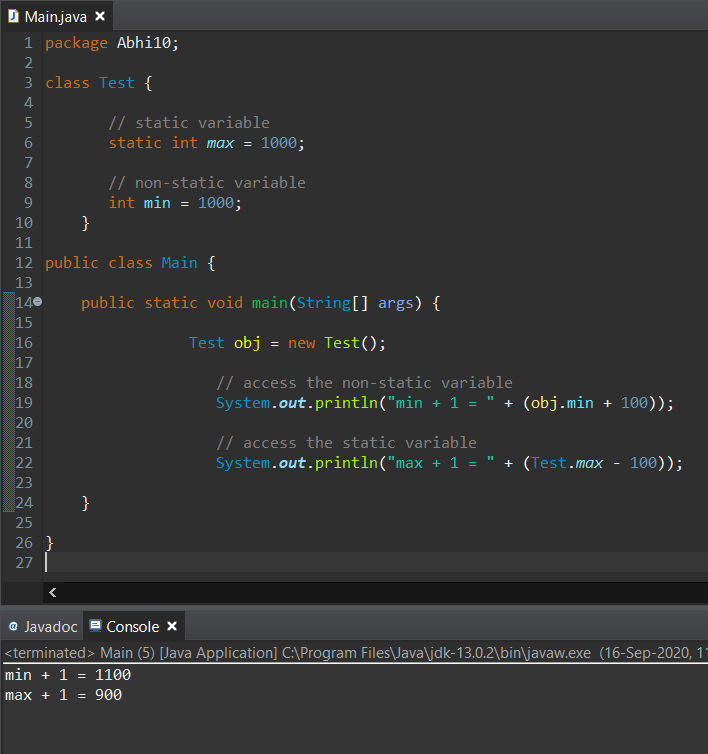
**19CSE204**

**OBJECT ORIENTED PARADIGM**

**S. Abhishek**

**AM. EN. U4CSE19147**



class Test {

This line is basically the declaration of the member named max which is of type int and declared with static keyword.

This variable is initialized to the value 1000 and it can be only accessed by the static member function.

**static int max = 1000;**

This is the declaration of another member which is non static and of type int.

It is initialized to the value 1000.

**int min = 1000;**

}

public class Main {

public static void main (String[] args) {

Creating of the object of class test and the instances of the class Test() is now invoked in the object “obj”.

**Test obj = new Test();**

This Print statement prints the value of the variable “min” which is declared in the class test() and called and invoked in the object “obj”.

Here we are accessing it through the object since it is a non static member.

The value of the member is accessed by the “obj.min” where “obj” is the object name and “.” Is the operator used to access the value and “min” is the member.

**System.out.println("min + 1 = " + (obj.min + 100));**

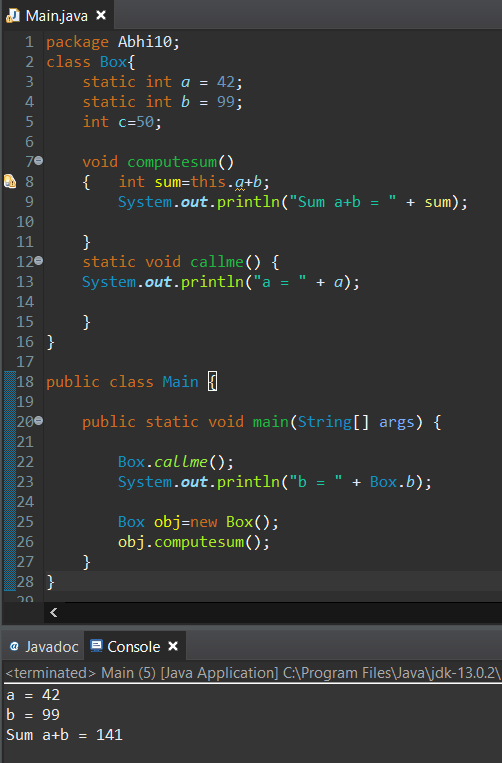
This Print statement prints the value of the variable “max” which is declared in the class test().

It is accessed directly by using the name of the class since it is a static member. Rather than using the “object.member” here we use “class.member”.

