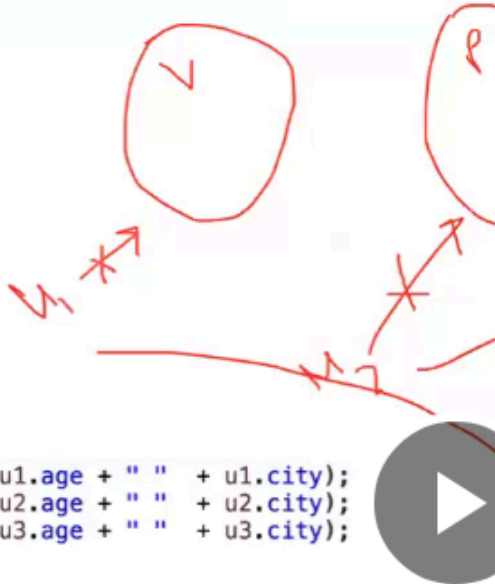


```
8
9
10 public static void main(String[] args) {
11
12
13     User u1 = new User();
14     u1.name = "Veena";
15     u1.age = 30;
16     u1.city = "Pune";
17
18     User u2 = new User();
19     u2.name = "Piyush";
20     u2.age = 35;
21     u2.city = "Bangalore";
22
23     User u3 = new User();
24     u3.name = "Suma";
25     u3.age = 40;
26     u3.city = "LA";
27
28     System.out.println(u1.name + " " + u1.age + " " + u1.city);
29     System.out.println(u2.name + " " + u2.age + " " + u2.city);
30     System.out.println(u3.name + " " + u3.age + " " + u3.city);
31
32
33     System.out.println("-----");
34     u1 = u2 = u3;
35
36     System.out.println(u1.name + " " + u1.age + " " + u1.city);
37     System.out.println(u2.name + " " + u2.age + " " + u2.city);
38     System.out.println(u3.name + " " + u3.age + " " + u3.city);
39
40 }
```

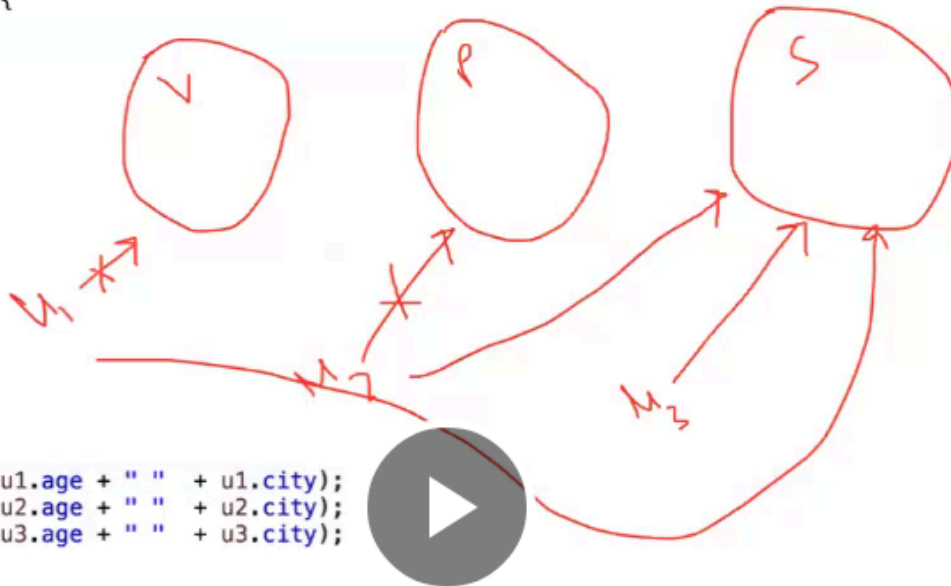


When we have multiple assignments, start from the right hand side.

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.

```
6
7 //no input —> no return
8 //no input —> return
9 //input —> no return
10 //input ———> return
11
12
13 |
```

