

```
TestingStack.java  A.java  B.java
1 package javasessions;
2
3 public class A {
4
5     public static void main(String[] args) {
6
7         System.out.println("A - main");
8
9         B.main(args);
10
11     }
12
13 }
14
```

```
TestingStack.java  A.java  B.java
1 package javasessions;
2
3 public class B {
4
5     public static void main(String[] args) {
6
7         System.out.println("B - main");
8
9     }
10
11 }
12
13
```

see how to call the main method from another class.

A main

B main

## paste class a

```

classa.java ×
1 package com.day15;
2
3 public class classa {
4
5     public static void main(String[] args) {
6         System.out.println("class a main method");
7         classb.main(args);
8
9         String a[]= {"tiger" , "cat"};
10        classb.main(a);
11
12    }
13
14 }
15
16 //class a main method
17 //b class main method
18 //0
19 //[Ljava.lang.String;@24d46ca6
20 //b class main method
21 //2
22 //[Ljava.lang.String;@4517d9a3
23 //tiger
24 //cat
25
26
27

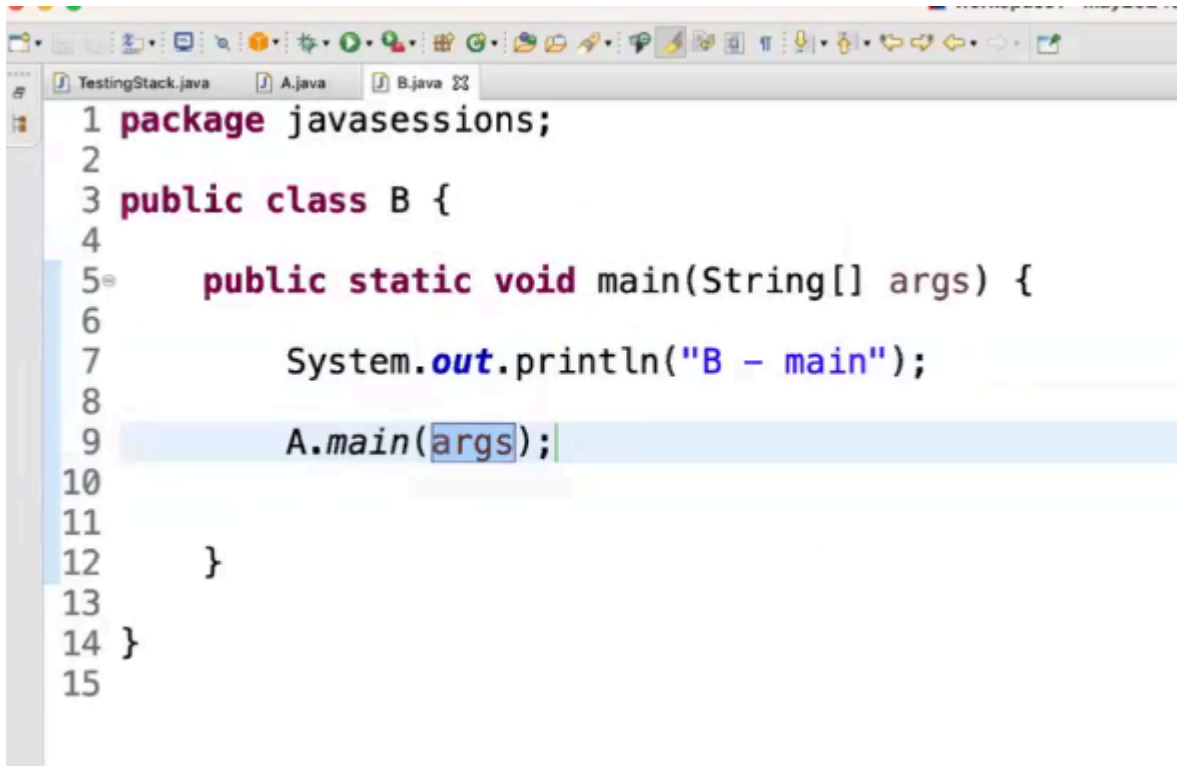
```

## paste class b

```

classb.java ×
1 package com.day15;
2
3 public class classb {
4
5     public static void main(String[] args) {
6         System.out.println("b class main method");
7         // System.out.println(args[0]);
8         // Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds for length 0
9         // at com.day15.classb.main(classb.java:7)
10        System.out.println(args.length); //0
11        System.out.println(args); //memory address printed.- [Ljava.lang.String;@24d46ca6
12
13        // Print each argument
14        for (String arg : args) {
15            System.out.println(arg);
16        }
17
18    }
19
20 }
21

```



```

1 package javasessions;
2
3 public class B {
4
5     public static void main(String[] args) {
6
7         System.out.println("B - main");
8
9         A.main(args);
10
11     }
12 }
13
14 }
15

```

Infinite times each will call other.

Stack overflow error after some time.

```

B - main
Exception in thread "main" java.lang.StackOverflowError
    at java.base/java.io.PrintStream.write(PrintStream.java)
    at java.base/sun.nio.cs.StreamEncoder.writeBytes(StreamEncoder.java)

```

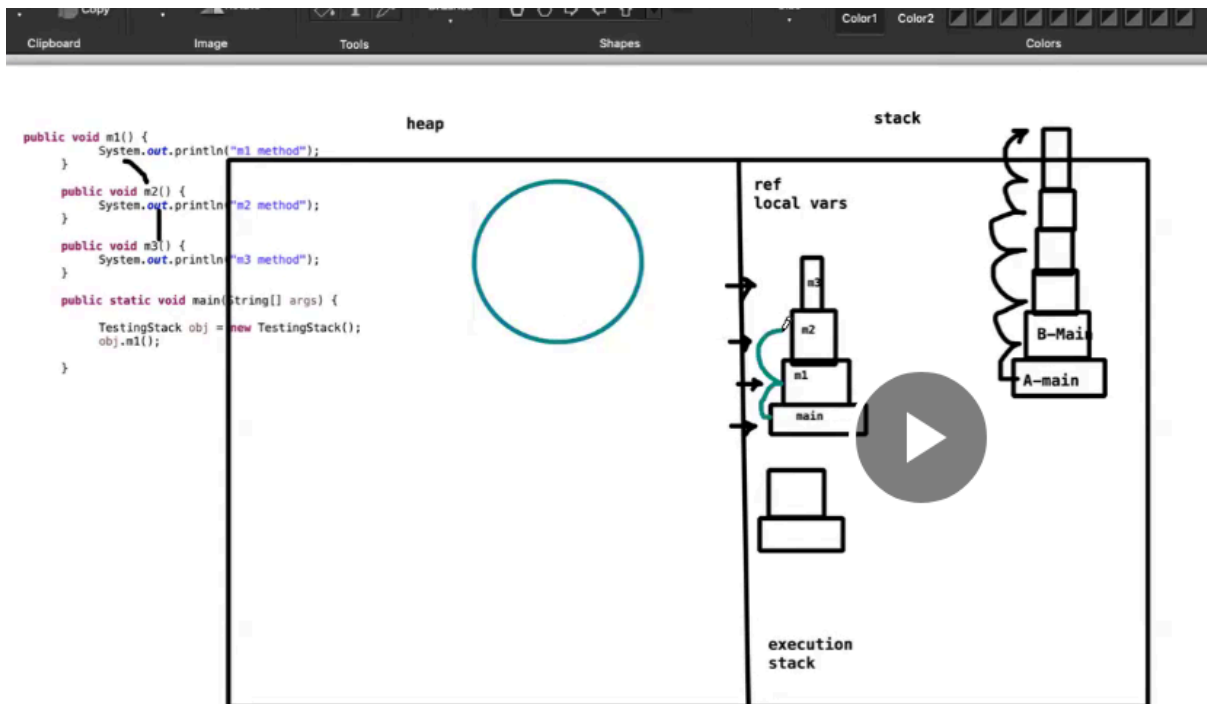
Stack memory also maintains the execution lines.  
the lines which will get executed are actually  
present inside stack memory.

```
TestingStack.java  A.java  B.java
1 package javasessions;
2
3 public class TestingStack {
4
5     public void m1() {
6         System.out.println("m1 method");
7         m2();
8     }
9
10    public void m2() {
11        System.out.println("m2 method");
12        m3();
13    }
14
15    public void m3() {
16        System.out.println("m3 method");
17    }
18
19    public static void main(String[] args) {
20
21        TestingStack obj = new TestingStack();
22        obj.m1();
23
24    }
25
26 }
27
```

