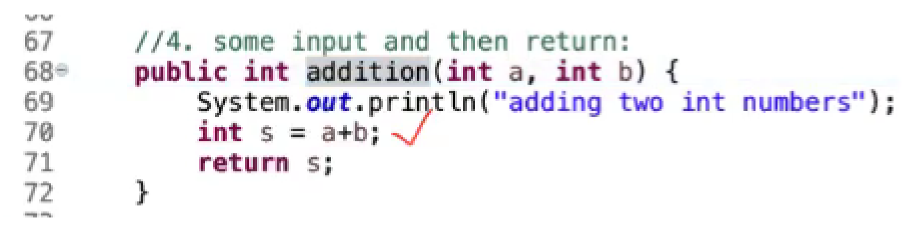
Obj.add(10,20); when we are calling the function, it is known as argument.



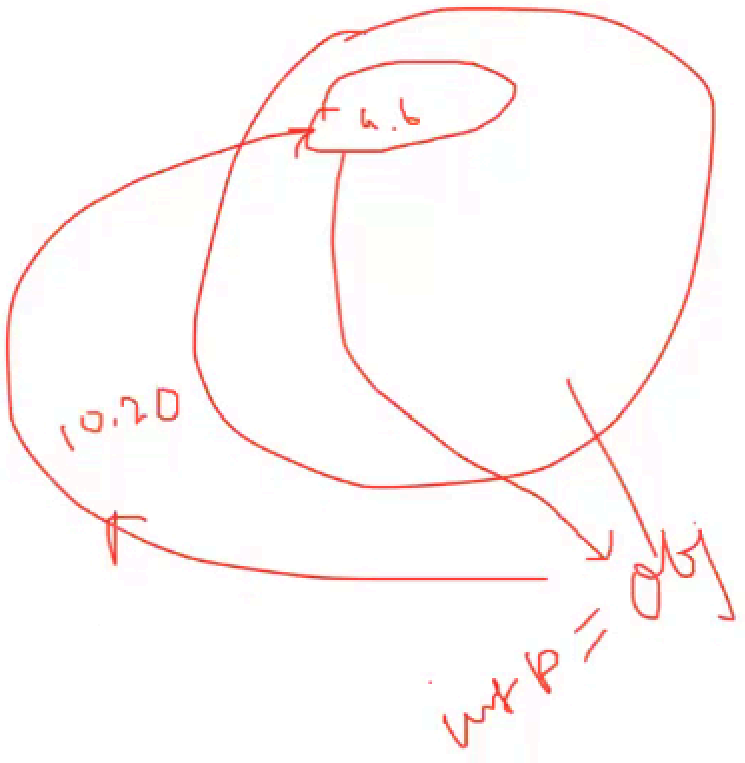
```
    //3. some input and no return:
    public void add(int a, int b) {//2 params
        System.out.println("adding two numbers");
        int sum = a+b;//30
        System.out.println(sum);//30
```

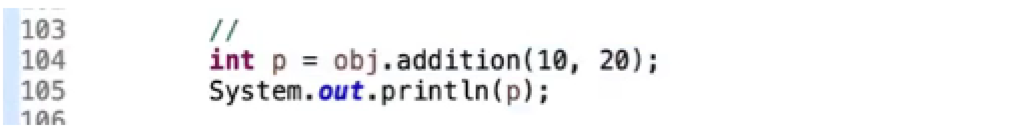
```
        obj.add(10, 20);
```

30



```
    //4. some input and then return:
    public int addition(int a, int b) {
        System.out.println("adding two int numbers");
        int s = a+b;
        return s;
    }
```

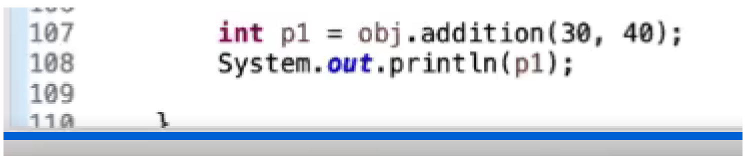




```
        //
        int p = obj.addition(10, 20);
        System.out.println(p);
```

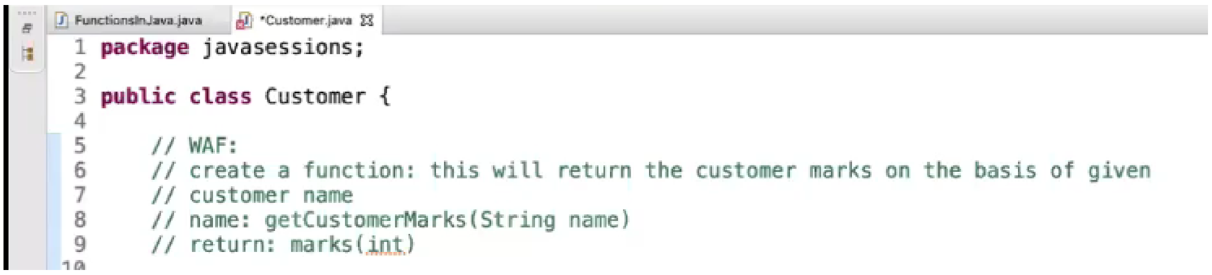
30

Call with any set of values-



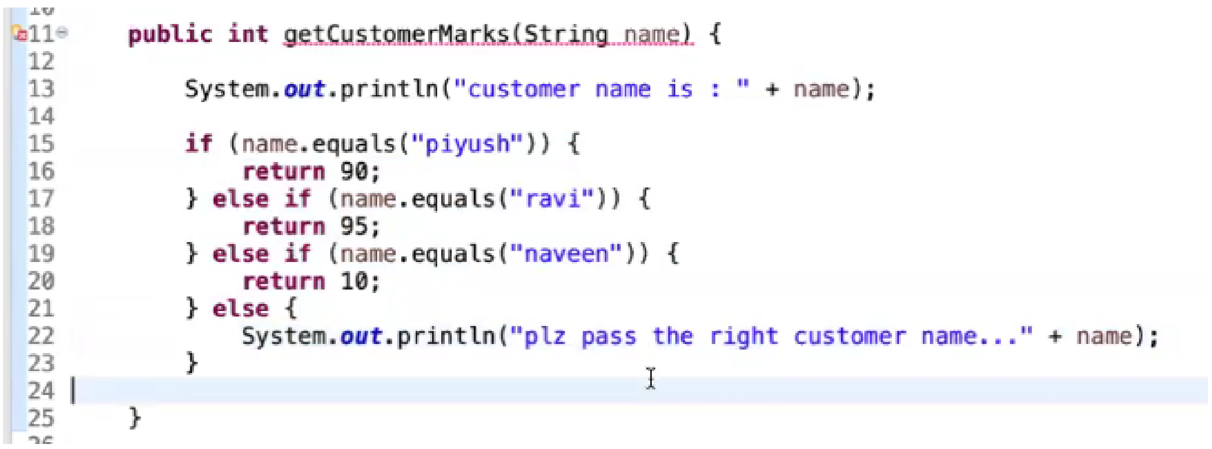
```
        int p1 = obj.addition(30, 40);
        System.out.println(p1);
    }
```