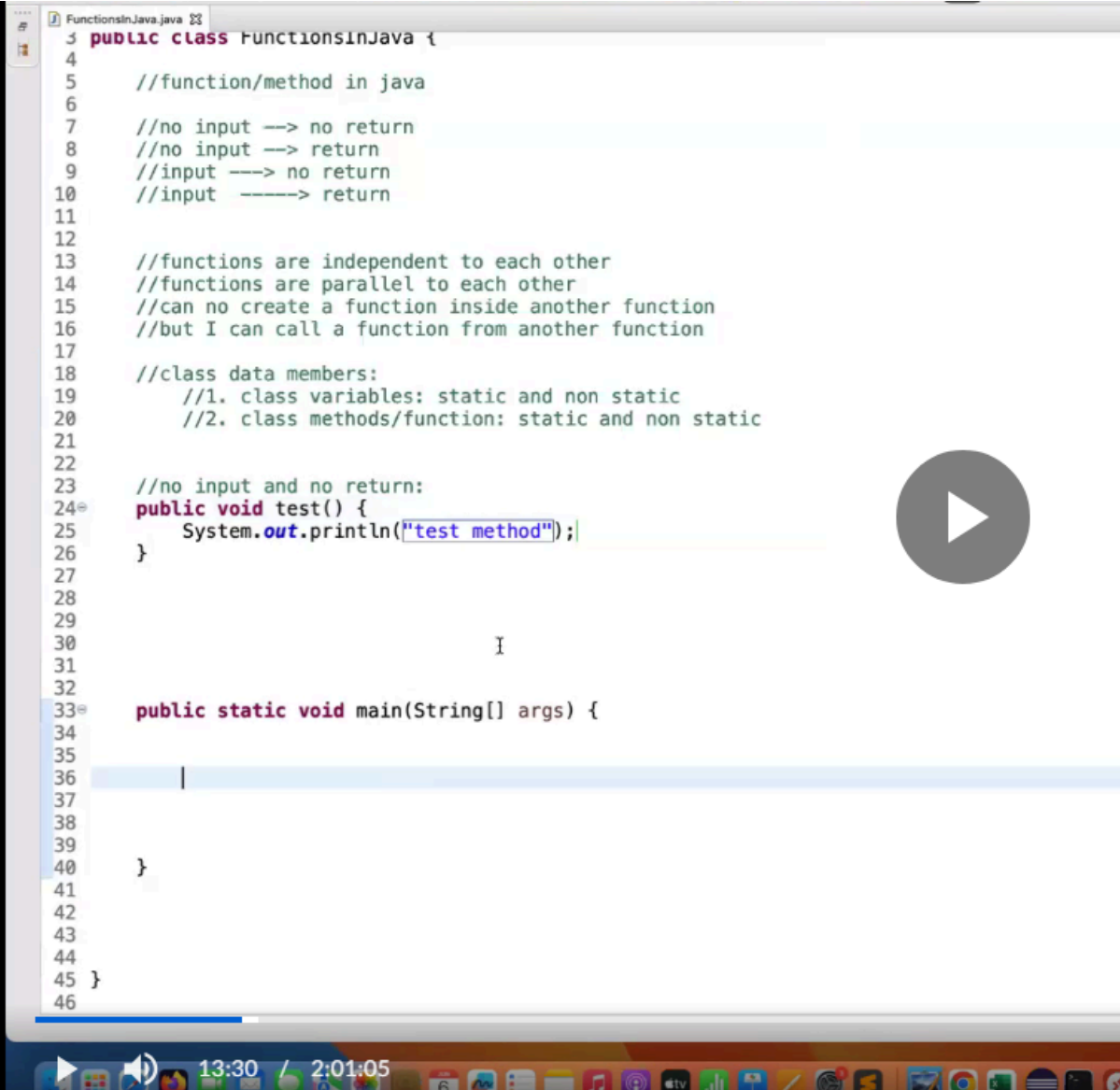
```
13 //functions are independent to each other
14 //functions are parallel to each other
15 //can no create a function inside another function
16 //but I can call a function from another function
17
```

```
17
18 //class data members:
19 //1. class variables: static and non static
20 //2. class methods/function: static and non static
21
```

Note-

Public void test();

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



```
1 public class FunctionsInJava {
2
3     //function/method in java
4
5     //no input --> no return
6     //no input --> return
7     //input ----> no return
8     //input -----> return
9
10
11
12
13     //functions are independent to each other
14     //functions are parallel to each other
15     //can no create a function inside another function
16     //but I can call a function from another function
17
18     //class data members:
19     //1. class variables: static and non static
20     //2. class methods/function: static and non static
21
22
23     //no input and no return:
24     public void test() {
25         System.out.println("test method");
26     }
27
28
29
30
31
32
33     public static void main(String[] args) {
34
35
36
37
38
39
40
41
42
43
44
45 }
46
```

```

21
22
23 //no input and no return:
24 public void test() {
25     System.out.println("test method");
26 }
27
28
29
30
31 public static void main(String[] args) {
32
33
34     //create the object:
35     FunctionsInJava obj = new FunctionsInJava();
36     obj.test();
37
38
39
40 }
41