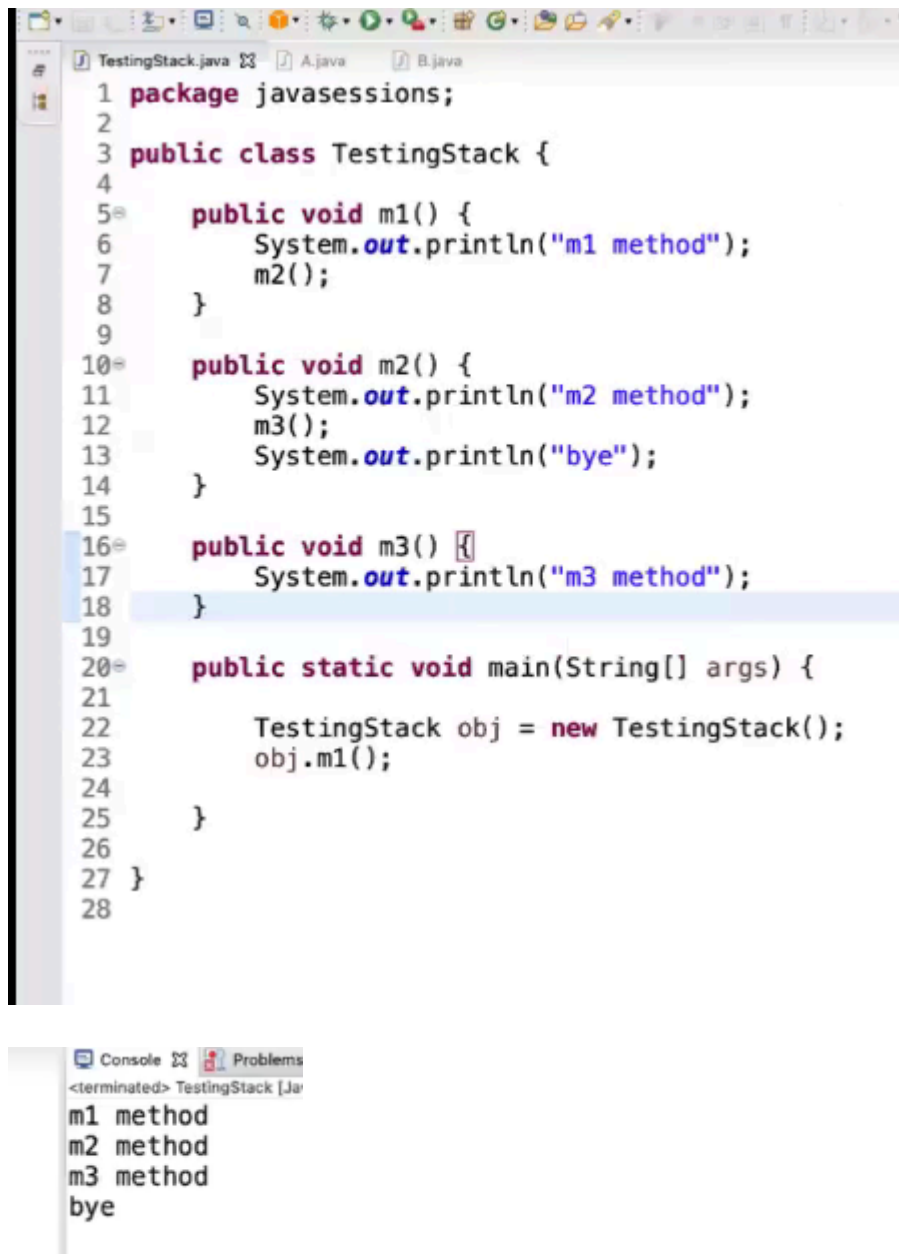
//m1 method

//m2 method

//m3 method



When stack keeps growing, we call it allocation.

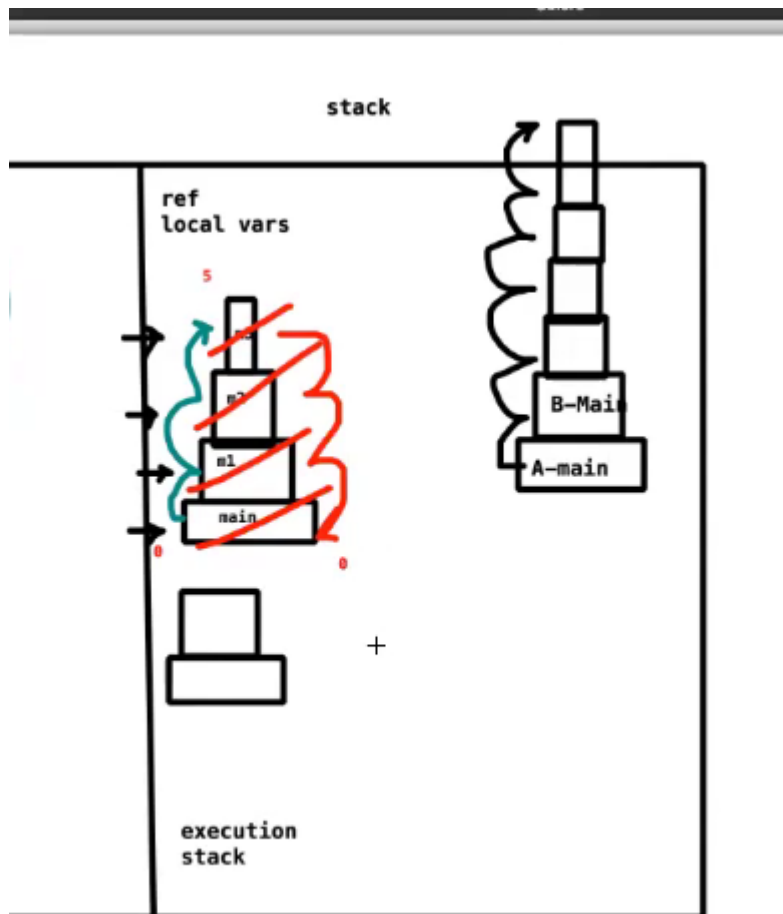


```
1 package javasessions;
2
3 public class TestingStack {
4
5     public void m1() {
6         System.out.println("m1 method");
7         m2();
8     }
9
10    public void m2() {
11        System.out.println("m2 method");
12        m3();
13        System.out.println("bye");
14    }
15
16    public void m3() {
17        System.out.println("m3 method");
18    }
19
20    public static void main(String[] args) {
21
22        TestingStack obj = new TestingStack();
23        obj.m1();
24    }
25 }
26
27 }
28
```

```
<terminated> TestingStack [Jav
m1 method
m2 method
m3 method
bye
```