70



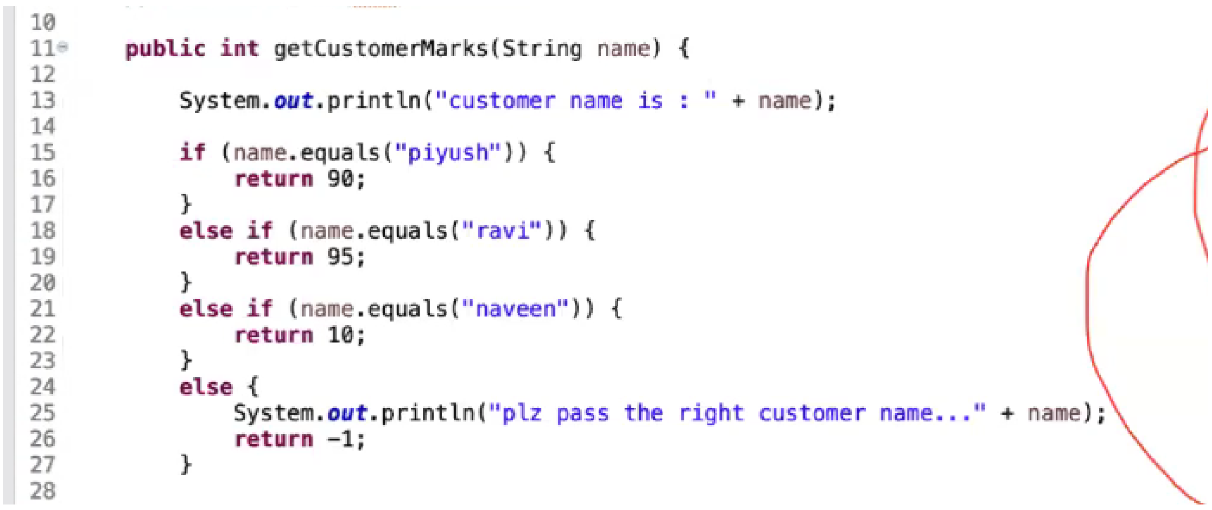
```java
package javasessions;

public class Customer {

    // WAF:
    // create a function: this will return the customer marks on the basis of given
    // customer name
    // name: getCustomerMarks(String name)
    // return: marks(int)
```

We get error. Why? Because we didn't mention what to return when no condition matches.

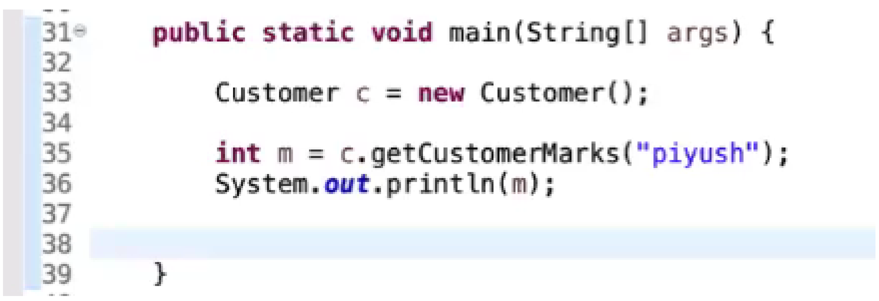


```java
    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        if (name.equals("piyush")) {
            return 90;
        } else if (name.equals("ravi")) {
            return 95;
        } else if (name.equals("naveen")) {
            return 10;
        } else {
            System.out.println("plz pass the right customer name..." + name);
        }

    }
```

Always return negative for else blocks when other scenarios fail-



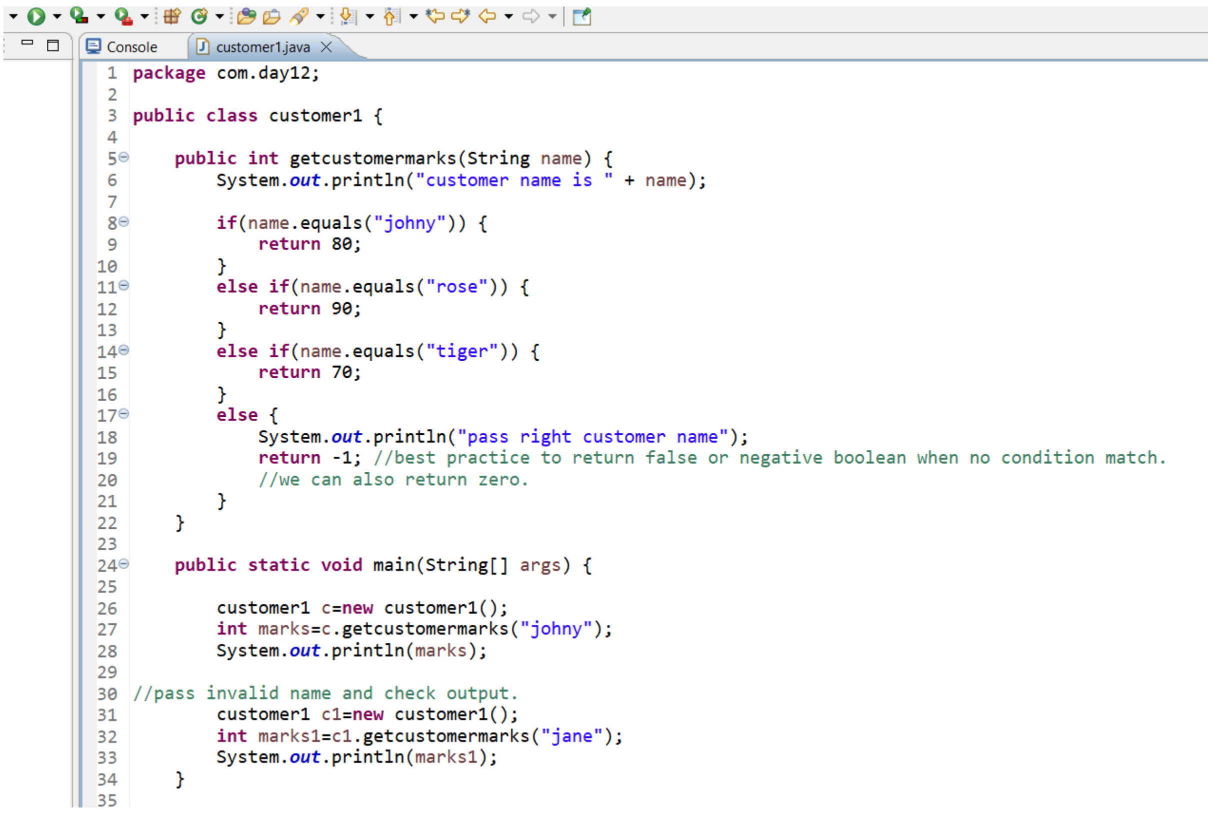
```java
    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        if (name.equals("piyush")) {
            return 90;
        }
        else if (name.equals("ravi")) {
            return 95;
        }
        else if (name.equals("naveen")) {
            return 10;
        }
        else {
            System.out.println("plz pass the right customer name..." + name);
            return -1;
        }
```