```

## Test method.

```

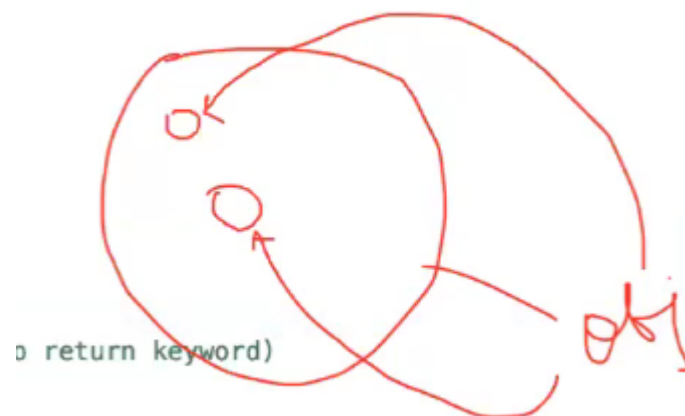
22
23 //no input and no return:
24 //void -- no return, function does not return anything (no return keyword)
25 public void test() {
26     System.out.println("test method");
27 }

```

```

28
29 public void calc() {
30     System.out.println("calc method");
31     int a = 10;
32     int b = 20;
33     int c = a+b;
34     System.out.println(c); //30
35 }
36
37
38
39 public static void main(String[] args) {
40
41
42     //create the object:
43     FunctionsInJava obj = new FunctionsInJava();
44     obj.test();
45
46     obj.calc();
47
48

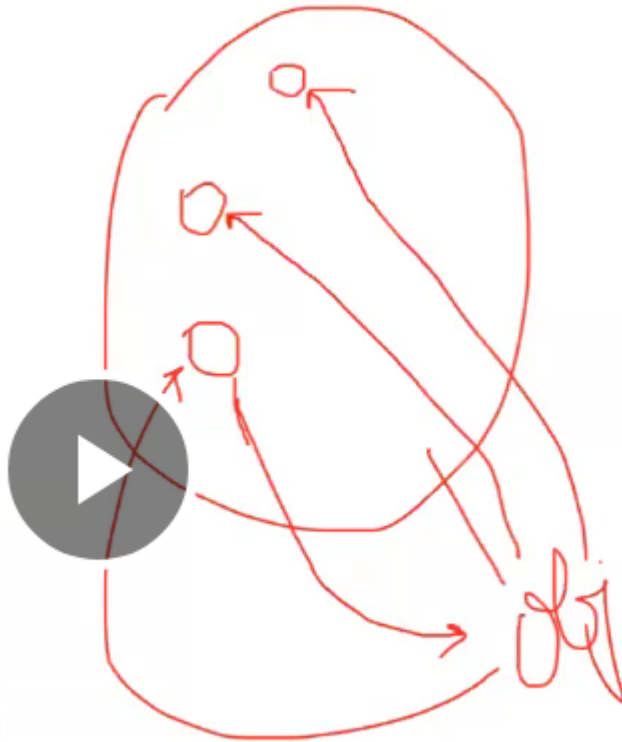
```



```
calc method  I
30
```

```
37
38 //2. no input but some return:
39 //return type: int
40 public int getNumber() { //0 param
41     System.out.println("get number");
42     int fee = 100;
43     int tax = 20;
44     int totalFee = fee + tax;
45     return totalFee;
46 }
```

```
59
60 System.out.println(obj.getNumber());
61 |
62
```



120

Not good practice to use the print the values directly.

Good practice to store in variables-

```

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

```

Useful for manipulating –

```

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

```

Note-

Function variables are local variables.

```

48
49 public String getTrainerName() {
50     System.out.println("get trainer name");
51     return "Naveen";
52 }
53

72
73     String tr = obj.getTrainerName();
74     System.out.println(tr);
75
76 }
77
78

```

Naveen

```

53
54 public boolean isUserActive() {
55     System.out.println("checking user status");
56     return true;
57 }
58

78
79     boolean flag = obj.isUserActive();
80     System.out.println(flag);

```

True

Use the variable for further operations-

```

81
82     if(flag) {
83         System.out.println("login to app");
84     }
85

```

Login to app.

```

59
60 //3. some input and no return:
61 public void add(int a, int b) { //2 params
62     System.out.println("adding two numbers");
63     int sum = a+b;
64     System.out.println(sum);
65 }

```

Here a and b are known as parameters.