Deallocation-

Reverse order, from top to bottom as and when job is done.



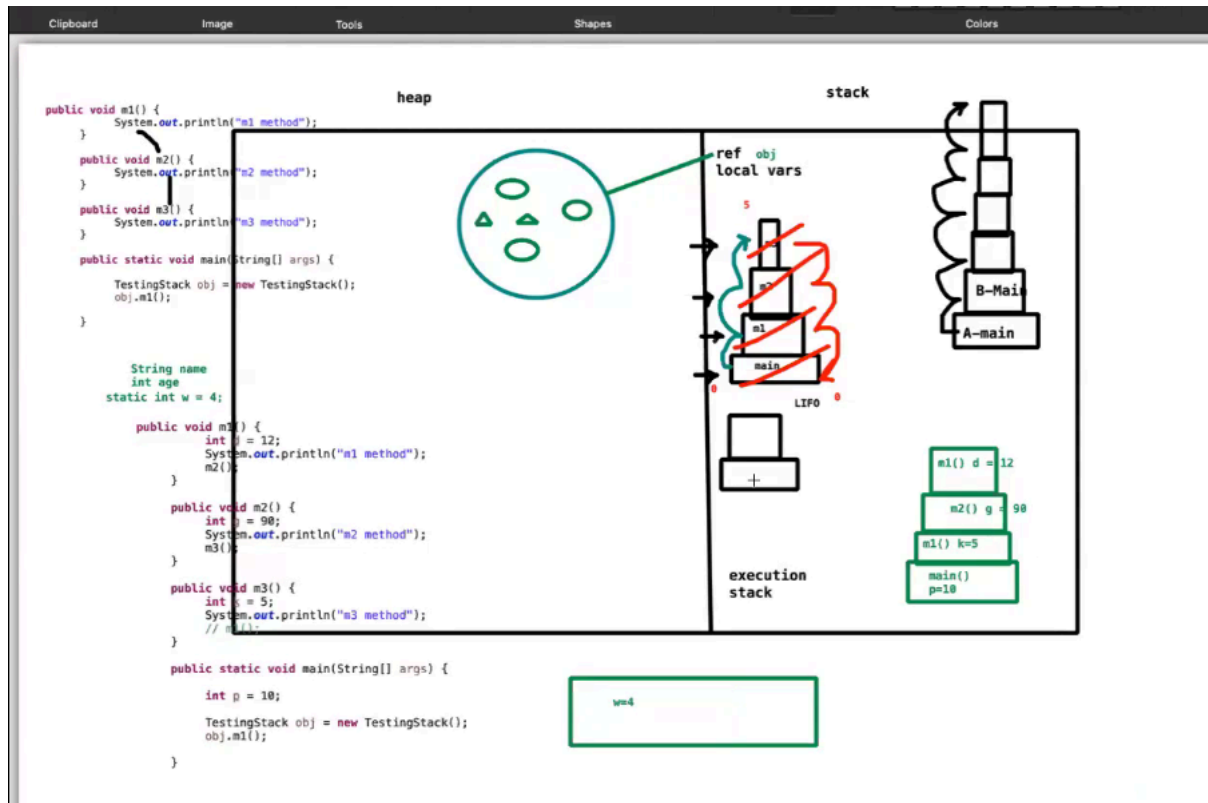
Stack is LIFO hence proved.

Heap memory algo is complex. Stack is simplest.

Stack is for storing reference and local variables.

Heap is for class variables and all copy of class items for the object.

Explanation-





```

1 package javasessions;
2
3 public class TestingStack {
4
5     public void m1() {
6         int d = 12;
7         System.out.println("m1 method");
8         m2();
9     }
10
11    public void m2() {
12        int g = 90;
13        System.out.println("m2 method");
14        m3();
15    }
16
17    public void m3() {
18        int k = 5;
19        System.out.println("m3 method");
20        // m1();
21    }
22
23    public static void main(String[] args) {
24
25        int p = 10;
26
27        TestingStack obj = new TestingStack();
28        obj.m1();
29    }
30 }
31
32 }
33

```

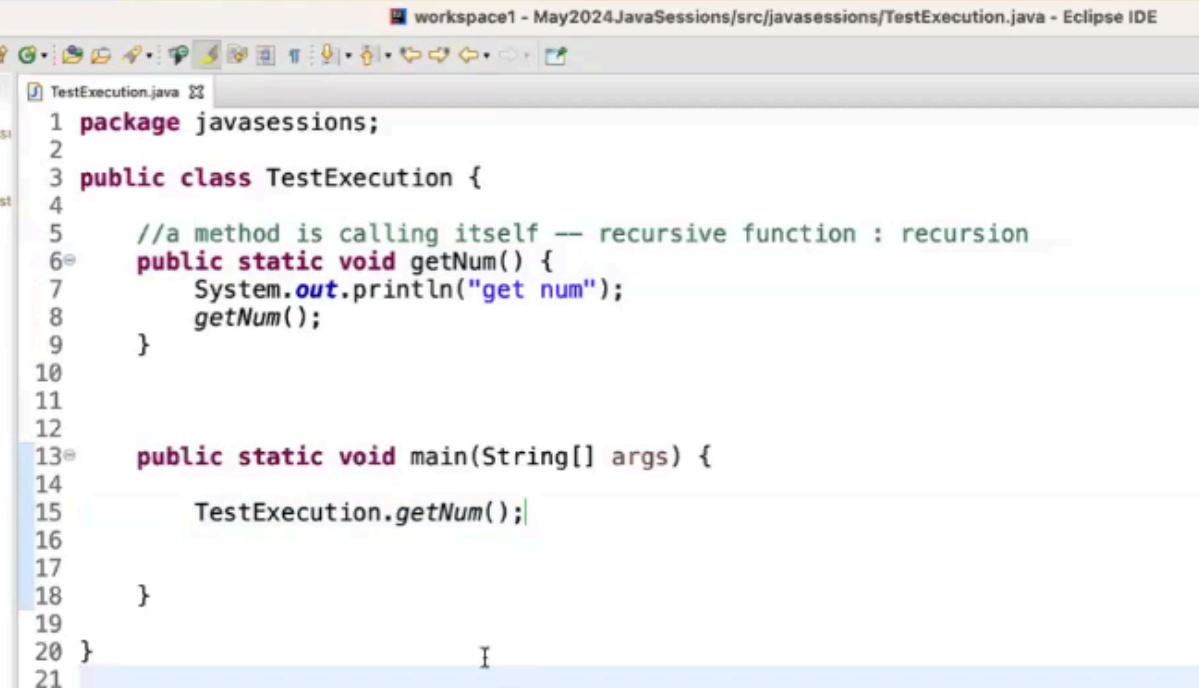
```

//m1 method
//m2 method
//m3 method
//byee

```

To avoid stack overflow, avoid circular flow of the code. M1 calling m2 calling m3 calling m1 etc, is an example of circular flow.