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

        Customer c = new Customer();

        int m = c.getCustomerMarks("piyush");
        System.out.println(m);


    }
```

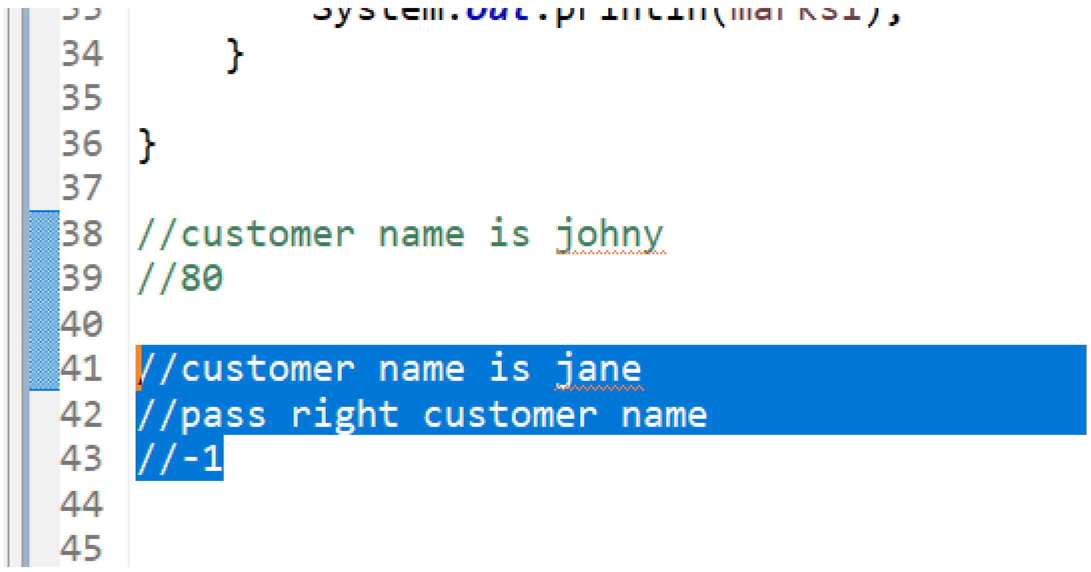
```
<terminated> Customer (10) [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse.j
customer name is : piyush
90
```

<u>paste customer1-</u>



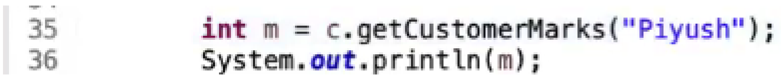
Console customer1.java

```java
package com.day12;

public class customer1 {

    public int getcustomermarks(String name) {
        System.out.println("customer name is " + name);

        if(name.equals("johny")) {
            return 80;
        }
        else if(name.equals("rose")) {
            return 90;
        }
        else if(name.equals("tiger")) {
            return 70;
        }
        else {
            System.out.println("pass right customer name");
            return -1; //best practice to return false or negative boolean when no condition match.
            //we can also return zero.
        }
    }

    public static void main(String[] args) {

        customer1 c=new customer1();
        int marks=c.getcustomermarks("johny");
        System.out.println(marks);

//pass invalid name and check output.
        customer1 c1=new customer1();
        int marks1=c1.getcustomermarks("jane");
        System.out.println(marks1);
    }
```



```java
        System.out.println(marks1);
    }

}

//customer name is johny
//80

//customer name is jane
//pass right customer name
//-1
```

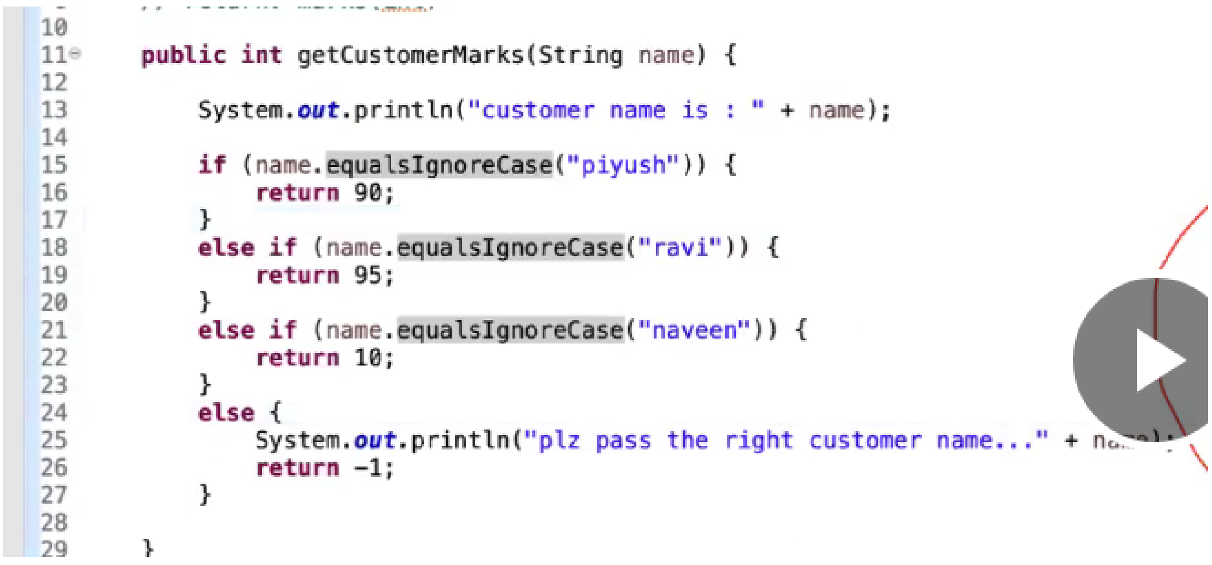
Case sensitive is there-



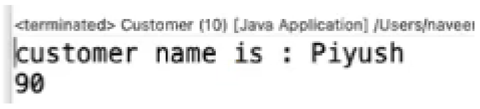
```java
        int m = c.getCustomerMarks("Piyush");
        System.out.println(m);
```

Returns -1.

Use equals ignore case to avoid above errors-

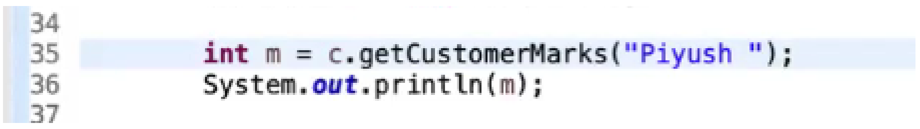