**System.out.println("max + 1 = " + (Test.max - 100));**

}

}



class Box{

This line is basically the declaration of the member named “a” which is of type int and declared with keyword static which is initialized to the value 42 and it can be only accessed by the static member function.

**static int a= 42;**

This line is basically the declaration of the member named “b” which is of type int and declared with keyword static which is initialized to the value 99 and it can be only accessed by the static member function.

**static int b= 99;**

This is the declaration of another member which is non static and of type int and it is initialized to the value 50.

**int c=50;**

This is the declaration of member function which is non static and of type void which doesn’t return anything.

**void computesum()**

**{**

In this “this.a+b” is used to add the two static members. We can use static members in the non-static member function but can’t use the non-static members in the static methods.

**int sum=this.a+b;**

This line prints the value of the sum that is calculated.

**System.out.println("Sum a+b = " + sum); }**

This is the declaration of member function which is static and of type void which doesn’t return anything.

**static void callme() {**

This line prints the value “42”, of the static member “a” that is initialized in it.

**System.out.println("a = " +a);**

**}**

**}**

**public class Main {**

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

Here we are directly calling the static member function of the class Box() to print the value “42”, of the static member “a” that is initialized in the same class Box().

**Box.callme();**

Here we are directly calling the static member “b” of the class Box() to print the value “99” that is initialized in the same class Box().

**System.out.println("b = " + Box.b);**

Creating of the object of class test and the instances of the class Box() is now invoked in the object “obj”.

**Box obj=new Box();**

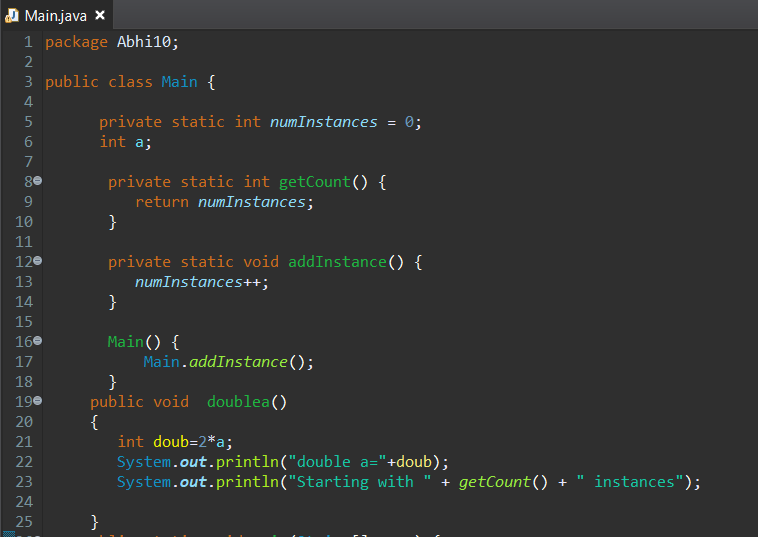
This statement calls the member function “computesum()” of the class Box() which is now invoked in the object “obj”.

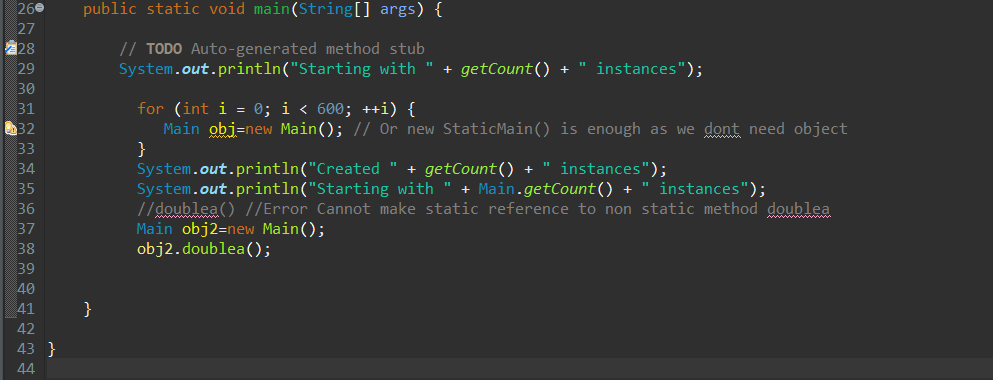
It line prints the value of the sum 42 + 99 = 141.