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

```

59
60 //3. some input and no return:
61 public void add(int a, int b) { //2 params
62     System.out.println("adding two numbers");
63     int sum = a+b; //30
64     System.out.println(sum); //30
65 }

```

```

94 //
95 obj.add(10, 20);
96

```

30

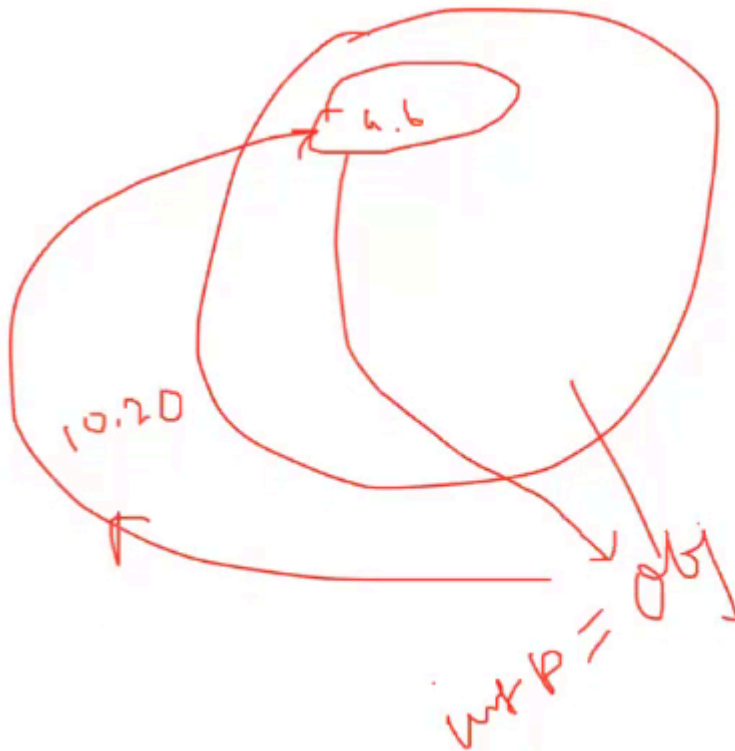




```

67 //4. some input and then return:
68 public int addition(int a, int b) {
69     System.out.println("adding two int numbers");
70     int s = a+b; ✓
71     return s;
72 }

```



```

103 //
104 int p = obj.addition(10, 20);
105 System.out.println(p);
106

```

30

Call with any set of values-

```

107 int p1 = obj.addition(30, 40);
108 System.out.println(p1);
109
110 }

```

70

```

1 package javasessions;
2
3 public class Customer {
4
5     // WAF:
6     // create a function: this will return the customer marks on the basis of given
7     // customer name
8     // name: getCustomerMarks(String name)
9     // return: marks(int)
10

```

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

```

11 public int getCustomerMarks(String name) {
12
13     System.out.println("customer name is : " + name);
14
15     if (name.equals("piyush")) {
16         return 90;
17     } else if (name.equals("ravi")) {
18         return 95;
19     } else if (name.equals("naveen")) {
20         return 10;
21     } else {
22         System.out.println("plz pass the right customer name..." + name);
23     }
24 }
25


```

Always return negative for else blocks when other scenarios fail-

```

10
11 public int getCustomerMarks(String name) {
12
13     System.out.println("customer name is : " + name);
14
15     if (name.equals("piyush")) {
16         return 90;
17     }
18     else if (name.equals("ravi")) {
19         return 95;
20     }
21     else if (name.equals("naveen")) {
22         return 10;
23     }
24     else {
25         System.out.println("plz pass the right customer name..." + name);
26         return -1;
27     }
28

```



```
31 public static void main(String[] args) {  
32  
33     Customer c = new Customer();  
34  
35     int m = c.getCustomerMarks("piyush");  
36     System.out.println(m);  
37  
38  
39 }
```