```
    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        if (name.equalsIgnoreCase("piyush")) {
            return 90;
        }
        else if (name.equalsIgnoreCase("ravi")) {
            return 95;
        }
        else if (name.equalsIgnoreCase("naveen")) {
            return 10;
        }
        else {
            System.out.println("plz pass the right customer name..." + na
            return -1;
        }

    }
```



```
<terminated> Customer (10) [Java Application] /Users/navee
customer name is : Piyush
90
```

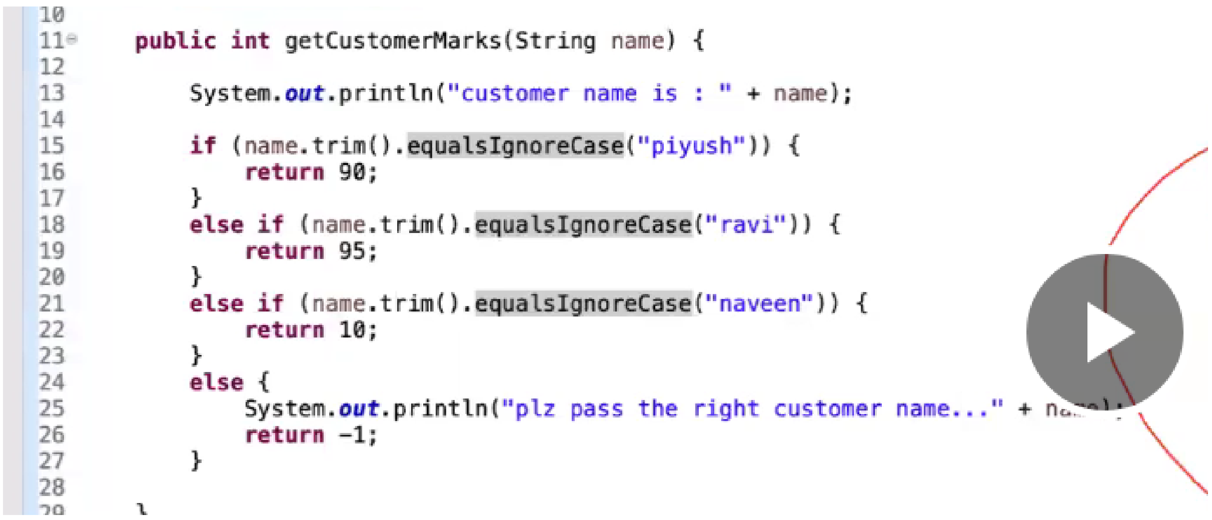
Pass space-



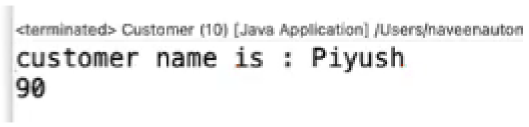
```
        int m = c.getCustomerMarks("Piyush ");
        System.out.println(m);
```

Returns -1.

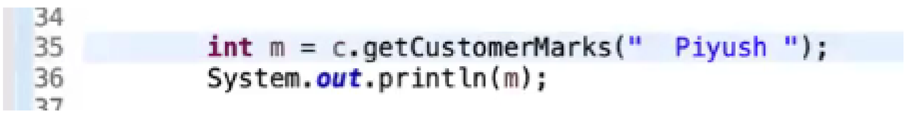
Use trim to avoid above errors-



```
    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        if (name.trim().equalsIgnoreCase("piyush")) {
            return 90;
        }
        else if (name.trim().equalsIgnoreCase("ravi")) {
            return 95;
        }
        else if (name.trim().equalsIgnoreCase("naveen")) {
            return 10;
        }
        else {
            System.out.println("plz pass the right customer name..." + name);
            return -1;
        }

    }
```

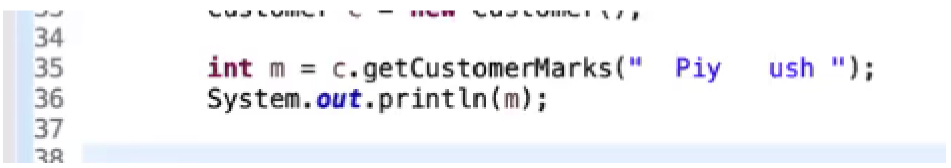


```
<terminated> Customer (10) [Java Application] /Users/naveenauton
customer name is : Piyush
90
```



```
        int m = c.getCustomerMarks("  Piyush ");
        System.out.println(m);
```

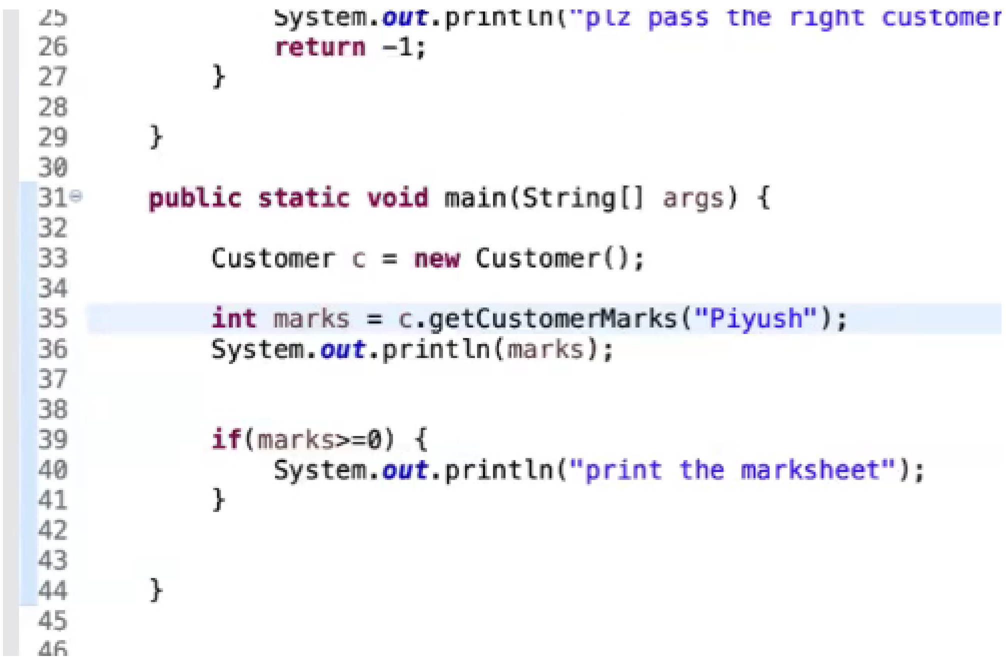
This will also return 90.



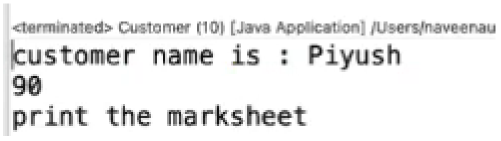
```
        Customer c = new Customer();