Recursive-



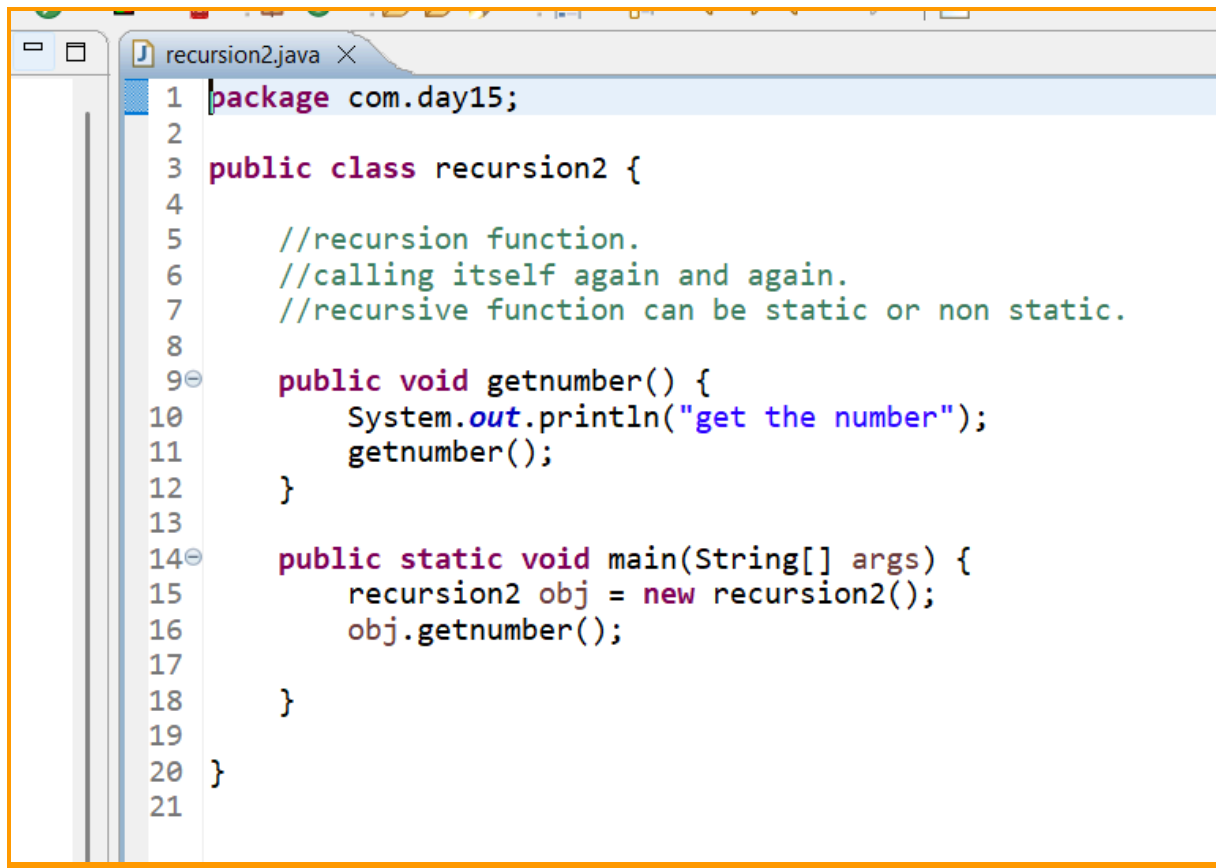
```
1 package javasessions;
2
3 public class TestExecution {
4     //a method is calling itself -- recursive function : recursion
5     public static void getNum() {
6         System.out.println("get num");
7         getNum();
8     }
9
10
11
12
13 public static void main(String[] args) {
14     TestExecution.getNum();
15 }
16
17
18
19
20 }
21
```

Function can be static or non static.

Output is

Getnum and then  
stack overflow.

paste recursion2

A screenshot of a Java IDE window titled 'recursion2.java'. The code defines a package 'com.day15' and a public class 'recursion2'. Inside the class, there is a non-static method 'getnumber()' which prints 'get the number' and then calls itself. There is also a static 'main' method that creates an instance of 'recursion2' and calls 'getnumber()'. The code is as follows:

```
1 package com.day15;
2
3 public class recursion2 {
4
5     //recursion function.
6     //calling itself again and again.
7     //recursive function can be static or non static.
8
9     public void getnumber() {
10         System.out.println("get the number");
11         getnumber();
12     }
13
14     public static void main(String[] args) {
15         recursion2 obj = new recursion2();
16         obj.getnumber();
17     }
18 }
19
20
21
```

Output is  
Getnum and then  
stack overflow.

```

1 package constrcutorConcept;
2
3 public class Employee {
4
5     String name;
6     int age;
7     double salary;
8     char gender;
9     String dob;
10    boolean isPerm;
11
12    public static void main(String[] args) {
13
14        Employee e1 = new Employee();
15        e1.name = "Tom";
16        e1.age = 20;
17        e1.salary = 12.33;
18
19        Employee e2 = new Employee();
20        e1.name = "Tom";
21        e1.age = 20;
22        e1.salary = 12.33;
23
24        Employee e3 = new Employee();
25        e1.name = "Tom";
26        e1.age = 20;
27        e1.salary = 12.33;
28
29        Employee e4 = new Employee();
30        e1.name = "Tom";
31        e1.age = 20;
32        e1.salary = 12.33;
33
34        Employee e5 = new Employee();
35        e1.name = "Tom";
36        e1.age = 20;
37        e1.salary = 12.33;

```

Unnecessary objects can be created-

```

38
39 Employee e5 = new Employee();
40 Employee e5 = new Employee();
41 Employee e5 = new Employee();
42 Employee e5 = new Employee();
43 Employee e5 = new Employee();
44 Employee e5 = new Employee();

```

Garbage collector wont remove it because it removes hanging objects (no reference or null reference).