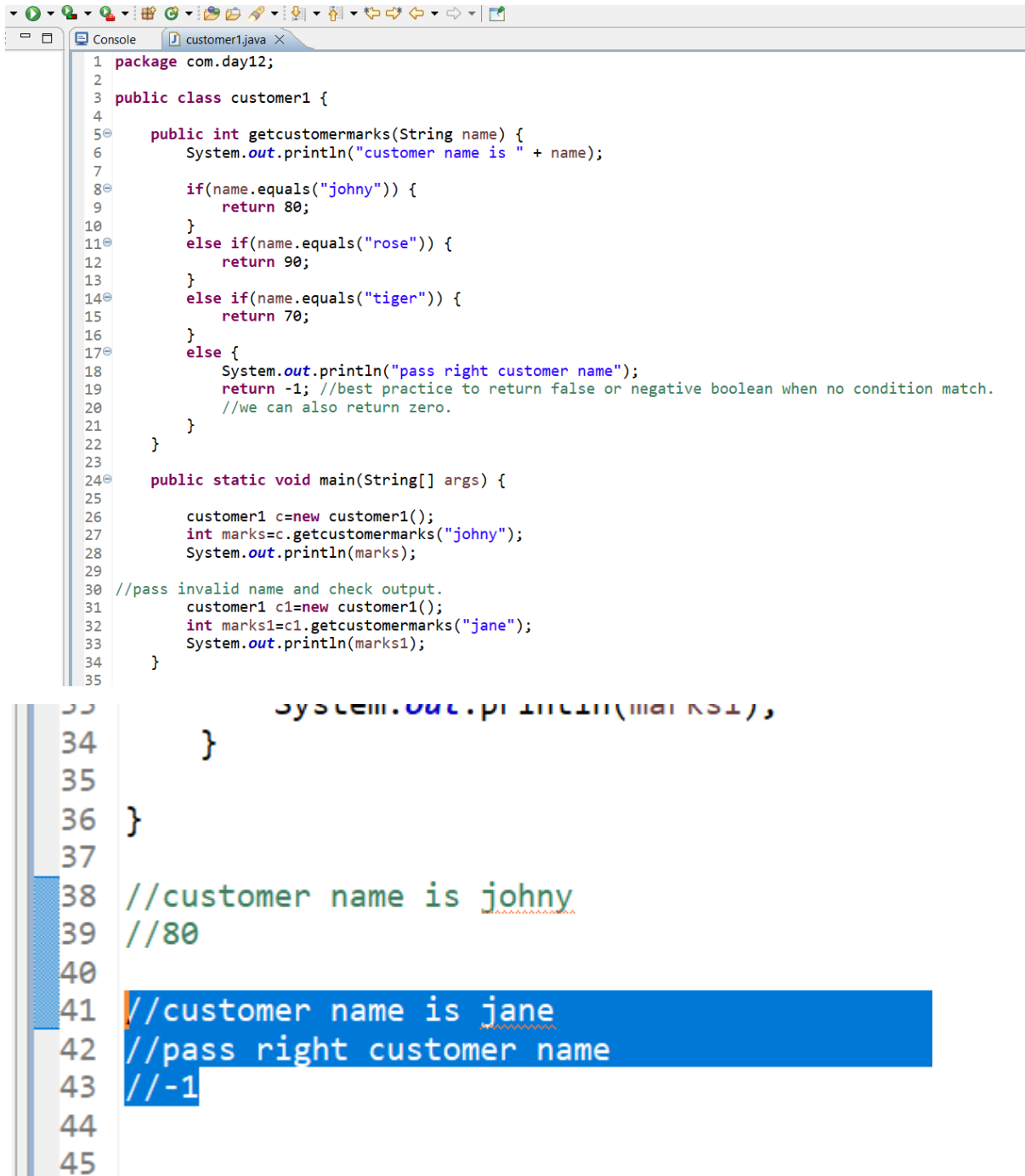
<terminated> Customer (10) [Java Application] /Users/naveenautomationlabs/p2/pool/plugins/org.eclipse.ji  
customer name is : piyush  
90

in



ie.

paste customer1-



```

1 package com.day12;
2
3 public class customer1 {
4
5     public int getcustomermarks(String name) {
6         System.out.println("customer name is " + name);
7
8         if(name.equals("johny")) {
9             return 80;
10        }
11        else if(name.equals("rose")) {
12            return 90;
13        }
14        else if(name.equals("tiger")) {
15            return 70;
16        }
17        else {
18            System.out.println("pass right customer name");
19            return -1; //best practice to return false or negative boolean when no condition match.
20            //we can also return zero.
21        }
22    }
23
24    public static void main(String[] args) {
25
26        customer1 c=new customer1();
27        int marks=c.getcustomermarks("johny");
28        System.out.println(marks);
29
30        //pass invalid name and check output.
31        customer1 c1=new customer1();
32        int marks1=c1.getcustomermarks("jane");
33        System.out.println(marks1);
34    }
35
36 }

```

```

35
36
37
38 //customer name is johny
39 //80
40
41 //customer name is jane
42 //pass right customer name
43 //-1
44
45

```

Case sensitive is there-

```

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

```

Returns -1.

