Section 1:

1. We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with 2 dimension fields and an abstract method namely 'area'. Now, write 3 classes ‘Rectangle', 'Circle' and ‘Triangle’ derived from the ‘Shape’ class. Implement the area() method as necessary in each class.

Now create another class 'Geometry' containing the main method and create objects of 'Rectangle', 'Square' and 'Circle' and printing their area.

1. Write 2 classes to give examples of function overloading and function overriding. Mention which methods are overloaded and which are overridden.
2. Write class declarations for the class hierarchy given:

Class Hierarchy: Superclass to Subclass

Top to Bottom

1. Take a look at the following code:

class Test4 {

int i;

Test4 (int m){

i = m;

}

public void printVal(){

System.out.println(i);

}

}

class Test5 extends Test4{

private int j;

**// add a constructor correctly**

public void printVal(){

System.out.println(j);

}

public void printSum(){

System.out.println(i+j);

}

}

Write two overloaded constructors for class Test5 – one taking no parameters and other taking 2 integer parameters to set the values of i and j. (Hint: you have to use super as the superclass has a constructor now that takes a parameter). Also write a main function that creates one Test4 object and one Test5 object. Then print the sum using the printSum() function. Also show the output if your code is run.

1. Write a class CountObject which has a method name numberfoObjects() which returns the number of objects been created of that class. Also write a class that will create few objects of CountObject class and show the count. (Hint: you have to use static variable in CountObject class).

# Section 2:

1. What would be the output of the following program:

public class StaticDemo {

            int num1 = 6;

            static int num2 = 10;

      public static void main(String args[]) {

            StaticDemo s1 = new StaticDemo();

            StaticDemo s2 = new StaticDemo();

            s1.num1 = 15;

            s1.num2 = 17;

            s2.num1 = 22;

            s2.num2 = 28;

            System.out.println(s1.num1 + " " + s1.num2 + " " +

s2.num1 + " "+ s2.num2);

   }

}

1. Take a look at the class declaration below:

|  |  |  |
| --- | --- | --- |
| package p1;  class Test8{     int i;     private int j;     protected int k;     public int l;  } | package p2;  import p1.Test8;  class Test9 extends Test8 {  } | package p3;  import p1.Test8;  import p2.Test9;  class TestDemo3{     public static void main(String args[]){             Test8 t8 = new Test8();             Test9 t9 = new Test9();     }  } |

Out of these 4 variables declared in Test8, which are accessible in Test9 directly and by creating Test8 reference? In TestDemo3 main function, which variables can be used by each of the references t8 and t9?

1. What would be the output of the following code:

|  |  |
| --- | --- |
| class Equals {  int i;    boolean equals(Object o){         System.out.println("Overridden");         Equals e = (Equals)o;         if (this.i==e.i)              return true;         else return false;      }  }  class Equals2 {      int i;  } | public class EqualsDemo {      public static void main(String args[]){          Equals e1 = new Equals();          Equals e2 = new Equals();          Equals e4 = new Equals();          e1.i = 10; e2.i = 20; e4.i = 10;          Equals e3 = e2;            Equals2 e6 = new Equals2();          Equals2 e7 = new Equals2();          e6.i = 30; e7.i = 30;          System.out.println(e1.equals(e2));          System.out.println(e2.equals(e3));          System.out.println(e1.equals(e4));          System.out.println(e6.equals(e7));      }  } |

Hint:  if not overridden, the equals() method the Object class is called.

1. What would be the output of the following code:

class Test1 {

private String str = “Test1”;

public void showme(){

System.out.println(str);

}

}

class Test2 extends Test1{

private String str = “Test2”;

public void showme(){

System.out.println(str);

}

}

class Test3 extends Test2 {

private String str = “Test3”;

public void showme(){

System.out.println(str);

}

}

class TestDemo{

public static void main(String args[]){

Test1 t1 = new Test1();

Test2 t2 = new Test2();

Test3 t3 = new Test3();

t1 = t2;

t1.showme();

t1 = t3;

t1.showme();

t2 = t3;

t2.showme();

}

}

1. What would be the output of the following program:

|  |  |
| --- | --- |
| class SuperShow {  String str = "SuperClass";    public void show(){      System.out.println("Super.Show: " + str);      }  } | class ExtendedShow extends SuperShow{      String str = "SubClass";      public void show(){          System.out.println("Sub.Show: " + str);      }  } |
| public class ExtensionDemo {      public static void main(String args[]){          ExtendedShow ext = new ExtendedShow();          SuperShow sup = ext;          sup.show();          ext.show();          System.out.println("sup.str: " + sup.str);          System.out.println("sub.str: " + ext.str);      }  } | |