To avoid these, use constructor.

```

12
13 //constructor:
14 //const... name will be same as the class name
15 //it looks like a function but its not a function
16 //const.. can not return anything, can not be void also -- there is no return type
17 //avoid buss logic in const...
18 //const.. is used to initialize the class (instance) variables
19 //const.. can be overloaded
20 //const.. will be called automatically when we create the object of the class
21
22 public Employee() { //0 param const.. default const..
23     System.out.println("default const...");
24 }
25
26

```

```

33
34
35 public static void main(String[] args) {
36
37     Employee e1 = new Employee();
38
39

```

Default const .....

Constructor overloading possible-

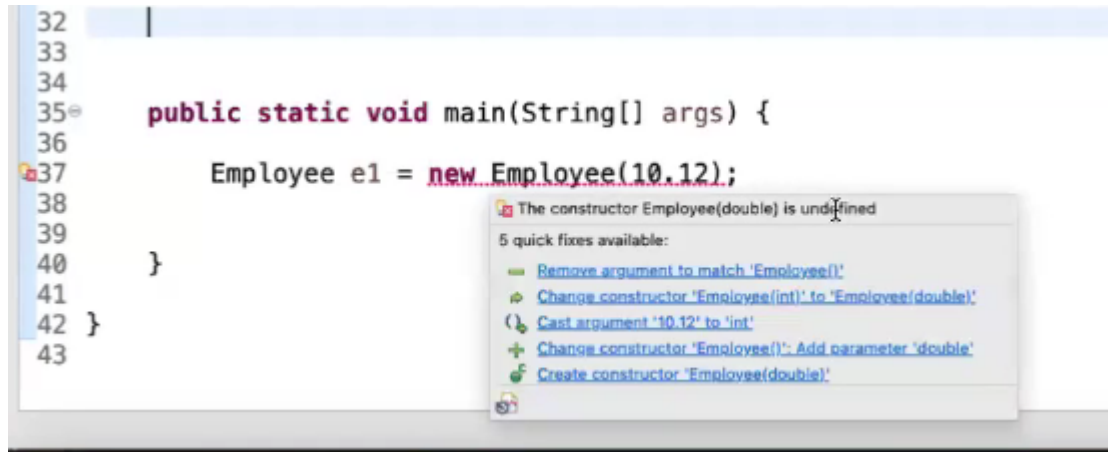
```

22 public Employee() { //0 param const.. default const..
23     System.out.println("default const...");
24 }
25
26
27 public Employee(int a) { //1 param const..
28     System.out.println("1 param const..." + a);
29 }
30
31
32
33
34
35 public static void main(String[] args) {
36
37     Employee e1 = new Employee(10);
38
39

```

```
<terminated> Employee (30) [Java Application]
1 param const...10
```

When constructor not there then error-



```

30 | }
31 |
32 | public Employee(double a) { //1 param const..
33 |     System.out.println("1 param const..." + a);
34 | }
35 |
36 |
37 |
38 |
39 |
40 | public static void main(String[] args) {
41 |
42 |     Employee e1 = new Employee(10.12);
43 |
44 | }
45 |

```

```

Console Problems Javadoc Decla
<terminated> Employee (30) [Java Application] /Users/n
1 param const...10.12

```

```

12 |
13 | //constructor:
14 | //const... name will be same as the class name
15 | //it looks like a function but its not a function
16 | //const.. can not return anything, can not be void also -- there is no return type
17 | //avoid buss logic in const...
18 | //const.. is used to initialize the class (instance) variables
19 | //const.. can be overloaded
20 | //const.. will be called automatically when we create the object of the class
21 | //const... will help to restrict the object creation

```

```

36 // }
37
38 public Employee(String name, int age) {
39
40 }
41
42 public static void main(String[] args) {
43
44     Employee e1 = new Employee("Priya", 30);
45     System.out.println(e1.name + " " + e1.age);
46
47 }

```

## Null 0.

```

12 // constructor:
13 // const... name will be same as the class name
14 // it looks like a function but its not a function
15 // const.. can not return anything, can not be void also -- there is no return
16 // type
17 // avoid buss logic in const...
18 // const.. is used to initialize the class (instance) variables with the help of local variables using this keyword
19 // const.. can be overloaded
20 // const.. will be called automatically when we create the object of the class
21 // const... will help to restrict the object creation

```

```

35 //
36 // }
37
38 public Employee(String name, int age) {
39     name = name;
40     age = age;
41 }
42
43 public static void main(String[] args) {
44
45     Employee e1 = new Employee("Priya", 30);
46     System.out.println(e1.name + " " + e1.age);
47
48 }

```

Null 0.

We are assigning local variable to local variables only.

Change name and it works-

```

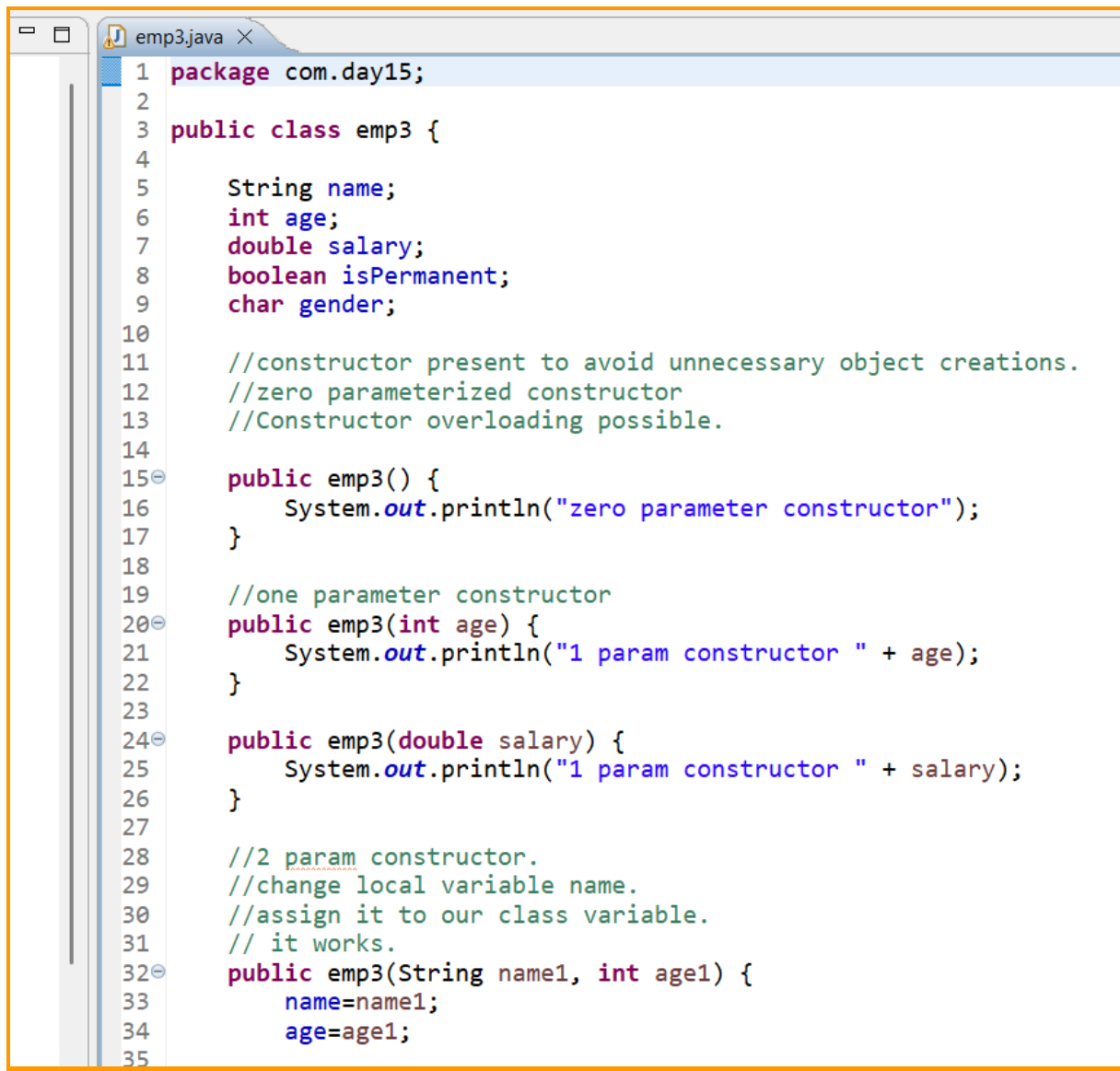
35 //
36 // }
37
38 public Employee(String name1, int age1) {
39     name = name1;
40     age = age1;
41 }
42
43 public static void main(String[] args) {
44
45     Employee e1 = new Employee("Priya", 30);
46     System.out.println(e1.name + " " + e1.age);
47
48 }