Use equals ignore case to avoid above errors-

```

10
11 public int getCustomerMarks(String name) {
12
13     System.out.println("customer name is : " + name);
14
15     if (name.equalsIgnoreCase("piyush")) {
16         return 90;
17     }
18     else if (name.equalsIgnoreCase("ravi")) {
19         return 95;
20     }
21     else if (name.equalsIgnoreCase("naveen")) {
22         return 10;
23     }
24     else {
25         System.out.println("plz pass the right customer name..." + name);
26         return -1;
27     }
28 }
29

```

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

Pass space-

```

34
35 int m = c.getCustomerMarks("Piyush ");
36 System.out.println(m);
37

```


Returns -1.

Use trim to avoid above errors-

```

10
11 public int getCustomerMarks(String name) {
12     System.out.println("customer name is : " + name);
13
14     if (name.trim().equalsIgnoreCase("piyush")) {
15         return 90;
16     }
17     else if (name.trim().equalsIgnoreCase("ravi")) {
18         return 95;
19     }
20     else if (name.trim().equalsIgnoreCase("naveen")) {
21         return 10;
22     }
23     else {
24         System.out.println("plz pass the right customer name..." + name);
25         return -1;
26     }
27 }
28
29

```



```

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

```

```

34
35 int m = c.getCustomerMarks(" Piyush ");
36 System.out.println(m);
37
38

```

This will also return 90.

```

33 customer c = new Customer();
34
35 int m = c.getCustomerMarks(" Piy ush ");
36 System.out.println(m);
37
38

```

This will return -1.

How to use the returned variables-

```

25         System.out.println("plz pass the right customer
26         return -1;
27     }
28
29 }
30
31 public static void main(String[] args) {
32
33     Customer c = new Customer();
34
35     int marks = c.getCustomerMarks("Piyush");
36     System.out.println(marks);
37
38
39     if(marks >= 0) {
40         System.out.println("print the marksheet");
41     }
42
43
44 }
45
46

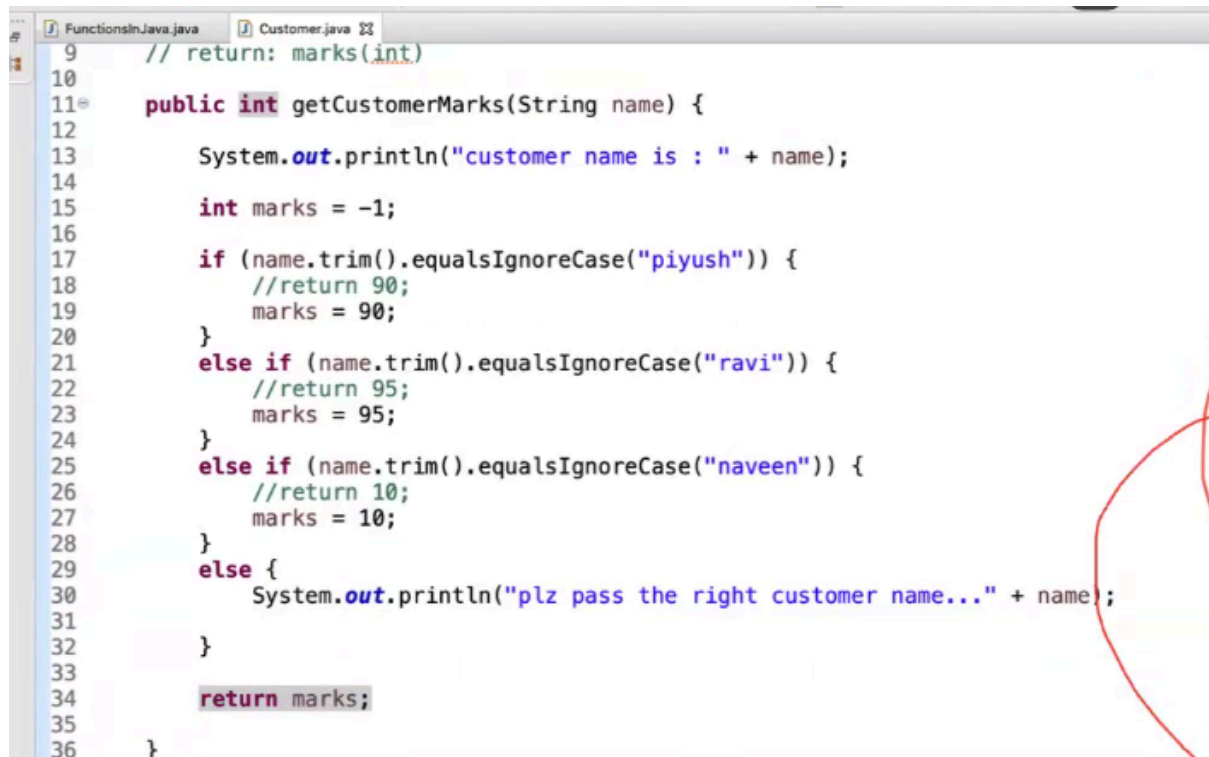
```

```

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

