|  |
| --- |
| // filename: Test2.java  class Test1 {      Test1(int x) {          System.out.println("Constructor called " + x);      }  }    // This class contains an instance of Test1  class Test2 {      Test1 t1 = new Test1(10);        Test2(int i) { t1 = new Test1(i); }        public static void main(String[] args) {           Test2 t2 = new Test2(5);      }  } |

The output of the program is Constructor called 10 Constructor called 5.  
First t2 object is instantiated in the main method. As the order of initialization of local variables comes first than the constructor,first the instance variable (t1), in the class Test2 is allocated to the memory. In this line a new Test1 object is created, constructor is called in class Test1 and ‘Constructor called 10’ is printed. Next the constructor of Test2 is called and again a new object of the class Test1 is created and ‘Constructor called 5’ is printed.

Please write comments if you find any of the answers

|  |
| --- |
| ==================================================================  class Complex {      private double re, im;      public String toString() {          return "(" + re + " + " + im + "i)";      }      Complex(Complex c) {          re = c.re;          im = c.im;      }  }    public class Main {      public static void main(String[] args) {          Complex c1 = new Complex();          Complex c2 = new Complex(c1);          System.out.println(c2);      }  } |

Output: Compiler Error in line “Complex c1 = new Complex();”  
In Java, if we write our own [copy constructor](https://www.geeksforgeeks.org/copy-constructor-in-java/) or parameterized constructor, then compiler doesn’t create the default constructor. This behavior is same as C++.

|  |
| --- |
| public class Main  {      public static void main(String args[])      {          String s1 = "abc";          String s2 = s1;          s1 += "d";          System.out.println(s1 + " " + s2 + " " + (s1 == s2));            StringBuffer sb1 = new StringBuffer("abc");          StringBuffer sb2 = sb1;          sb1.append("d");          System.out.println(sb1 + " " + sb2 + " " + (sb1 == sb2));      }  } //end class |

Output:

abcd abc false

abcd abcd true

**Explanation :**In Java, String is immutable and string buffer is mutable.  
So string s2 and s1 both pointing to the same string abc. And, after making the changes the string s1 points to abcd and s2 points to abc, hence false. While in string buffer, both sb1 and sb2 both point to the same object. Since string buffer are mutable, making changes in one string also make changes to the other string. So both string still pointing to the same object after making the changes to the object (here sb2).

**Program 4:**

**What is the output of the following program?**

|  |
| --- |
| class Helper  {      private int data;      private Helper()      {          data = 5;      }  }  public class Test  {      public static void main(String[] args)      {          Helper help = new Helper();          System.out.println(help.data);      }  } |

a) Compilation error  
b) 5  
c) Runtime error  
d) None of these

A [private constructor](https://www.geeksforgeeks.org/private-constructors-and-singleton-classes-in-java/) cannot be used to initialize an object outside the class that it is defined within because it is no longer visible to the external class.

|  |
| --- |
| public class Test implements Runnable  {      public void run()      {          System.out.printf(" Thread's running ");      }        try      {          public Test()          {              Thread.sleep(5000);          }      }      catch (InterruptedException e)      {          e.printStackTrace();      }        public static void main(String[] args)      {          Test obj = new Test();          Thread thread = new Thread(obj);          thread.start();          System.out.printf(" GFG ");      }  } |

a) GFG Thread’s running  
b) Thread’s running GFG  
c) Compilation error  
d) Runtimer error

**Ans.**(c)  
**Explanation:**A constructor cannot be enclosed inside a try/catch block.

**3) What is the output of the following program?**

|  |
| --- |
| class Temp  {      private Temp(int data)      {          System.out.printf(" Constructor called ");      }      protected static Temp create(int data)      {          Temp obj = new Temp(data);          return obj;      }      public void myMethod()      {          System.out.printf(" Method called ");      }  }    public class Test  {      public static void main(String[] args)      {          Temp obj = Temp.create(20);          obj.myMethod();      }  } |

a) Constructor called Method called  
b) Compilation error  
c) Runtime error  
d) None of the above

**Ans.**(a)  
**Explanation:**When a constructor is marked as private, the only way to create a new object of that class from some external class is using a method that creates a new object, as defined above in the program. The method create() is responsible for creation of Temp object from some other external class. Once the object is created, its method can be invoked from the class in which the object is created.

|  |
| --- |
| ublic class Outer  {      private static int data = 10;      private static int LocalClass()      {          class Inner          {              public int data = 20;              private int getData()              {                  return data;              }          };          Inner inner = new Inner();          return inner.getData();      }        public static void main(String[] args)      {          System.out.println(data \* LocalClass());      }  } |

a) Compilation error  
b) Runtime Error  
c) 200  
d) None of the above

**Ans.**(c)  
**Explanation:**LocalClass() method defines a local inner class. This method creates an object of class Inner and return the value of the variable data that resides within it.

public class Outer

{

private int data = 10;

class Inner

{

private int dat = 20;

private int getData()

{

return dat;

}

public void main(String[] args)

{

Inner inner = new Inner();

System.out.println(inner.getData());

}

}

private int getData()

{

return data;

}

public static void main(String[] args)

{

Outer outer = new Outer();

Outer.Inner inner = outer.new Inner();

System.out.printf("%d", outer.getData());

inner.main(args);

}

}

1020

|  |
| --- |
| public class Outer  {      private int data = 10;        class Inner      {          private int data = 20;          private int getData()          {              return data;          }          public void main(String[] args)          {              Inner inner = new Inner();              System.out.println(inner.getData());            }      }      private int getData()      {          return data;      }      public static void main(String[] args)      {          Outer outer = new Outer();          Outer.Inner inner = outer.new Inner();          System.out.printf("%d", outer.getData());          inner.main(args);      }  } |

1020

a) 2010  
b) 1020  
c) Compilation Error  
d) None of these

**Ans.**(c)  
**Explanation:** Inner class defined above though, have access to the private variable data of the Outer class, but declaring a variable data inside an inner class makes it specific to the Inner class with no conflicts in term of variable declaration. For more see [Shadowing](https://docs.oracle.com/javase/tutorial/java/javaOO/localclasses.html#shadowing-and-local-classes)