        int m = c.getCustomerMarks("  Piy   ush ");
        System.out.println(m);
```

This will return -1.

How to use the returned variables-

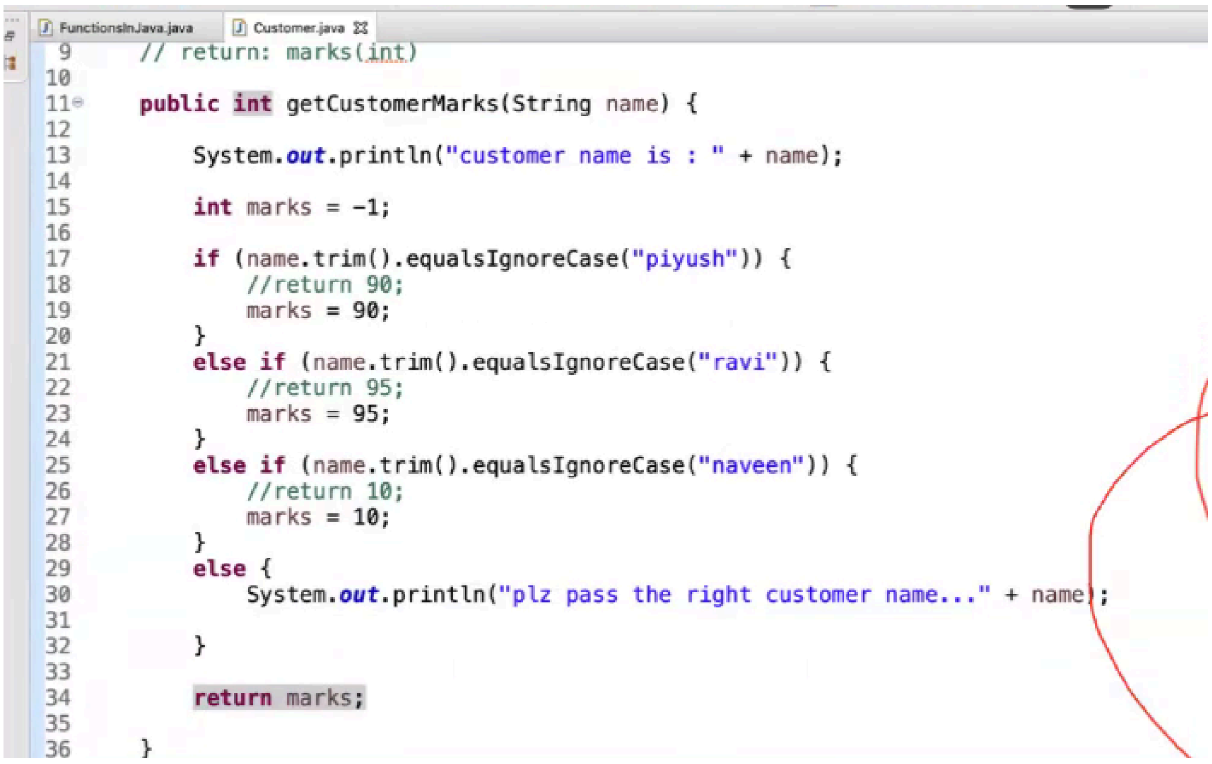


```
                System.out.println("plz pass the right customer
                return -1;
            }

        }

        public static void main(String[] args) {

            Customer c = new Customer();

            int marks = c.getCustomerMarks("Piyush");
            System.out.println(marks);


            if(marks>=0) {
                System.out.println("print the marksheet");
            }


        }
```

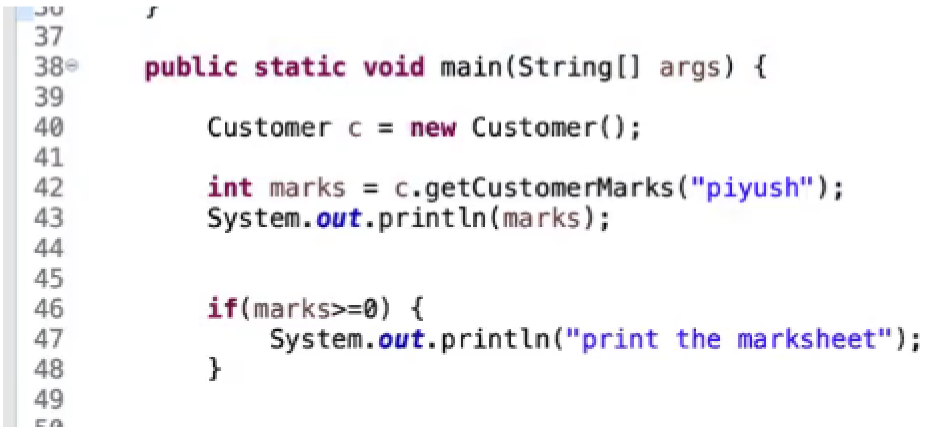


```
<terminated> Customer (10) [Java Application] /Users/naveenau
customer name is : Piyush
90
print the marksheet
```

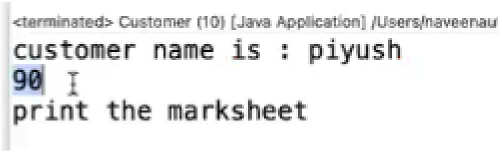
Single return statement but returning marks-



```
    // return: marks(int)

    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        int marks = -1;

        if (name.trim().equalsIgnoreCase("piyush")) {
            //return 90;
            marks = 90;
        }
        else if (name.trim().equalsIgnoreCase("ravi")) {
            //return 95;
            marks = 95;
        }
        else if (name.trim().equalsIgnoreCase("naveen")) {
            //return 10;
            marks = 10;
        }
        else {
            System.out.println("plz pass the right customer name..." + name);

        }

        return marks;