```

## Single return statement but returning marks-



```

9 // return: marks(int)
10
11 public int getCustomerMarks(String name) {
12
13     System.out.println("customer name is : " + name);
14
15     int marks = -1;
16
17     if (name.trim().equalsIgnoreCase("piyush")) {
18         //return 90;
19         marks = 90;
20     }
21     else if (name.trim().equalsIgnoreCase("ravi")) {
22         //return 95;
23         marks = 95;
24     }
25     else if (name.trim().equalsIgnoreCase("naveen")) {
26         //return 10;
27         marks = 10;
28     }
29     else {
30         System.out.println("plz pass the right customer name..." + name);
31     }
32
33
34     return marks;
35
36 }

```

```

37
38 public static void main(String[] args) {
39
40     Customer c = new Customer();
41
42     int marks = c.getCustomerMarks("piyush");
43     System.out.println(marks);
44
45
46     if(marks >= 0) {
47         System.out.println("print the marksheet");
48     }
49
50

```

```

<terminated> Customer (10) [Java Application] /Users/naveenau
customer name is : piyush
90 I
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-



```

37
38 //PSVM - String []
39 public static void main(String a[]) {
40
41     Customer c = new Customer();
42
43     int marks = c.getCustomerMarks("naveen");
44     System.out.println(marks);
45
46
47     if(marks >= 0) {
48         System.out.println("print the marksheet");
49     }
50
51 }
52
53
54 public void main(int a) {
55     System.out.println("bye");
56 }
57
58
59 }
60

```

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-

```
9 // return: marks(int)
10
11 public int getCustomerMarks(String name) {
12     System.out.println("customer name is : " + name);
13     //int marks = -1;
14
15     if (name.trim().equalsIgnoreCase("piyush")) {
16         return 90;
17         //marks = 90;
18     }
19     if (name.trim().equalsIgnoreCase("ravi")) {
20         return 95;
21         //marks = 95;
22     }
23     if (name.trim().equalsIgnoreCase("naveen")) {
24         return 10;
25         //marks = 10;
26     }
27     else {
28         System.out.println("plz pass the right customer name..." + name);
29         return -1;
30     }
31 }
32
33
34
35
```



What If we write return 0 for invalid case-

```

11 public int getCustomerMarks(String name) {
12
13     System.out.println("customer name is : " + name);
14
15     //int marks = -1;
16
17     if (name.trim().equalsIgnoreCase("piyush")) {
18         return 90;
19         //marks = 90;
20     }
21     else if (name.trim().equalsIgnoreCase("ravi")) {
22         return 95;
23         //marks = 95;
24     }
25     else if (name.trim().equalsIgnoreCase("naveen")) {
26         return 10;
27         //marks = 10;
28     }
29     else {
30         System.out.println("plz pass the right customer name..." + name);
31         return 0;
32     }
33 }
34
35 }
36
37 //PSVM - String []
38 public static void main(String a[]) {
39
40     Customer c = new Customer();
41
42     int marks = c.getCustomerMarks("Pooja");
43     System.out.println(marks);
44
45     if(marks>=0) {
46         System.out.println("print the marksheet");
47     }
48
49 }
50
51 }
52
53
54

```

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-