**obj.computesum();**

**}**

**}**

****

****

This statement is to create a static member of type int.

**private static int numInstances= 0;**

This statement is to create a non-static member of type int.

**int a;**

This statement is to create a static member function named “getCount()” of type int with private as access specifier.

**private static int getCount() {**

This static member function returns numInstances when it is called.

**return numInstances;**

**}**

This statement is to create a static member function named “addInstance()” of type int with private as access specifier.

**private static void addInstance() {**

This static member function increments the numInstances when it is called.

**numInstances++**

**}**

This is a Default Constructor which calls the method addinstance when this constructor is called.

**Main() {**

**Main.addinstance;**

**}**

This statement is to create a non-static member function named “void doublea()”of type int with public as access specifier. It performs some computation.

**public void doublea() {**

**int doub=2\*a;**

Prints the value stored in the member doub

**System.out.println("double a="+doub);**

Calls the “getCount()” method which returns numInstances

**System.*out*.println("Starting with " + *getCount*() + " instances");**

**}**

Calls the “getCount()” method which returns numInstances which is initially 0;

**System.out.println("Starting with " + getCount() + " instances");**

This For loop iterates 600 times and creates the obj 600 times which calls the constructor 600 times which calls the “addinstance()” method 600 times and keeps on incrementing the “numInstances”.

**for (int i = 0; i < 600; ++i) {**

**Main obj=new Main(); }**

Calls the “getCount()” method without the class name which returns “numInstances” which is now 600.

**System.out.println("Created " + getCount() + " instances");**

Calls the “getCount()” method with the class name which returns “numInstances” which is now 600.

**System.out.println("Starting with " + Main.getCount() + " instances");**

This statement will throw an error because we are trying to call the non-static member function directly using class.

Only static member functions can be accessed directly using class.

Non static member function can be only accessed using the object of the particular class.

**“doublea()”**

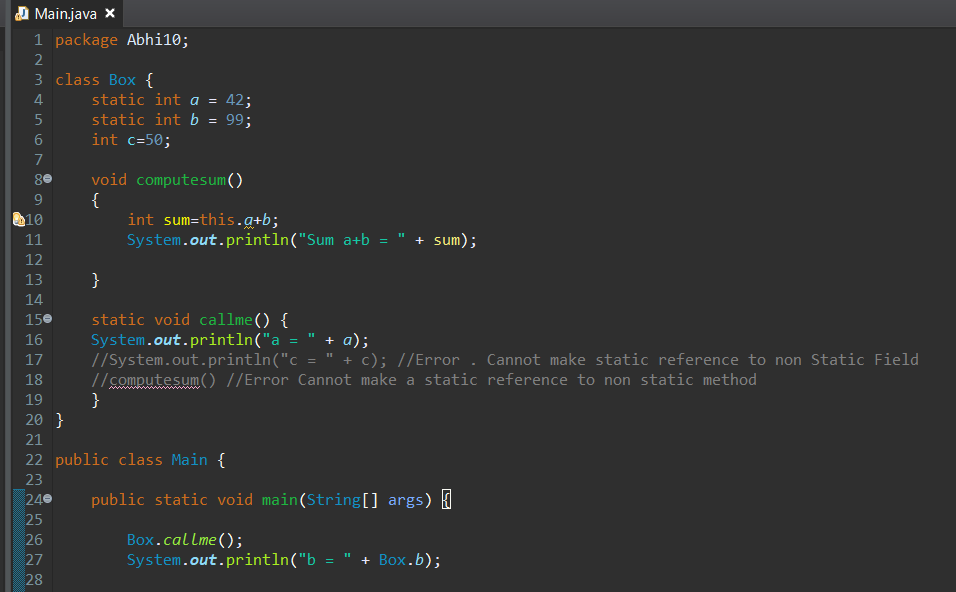
Creating of the object of class Main() and the instances of the class Main() is now invoked in the object “obj2”.