```

Priya 30

paste emp3





```
1 package com.day15;
2
3 public class emp3 {
4     String name;
5     int age;
6     double salary;
7     boolean isPermanent;
8     char gender;
9
10    //constructor present to avoid unnecessary object creations.
11    //zero parameterized constructor
12    //Constructor overloading possible.
13
14    public emp3() {
15        System.out.println("zero parameter constructor");
16    }
17
18    //one parameter constructor
19    public emp3(int age) {
20        System.out.println("1 param constructor " + age);
21    }
22
23    public emp3(double salary) {
24        System.out.println("1 param constructor " + salary);
25    }
26
27    //2 param constructor.
28    //change local variable name.
29    //assign it to our class variable.
30    // it works.
31    public emp3(String name1, int age1) {
32        name=name1;
33        age=age1;
34    }
35}
```

```

34         age=age1;
35
36     }
37     public static void main(String[] args) {
38
39         emp3 e1=new emp3();
40         emp3 e2=new emp3(30);
41         emp3 e4=new emp3(50000.454350);
42         emp3 e5=new emp3("tiger", 30);
43         System.out.println(e5.name + " " + e5.age);
44
45     }
46
47 }
48
49 //zero parameter constructor
50 //1 param constructor 30
51 //1 param constructor 50000.45435
52 //tiger 30
53
54
55

```

To access global variables use this-

```

35 //
36 // }
37
38 public Employee(String name, int age) {
39     //this.Global=Local
40     this.name = name;
41     this.age = age;
42 }
43
44 public static void main(String[] args) {
45
46     Employee e1 = new Employee("Priya", 30);
47     System.out.println(e1.name + " " + e1.age);
48
49 }

```

Priya 30

```

44 public static void main(String[] args) {
45
46     Employee e1 = new Employee("Priya", 30);
47     System.out.println(e1.name + " " + e1.age + " " + e1.salary + " " + e1.dob + " " + e1.isPerm);
48
49 }

```

```

Console 25 Problems Javadoc Decla
<terminated> Employee (30) [Java Application] /Users/n
Priya 30 0.0 null false

```

```

43
44 public Employee(String name, int age, double salary, char gender) {
45     //this.Global=Local
46     this.name = name;
47     this.age = age;
48     this.salary = salary;
49     this.gender = gender;
50 }

```

```

58
59 Employee e2 = new Employee("Devesh", 25, 30.66, 'm');
60 System.out.println(e2.name + " " + e2.age + " " + e2.salary + " " + e2.dob + " " + e2.isPerm + " " + e2.gender);
61
62

```

```

Devesh 25 30.66 null false m

```

## Error when creating object without any value-

```

62
63 Employee e3 = new Employee();
64
65 }
66

```

//The constructor **emp6()** is undefined

## Constructors restrict unnecessary object creation.

```

51     }
52     }
53     public Employee(String name, int age, char gender, String dob) {
54         this.name = name;
55         this.age = age;
56         this.gender = gender;
57         this.dob = dob;
58     }
59
60     public Employee(String name, int age, double salary, char gender, String dob, boolean isPerm) {
61         this.name = name;
62         this.age = age;
63         this.salary = salary;
64         this.gender = gender;
65         this.dob = dob;
66         this.isPerm = isPerm;
67     }

```

```

77     Employee e3 = new Employee("naveen", 25, 12.33, 'm', "01-01-2000", true);
78     System.out.println(e3.name + " " + e3.age + " " + e3.salary + " " + e3.dob + " " + e3.isPerm + " " + e3.gender);
79

```

```

Devesh 25 30.66 null false m
naveen 25 12.33 01-01-2000 true m

```

## Update values-

```

72     System.out.println(e1.name + " " + e1.age + " " + e1.salary + " " + e1.dob + " " + e1.isPerm);
73     e1.salary = 23.44;
74     System.out.println(e1.name + " " + e1.age + " " + e1.salary + " " + e1.dob + " " + e1.isPerm);

```

```

Priya 30 23.44 null false
Devesh 25 30.66 null false

```

## Default constructor added-

```

38
39     public Employee() { //default
40
41     }

```

Now the whole purpose is lost. Now we can create unnecessary objects also.

```

Employee.java Car.java
2
3 public class Car {
4
5     String name;
6     String color;
7     double price;
8     String model;
9     String chasisNumber;
10
11     public Car(String name, String color) {
12         this.name = name;
13         this.color = color;
14     }
15
16     public Car(String name, double price) {
17         this.name = name;
18         this.price = price;
19     }
20
21     public Car(String name, String color, double price, String model) {
22         this.name = name;
23         this.color = color;
24         this.price = price;
25         this.model = model;
26     }
27
28     public Car(String name, String color, double price, String model, String chasisNumber) {
29         this.name = name;
30         this.color = color;
31         this.price = price;
32         this.model = model;
33         this.chasisNumber = chasisNumber;
34     }
35
36     public static void main(String[] args) {
37
38
39         Car c1 = new Car("BMW", 50.44);
40         Car c2 = new Car("BMW", "Red", 80.44, "x3");
41         Car c3 = new Car("Audi", "Green", 75.44, "q3", "12121wewew");
42
43
44         System.out.println(c1.name + " " + c1.price + " " + c1.model + " " + c1.chasisNumber + " " + c1.color);
45         System.out.println(c2.name + " " + c2.price + " " + c2.model + " " + c2.chasisNumber + " " + c2.color);
46         System.out.println(c3.name + " " + c3.price + " " + c3.model + " " + c3.chasisNumber + " " + c3.color);
47
48     }
49
50