    }
```



```java
    }

    public static void main(String[] args) {

        Customer c = new Customer();

        int marks = c.getCustomerMarks("piyush");
        System.out.println(marks);


        if(marks>=0) {
            System.out.println("print the marksheet");
        }

```



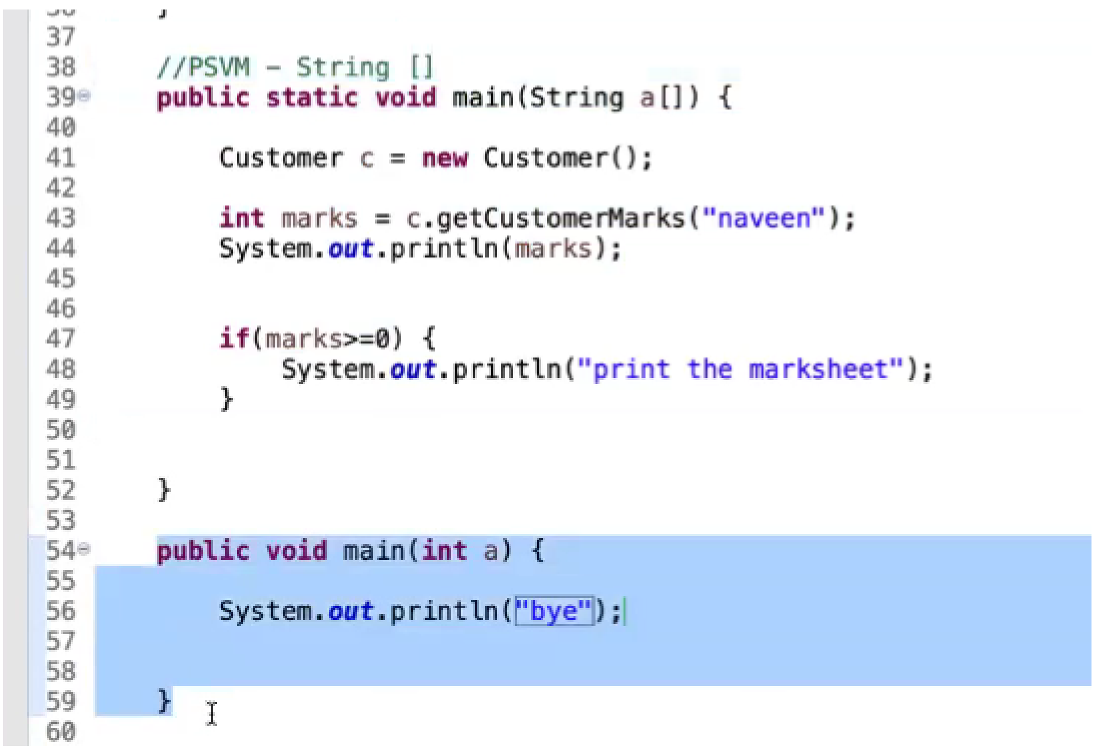
```
<terminated> Customer (10) [Java Application] /Users/naveenau
customer name is : piyush
90
print the marksheet
```

Multiple returns better performing compared to one return in end. In one return every condition is checked.

Note-

Int marks = -111111; any value can be written, its just for printing when no condition matches.

Overloaded main method-



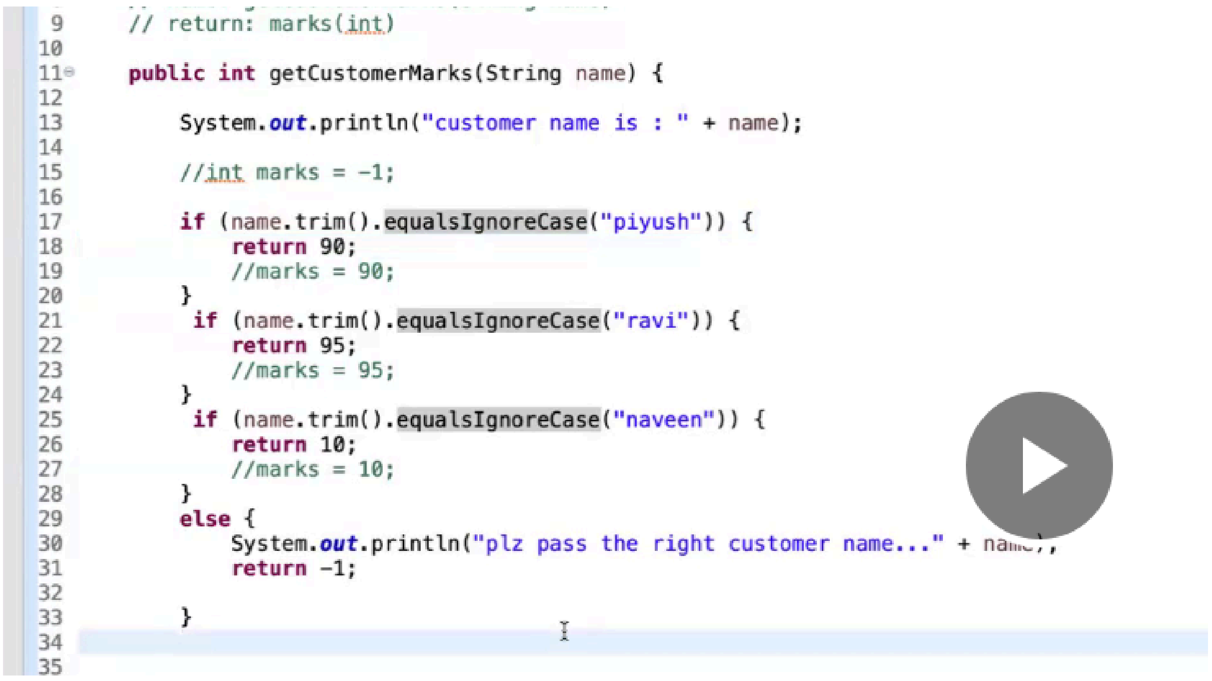
```
    }

    //PSVM - String []
    public static void main(String a[]) {

        Customer c = new Customer();

        int marks = c.getCustomerMarks("naveen");
        System.out.println(marks);


        if(marks>=0) {
            System.out.println("print the marksheet");
        }


    }

    public void main(int a) {

        System.out.println("bye");