```

11 public int getCustomerMarks(String name) {
12
13     System.out.println("customer name is : " + name);
14
15     //int marks = -1;
16
17     if (name.trim().equalsIgnoreCase("piyush")) {
18         return 90;
19         //marks = 90;
20     }
21     else if (name.trim().equalsIgnoreCase("ravi")) {
22         return 95;
23         //marks = 95;
24     }
25     else if (name.trim().equalsIgnoreCase("naveen")) {
26         return 10;
27         //marks = 10;
28     }
29     else {
30         System.out.println("plz pass the right customer name..." + name);
31         return null;
32     }
33 }
34
35

```



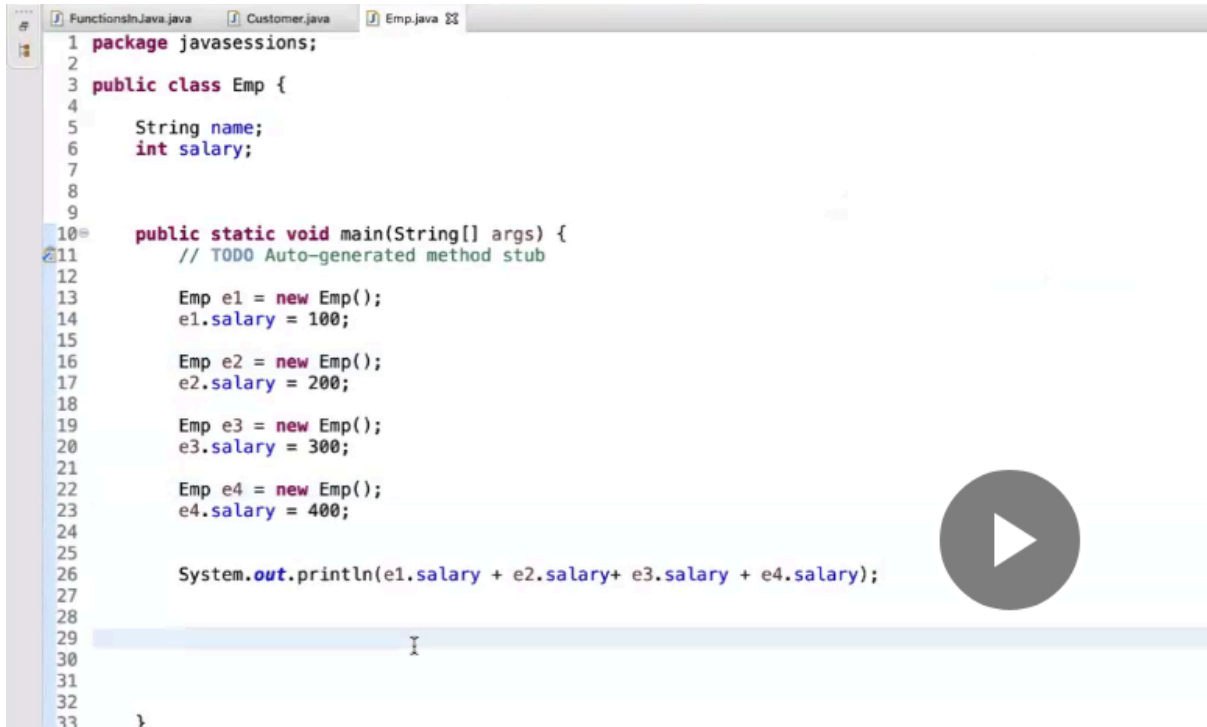
Don't use null as return for primitive data types.

//Type mismatch: cannot convert from null to int

Note-

One method cannot have more than 255 parameters.

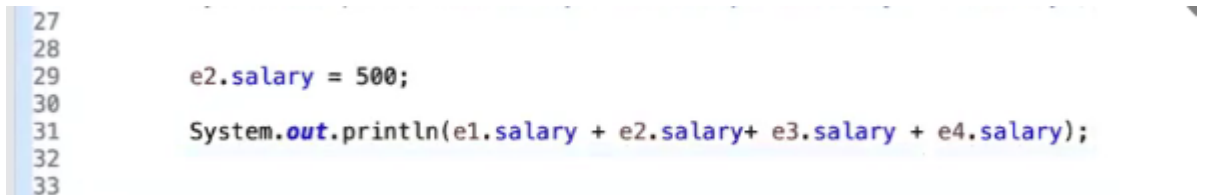
Multiple objects for one class-



```
1 package javasessions;
2
3 public class Emp {
4
5     String name;
6     int salary;
7
8
9
10    public static void main(String[] args) {
11        // TODO Auto-generated method stub
12
13        Emp e1 = new Emp();
14        e1.salary = 100;
15
16        Emp e2 = new Emp();
17        e2.salary = 200;
18
19        Emp e3 = new Emp();
20        e3.salary = 300;
21
22        Emp e4 = new Emp();
23        e4.salary = 400;
24
25
26        System.out.println(e1.salary + e2.salary+ e3.salary + e4.salary);
27
28
29
30
31
32
33 }
```

1000.

Update salary and print total-



```
27
28
29     e2.salary = 500;
30
31     System.out.println(e1.salary + e2.salary+ e3.salary + e4.salary);
32
33
```

1300.