```

Console Problems Javadoc Declaration Res

```

<terminated> Car (10) [Java Application] /Users/naveenautomationlabs/
BMW 50.44 null null null
BMW 80.44 x3 null Red
Audi 75.44 q3 12121wewew Green

```

Constructor versus function-

```

11
12 // constructor:
13 // const... name will be same as the class name but function name could be anything
14 // it looks like a function but its not a function
15 // const.. can not return anything, can not be void also — there is no return but function may or may not return
16 // avoid buss logic in const...but function should have the buss logic
17
18 // const.. is used to initialize the class (instance) variables with the help of local variables using this keyword
19
20 // const.. can be overloaded, same with function also
21
22 // const.. will be called automatically when we create the object of the class
23 //but function will be called using the object ref name or static function will be called using class name
24
25 // const... will help to restrict the object creation
26
27
28 //const vs function:

```

Don't create function name same as class name-  
constructors cannot have return type.

Gives warning and bad practice.

```

29
30
31 // public void Employee() {
32 //
33 // }

```

//This method has a constructor name

Constructor cannot have static keyword-

```

55 public static Employee(String name, int age) {
56     // this.Global=Local
57     this.name = name;
58     this.age = age;
59 }

```

Constructor and objects are related. Objects call constructor. Objects cannot come with static.

//Illegal modifier for the constructor in type function1; only public, protected & private are permitted.

Constructor stored in class loader.(1.30.07)

We can add anything inside constructor-

Static variable created and accessed inside constructor.

```

11
12 static String compName = "IBM";|
13
56
57 public Employee(String name, int age) {
58     // this.Global=Local
59     this.name = name;
60     this.age = age;|
61     System.out.println(Employee.compName);|
62 }
63
64 public Employee(String name, int age, double salary) {

```

paste function2

```

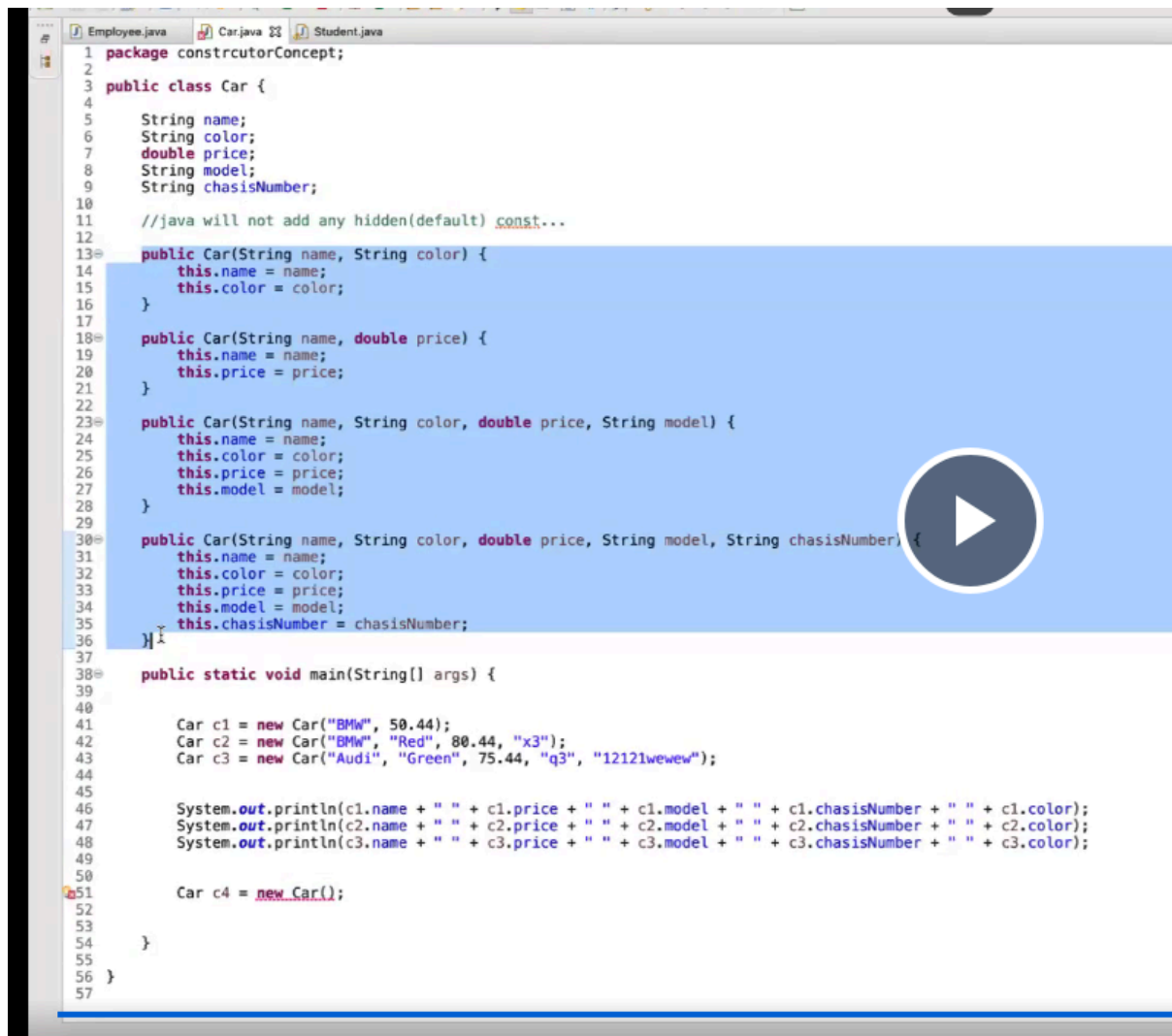
function2.java ×
1 package com.day15;
2
3 public class function2 {
4
5     String name;
6     int age;
7
8     //we can have any damn thing inside constructor.
9     //create static variable string called company name and assign value.
10    public static String companyName = "TCS";
11
12    //create constructor with two parameters string name and int age.
13    //print the value of static variable inside constructor.
14    public function2(String name, int age) {
15        this.name = name;
16        this.age = age;
17        System.out.println("company name is " + companyName);
18    }
19
20    //create main method.
21    public static void main(String[] args) {
22        //create two objects of function2 class.
23        function2 obj1 = new function2("A", 1);
24        function2 obj2 = new function2("B", 2);
25
26        //print obj1 and obj2 values.
27        System.out.println(obj1.name + " " + obj1.age);
28        System.out.println(obj2.name + " " + obj2.age);
29    }
30
31 }
32
33 //company name is TCS
34 //company name is TCS
35 //A 1
36 //B 2
37
38

```

Note-

Once we define our constructor then default constructor will be removed from that class.