    }
```

For java, only the psvm with arguments string array is read first.

Why main method is void –

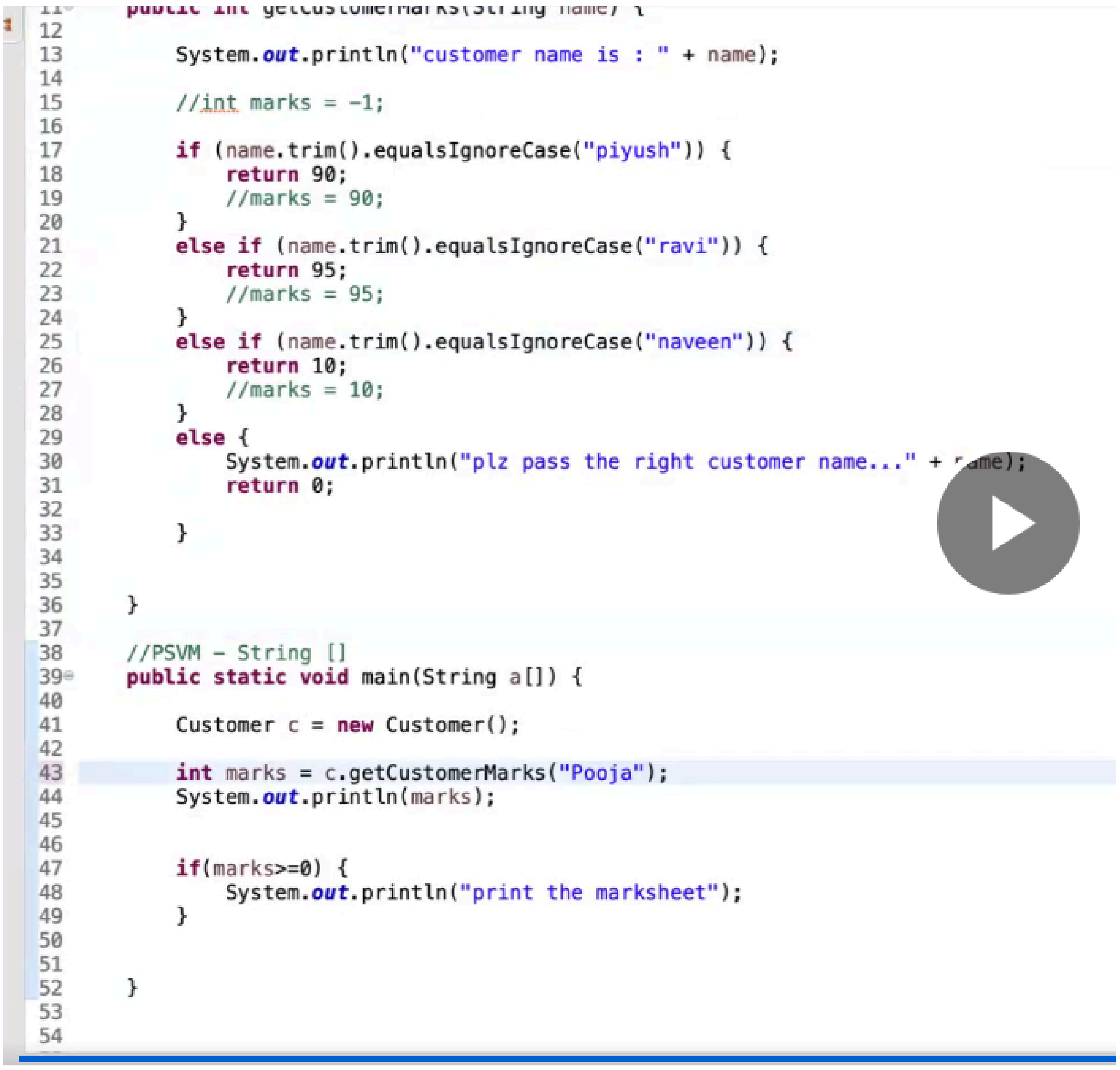
We write code to call other methods, variables, class inside main. No logic written under it(main).

Below code wont go into every if condition, thanks to return statement-



```java
    // return: marks(int)

    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        //int marks = -1;

        if (name.trim().equalsIgnoreCase("piyush")) {
            return 90;
            //marks = 90;
        }
         if (name.trim().equalsIgnoreCase("ravi")) {
            return 95;
            //marks = 95;
        }
         if (name.trim().equalsIgnoreCase("naveen")) {
            return 10;
            //marks = 10;
        }
        else {
            System.out.println("plz pass the right customer name..." + name);
            return -1;

        }

```

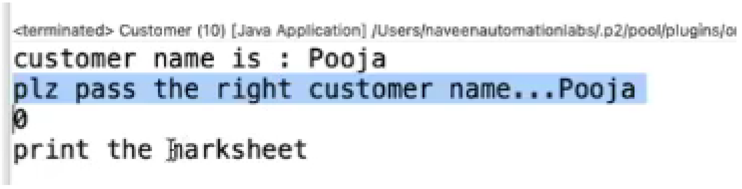
What If we write return 0 for invalid case-



```
    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        //int marks = -1;

        if (name.trim().equalsIgnoreCase("piyush")) {
            return 90;
            //marks = 90;
        }
        else if (name.trim().equalsIgnoreCase("ravi")) {
            return 95;
            //marks = 95;
        }
        else if (name.trim().equalsIgnoreCase("naveen")) {
            return 10;
            //marks = 10;
        }
        else {
            System.out.println("plz pass the right customer name..." + name);
            return 0;

        }


    }

    //PSVM - String []
    public static void main(String a[]) {

        Customer c = new Customer();

        int marks = c.getCustomerMarks("Pooja");
        System.out.println(marks);