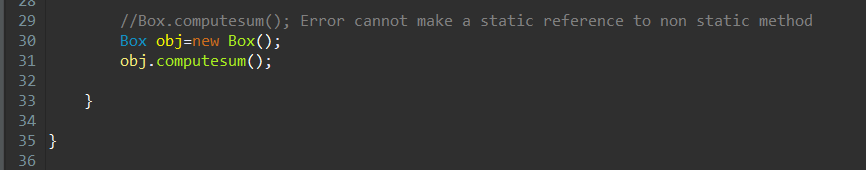
**Main obj2=new Main();**

Accessing the member function “doublea()” through the object “obj2” since the member function is non-static. It returns the “numInstances” through the “getcount()” method called inside it.

**obj2.doublea();**

**} }**

****

****

**static void callme() {**

**System.*out*.println("a = " + *a*);**

You cannot access the value of the non-static member in a static method. Show this statement throws an Error.

**//System.out.println("c = " + c);**

You cannot access the non-static member function in a static method. Show this statement throws an Error.

**//computesum() }**

This statement will throw an error because we are trying to call the non-static member function directly using class.

Only static member functions can be accessed directly using class.

Non static member function can be only accessed using the object of the particular class.

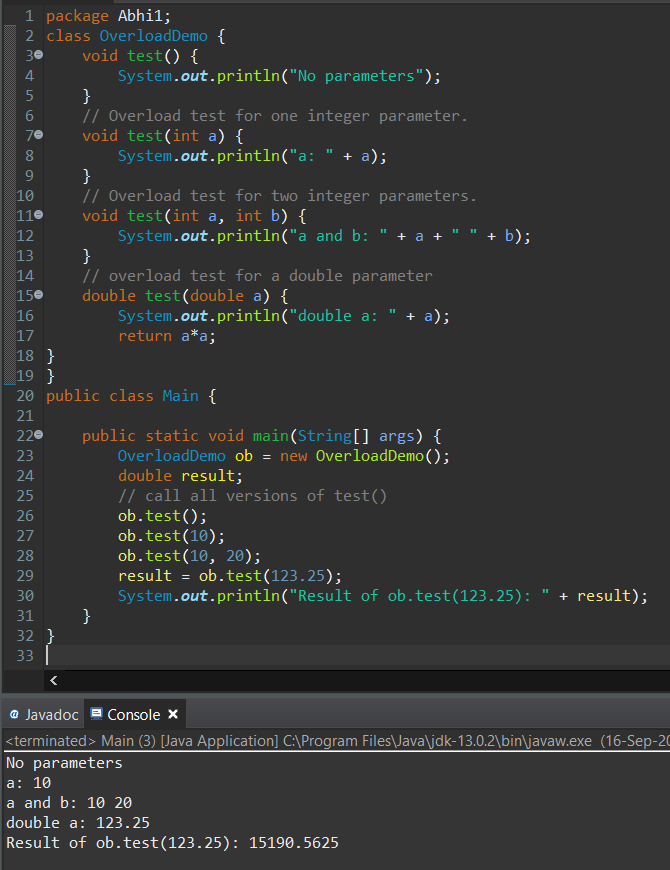
**//Box.computesum();**

Creating the object of class Box() and the instances of the class Box() is now invoked in the object “obj”.

**Box obj=new Box();**

Non static member function “computesum()” can be only accessed using the object of the class Box().

**obj.computesum();**

****

If two or more [methods](https://www.programiz.com/java-programming/methods) have same name and differ in number of parameters or data types of parameters then these methods are called overloaded methods and this feature is called method overloading.

Here the class OverloadDemo have 4 member function with same name and different parameters and different data types. So they are considered as the method overloading.

**class OverloadDemo {**

**1 -** This Method is called when no parameters are passed.

**void test() {**

**System.out.println("No parameters");**

**}**

**2 -** This Method is called when only one parameter is passed.

**void test(int a) {**

**System.out.println("a: " + a);**

**}**

**3 -** This Method is called when two parameters are passed with data type int.

**void test(int a, int b) {**

**System.out.println("a and b: " + a + " " + b);}**

**4 -** This Method is called when two parameters are passed with data type double.

**double test(double a) {**

**System.out.println("double a: " + a);**

**return a\*a;**

**}**

Creating the object of class **OverloadDemo**() and the instances of the class **OverloadDemo**() is now invoked in the object “ob”.