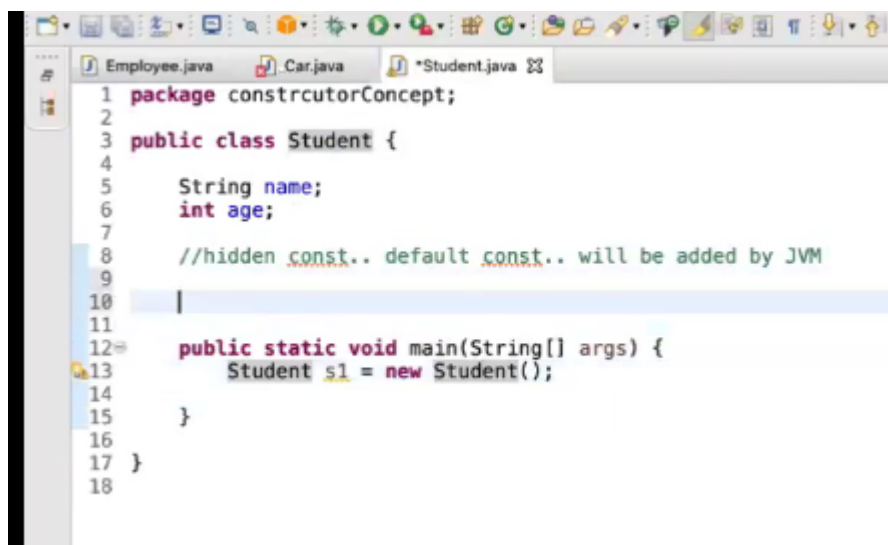


```

1 package construtorConcept;
2
3 public class Car {
4
5     String name;
6     String color;
7     double price;
8     String model;
9     String chasisNumber;
10
11     //java will not add any hidden(default) const...
12
13     public Car(String name, String color) {
14         this.name = name;
15         this.color = color;
16     }
17
18     public Car(String name, double price) {
19         this.name = name;
20         this.price = price;
21     }
22
23     public Car(String name, String color, double price, String model) {
24         this.name = name;
25         this.color = color;
26         this.price = price;
27         this.model = model;
28     }
29
30     public Car(String name, String color, double price, String model, String chasisNumber) {
31         this.name = name;
32         this.color = color;
33         this.price = price;
34         this.model = model;
35         this.chasisNumber = chasisNumber;
36     }
37
38     public static void main(String[] args) {
39
40         Car c1 = new Car("BMW", 50.44);
41         Car c2 = new Car("BMW", "Red", 80.44, "x3");
42         Car c3 = new Car("Audi", "Green", 75.44, "q3", "12121wewew");
43
44         System.out.println(c1.name + " " + c1.price + " " + c1.model + " " + c1.chasisNumber + " " + c1.color);
45         System.out.println(c2.name + " " + c2.price + " " + c2.model + " " + c2.chasisNumber + " " + c2.color);
46         System.out.println(c3.name + " " + c3.price + " " + c3.model + " " + c3.chasisNumber + " " + c3.color);
47
48         Car c4 = new Car();
49     }
50 }
51
52
53
54
55
56
57

```

//The constructor function3() is undefined



```

1 package construtorConcept;
2
3 public class Student {
4
5     String name;
6     int age;
7
8     //hidden const.. default const.. will be added by JVM
9
10     |
11
12     public static void main(String[] args) {
13         Student s1 = new Student();
14     }
15 }
16
17 }
18

```

## Static variables initialising-

```
11
12 static int w;|
13
```

No error but warning.

```
56 // }
57
58 public Employee(String name, int age) {
59     // this.Global=Local
60     this.name = name;
61     this.age = age;
62     this.w= 4;
63 }
```

This keyword is again related to objects not class.(1.43.53)

//warning - The assignment to variable w has no effect

## Right way for static-

```
56 // }
57
58 public Employee(String name, int age) {
59     // this.Global=Local
60     this.name = name;
61     this.age = age;
62     Employee.w= 4;
63 }
64 }
```

```
100
101 Employee e3 = new Employee("naveen", 25, 12.33, 'm', "01-01-2000", true);
102 System.out.println(e3.name + " " + e3.age + " " + e3.salary + " " + e3.dob + " " + e3.isPerm + " " + e3.gender + Employee.w);
103
```

```
Devesh 25 30.66 null false m
naveen 25 12.33 01-01-2000 true m4 I
```

## Static and local mix-

```

55 //
56 // }
57
58 public Employee(String name, int age, int w) {
59     // this.Global=Local
60     this.name = name;
61     this.age = age;
62     Employee.w = w;
63
64 }
65

```

paste emp11

```

emp11.java X
1 package com.day15;
2
3 public class emp11 {
4
5     String name;
6     int age;
7     double salary;
8     boolean isPermanent;
9     char gender;
10
11     //static variable.
12     static int w;
13
14     //create constructor with all arguments.
15     //passed static into constructor also.
16     //no error.
17 public emp11(String name, int age, double salary, boolean isPermanent, char gender, int w) {
18     this.name=name;
19     this.age=age;
20     this.salary=salary;
21     this.isPermanent=isPermanent;
22     this.gender=gender;
23
24     //initialised static with this.
25     //not right way.
26     //warning.
27     //The static field emp11.w should be accessed in a static way
28     this.w=34324;
29 }
30

```

```

28         this.w=34324;
29     }
30
31     //create main method.
32     public static void main(String[] args) {
33
34         emp11 e1=new emp11("karan", 4324, -34234.544545,false, 'm', 50);
35         System.out.println(e1);
36         System.out.println(e1.name + " " + e1.age + " " + e1.gender + " " + e1.salary + " "
37         +e1.isPermanent + " " +emp11.w);
38     }
39
40 }
41
42 //com.day15.emp11@5ca881b5
43 //karan 4324 m -34234.544545 false 34324
44
45
46
47
48
49
50
51
52

```

Writable Smart Insert 37 : 9 : 888

paste emp12

```

emp12.java
1 package com.day15;
2
3 public class emp12 {
4
5     String name;
6     int age;
7     double salary;
8     boolean isPermanent;
9     char gender;
10
11     //static variable.
12     static int w;
13
14     //create constructor with all arguments.
15     //not pasted the static to constructor.
16     //no error.
17     public emp12(String name, int age, double salary, boolean isPermanent, char gender) {
18         this.name=name;
19         this.age=age;
20         this.salary=salary;
21         this.isPermanent=isPermanent;
22         this.gender=gender;
23
24         //initialised static with this.
25         //not right way.
26         //warning.
27         //The static field emp11.w should be accessed in a static way
28         this.w=34324;
29     }
30
31     //create main method.
32     public static void main(String[] args) {
33

```

```
30
31 //create main method.
32 public static void main(String[] args) {
33
34     emp12 e1=new emp12("karan", 4324, -34234.544545,false, 'm');
35     System.out.println(e1);
36     System.out.println(e1.name + " " + e1.age + " " + e1.gender + " " + e1.salary + " "
37         +e1.isPermanent + " " +emp12.w);
38 }
39
40 }
41
42 //com.day15.emp12@5ca881b5
43 //karan 4324 m -34234.544545 false 34324
44
```

Writable

Smart Insert

44 : 1 : 993



Codeaur