        if(marks>=0) {
            System.out.println("print the marksheet");
        }


    }

```

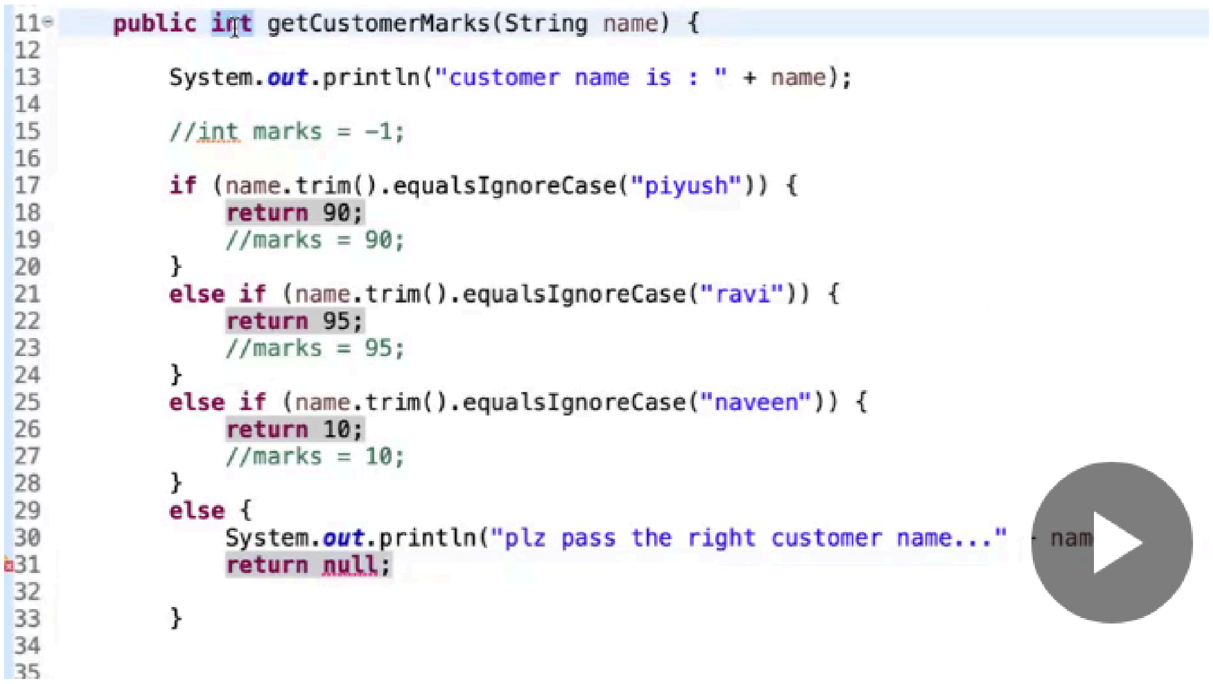
Not proper output.

Zero is valid mark.



```
<terminated> Customer (10) [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/o
customer name is : Pooja
plz pass the right customer name...Pooja
0
print the marksheet
```

Null is only for non primitive-



```
    public int getCustomerMarks(String name) {

        System.out.println("customer name is : " + name);

        //int marks = -1;

        if (name.trim().equalsIgnoreCase("piyush")) {
            return 90;
            //marks = 90;
        }
        else if (name.trim().equalsIgnoreCase("ravi")) {
            return 95;
            //marks = 95;
        }
        else if (name.trim().equalsIgnoreCase("naveen")) {
            return 10;
            //marks = 10;
        }
        else {
            System.out.println("plz pass the right customer name..." + nam
            return null;

        }
```

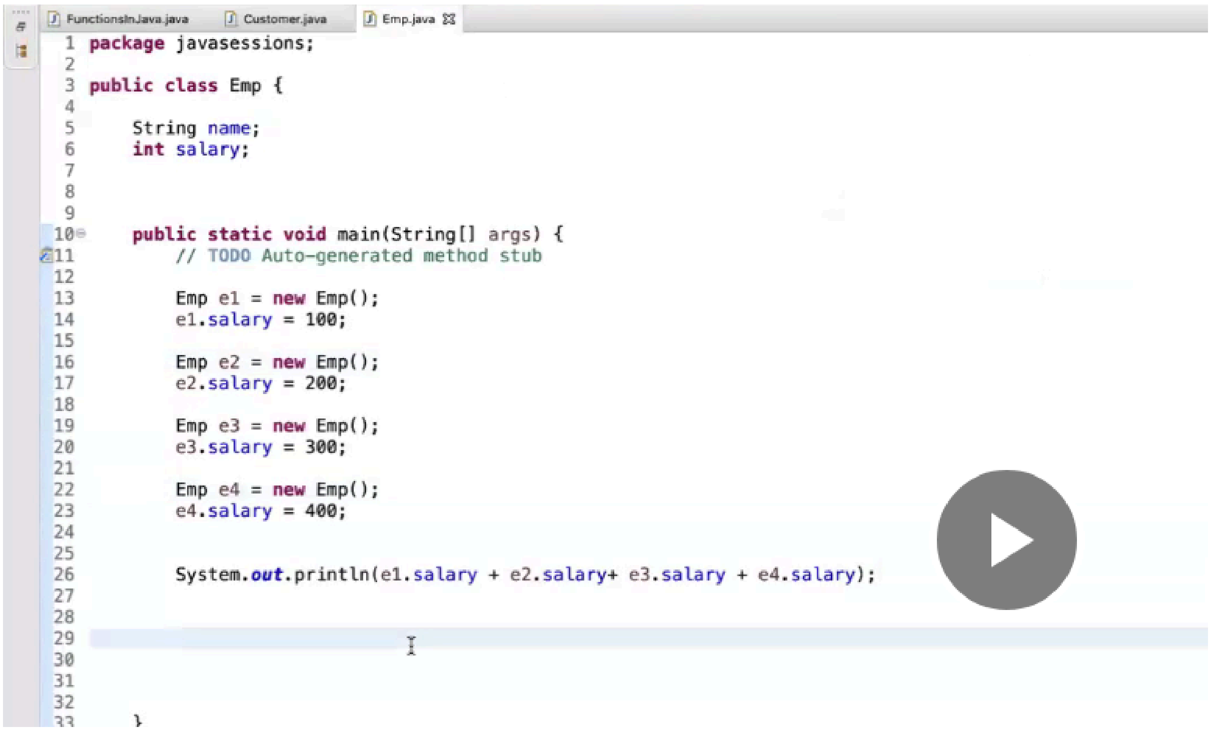
Don’t use null as return for primitive data types.

//Type mismatch: cannot convert from null to int

Note-

One method cannot have more than 255 parameters.

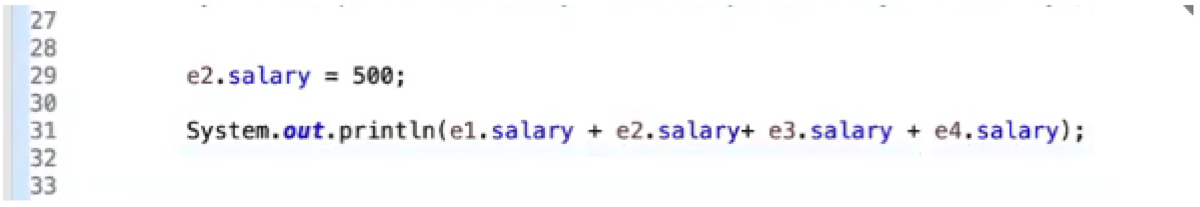
Multiple objects for one class-



```
package javasessions;

public class Emp {

    String name;
    int salary;



    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Emp e1 = new Emp();
        e1.salary = 100;

        Emp e2 = new Emp();
        e2.salary = 200;

        Emp e3 = new Emp();
        e3.salary = 300;

        Emp e4 = new Emp();
        e4.salary = 400;


        System.out.println(e1.salary + e2.salary+ e3.salary + e4.salary);







    }
```

1000.

Update salary and print total-



```
        e2.salary = 500;

        System.out.println(e1.salary + e2.salary+ e3.salary + e4.salary);
```

1300.