**OverloadDemo ob = new OverloadDemo();**

Declaration of member “result” with type double.

**double result;**

Calling the member function test() using the object “ob” without passing any parameter. So it calls the method 1.

**ob.test();**

Calling the member function test() using the object “ob” with only one parameter. So it calls the method 2.

**ob.test(10);**

Calling the member function test() using the object “ob” with two parameters of type int. So it calls the method 3.

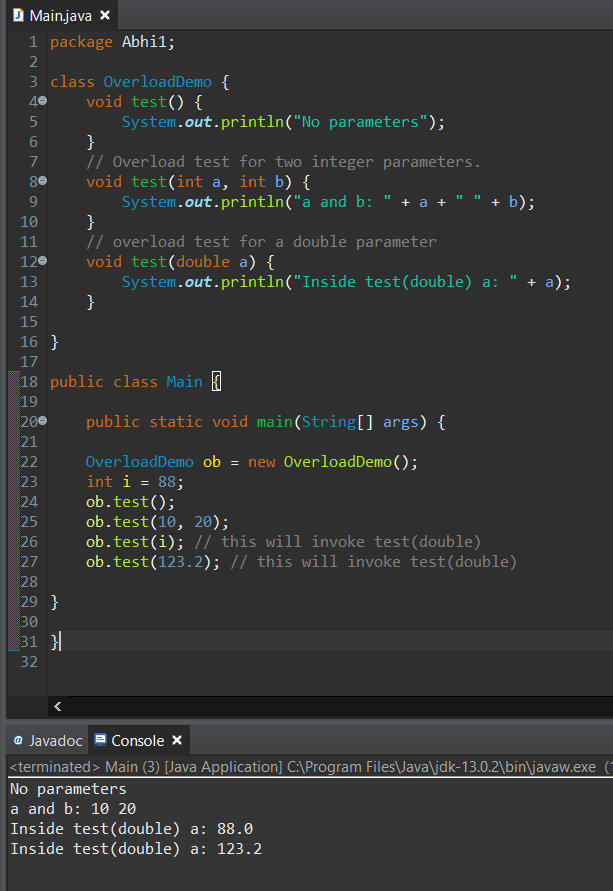
**ob.test(10, 20);**

Calling the member function test() using the object “ob” with two parameters of type double. So it calls the method 4.

**result = ob.test(123.25);**

This statement prints the result which have the value computed from the member function test() using the object “ob” with two parameters of type double.

**System.out.println("Result of ob.test(123.25): " + result);**



**1 -** This Method is called when no parameters are passed.

**void test() {**

**System.out.println("No parameters");**

**}**

**2 -** This Method is called when two parameters are passed with data type int.

**void test(int a, int b) {**

**System.out.println("a and b: " + a + " " + b);**

**}**

**3 -** This Method is called when one parameter is passed with data type double.

**void test(double a) {**

**System.out.println("Inside test(double) a: " + a);**

**}**

Creating the object of class **OverloadDemo**() and the instances of the class **OverloadDemo**() is now invoked in the object “ob”.

**OverloadDemo ob = new OverloadDemo();**

Declaration of member “i” with type int and assigning the value 88 to it.

**int i = 88;**

Calling the member function test() using the object “ob” without passing any parameter. So it calls the method 1.

**ob.test();**

Calling the member function test() using the object “ob” which passes two parameters. So it calls the method 2.

**ob.test(10, 20);**

Calling the member function test() using the object “ob” which passes one parameter of type int.

Here member “i” is int but the member function called is of type double. It doesn’t find any method with type double. So it type casts the int member to double.

**ob.test(i); // this will invoke test(double)**

Calling the member function test() using the object “ob” which passes one parameter of type double. So it calls the method 3.

**ob.test(123.2); // this will invoke test(double)**

**THANKYOU!!**