Roll Number:

The LNM Institute of Information Technology
Mid Semester Exam (OOPs (using Java))

Max Marks: 30 **Max Time: 1.5 Hr.**

Please Note:

A). For the Questions 1-8:

1. Correct syntax errors (if any) in the code snippets and find the outputs.
2. Give proper working before writing the final answer in the answer script.
3. No marks will be given if working is not shown or working is incorrect.
4. Questions 1-6 carry 2 Marks each and questions 7-8 carry 3 marks each.

B). For the questions 9-10.

1. Big syntaxes are not important; the logic and design in Java are more important.
2. Each question carries 6 Marks.

Q1.

```
package main;
class Base
{          public void Print()
          {    System.out.println("Base");     }
}
class Derived extends Base
 {         public void Print()
           {System.out.println("Derived");   }
}
class Main
{          public static void DoPrint( Base o )
           {            o.Print();       }
          public static void main(String[] args)
           {    Base x = new Base();
                Base y = new Derived();
                Derived z = new Derived();
                DoPrint(x);
                DoPrint(y);
                DoPrint(z);
           }
}
```

Q2.

```
public class ObjComp
{
 public static void main(String [] args )
  {
       int result = 0;
       ObjComp oc = new ObjComp();
       Object o = oc;

       if (o == oc)
          result = 1;
       if (o != oc)
          result = result + 10;
       if (o.equals(oc) )
          result = result + 100;
       if (oc.equals(o) )
          result = result + 1000;

        System.out.println("result = " + result);
    }
}
```

Q3.

```
class StringTest
{     public void twist(String[] w)
       {
            String    temp = w[0].substring(0, 1);
              w[0] = w[1].substring(0, 1) + w[0].substring(1);
              w[1] = temp + w[1].substring(1);
       }
      public static void main(String args[])
        {
                    String[] words = {"HOW", "NEAT"};
                    twist(words):
                    System.out.println(words[0] + " " + words[1]);
        }
 }
```

Q4

```
public class X {
   public static void main(String[] args)
   {
      try {
           badMethod();
           System.out.print("A");
         }
       catch (RunTimeException ex)
        {
          System.out.print("B");
        }

       catch (Exception ex)
        {
          System.out.print("C");
        }
      finally
       {
         System.out.print("D");
       }
      System.out.print("E");
   }
   public static void badMethod()
   {
      throw new Error();
   }
}
```

Q5.

```
class Test1
{       Test1(int x)
        {
          System.out.println("Constructor called
"+x);
        }
}
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);
           }
}
```

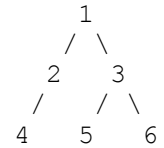
Q6.

```
public class Test
{     static int i = 10;
      public int methodA()
       {
                    i= i+10;
                    return i;
       }
       public static void main(String args[])
           {
                 Test test = new Test();
                  test.methodA();
                  int j = test.methodA();
                 System.out.println("j = "+ j);
           }
}
```

Q7.

```
private  void  treeTraversal(TreeNode root)
{
 if (root == null || (root.getLeft() == null && root.getRight() == null))
        return ;
  else
  {
        treeTraverse(root.getLeft()));
        System.out.println(root.getData());
        treeTraverse(root.getRight()));
        return;
    }
}
```

If **root** refers to the root of the given below tree



What will be the output if the method **treeTraversal(root)** is called from the **main ()** method.

```
class TreeNode
{
      int data;
      TreeNode left, right;
      public TreeNode()
      {
            data =0;
            left = right = null;
      }
     public TreeNode getLeft()
     {
        return left;
     }
     public TreeNode getRight()
     {
        return right;
     }
      public int getData()
      {
          return data;
     }
}
```

Q8.

```
class Person
{  private String name;
     public Person(String name) { this.name = name; }
     public String getName() { return name; }
     public boolean equals(Person other)
     {
          return other != null && name.equals(other.name);
     }
     public int compareTo(Person other)
     {
          return name.compareTo(other.name);
     }
}
class CricketPlayer extends Person
{
     private int totalRuns;
     public CricketPlayer(String name, int n)
     {
          super(name);
          totalRuns = n;
     }
     public int getTotalRuns() { return totalRuns; }
     public void score() { totalRuns++; }
     public int compareTo(CricketPlayer other)
     {
           return getTotalRuns() - other.getTotalRuns();
     }
     public String toString()
     {
          return getName() + "/" + getTotalRuns();
     }
}
public class Test
{

public static void main(String args[])
{
    Person players[] = { new CricketPlayer("Virat Kohali", 6700), new CricketPlayer("Rohit Sharma", 6500)  };

   CricketPlayer cricketplayers[] ={ new CricketPlayer("Virat Kohali", 6700), new CricketPlayer("Rohit Sharma", 6500) };

    System.out.println(players[0].compareTo(players[1]));

    System.out.println(players[0].equals(cricketplayers[1]));

    System.out.println(cricketplayers[0].compareTo(cricketplayers[1]));

 }
}
```

B) Programming Questions[2X6]

Q9. Carefully read this code and write the code indicated in **<Write code block >** to complete the class implementation.

```
abstract class Point
{   protected int x, y;
    public Point(int x, int y)
     {
        this.x = x; this.y = y;
     }
    public int getX() { return x; }
    public int getY() { return y; }
    public abstract Point moveBy(int dx, int dy);
    public abstract double distanceFrom(Point p);
}
class CartesianPoint extends Point
{
    public CartesianPoint(int x, int y)
      { < Write Code > }
    public double distanceFrom(Point p)
      { < Write Code > }
    public  Point moveBy(int dx, int dy);
      { < Write Code > }
    public String toString()
     { return "(" + getX() + "," + getY() + ")";
     }

}
```

```
class Test
{
  public static void main(String args[])
  {
     Point cp= new CartesianPoint(10,5);
     System.out.println(cp);
     cp.moveBy(5,10);
     System.out.println(cp);
     System.out.println(cp.distanceFrom(cp));
   }
  }
```

In the figure 9.1 **(dx, dy)** denote the displacement factor by which the point **(x,y)** will be moved

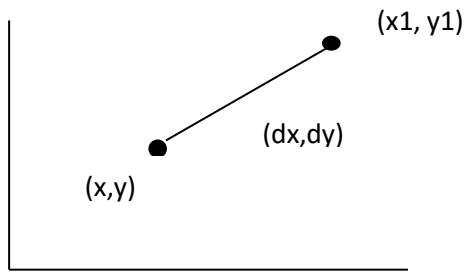


Figure 9.1

Q10. In a Beauty parlor only one service chair is available for various services, but five chairs are available for waiting. Each customer will be assigned a customer-id while entering into shop and will get the services based on first come first serve basis.
Give design and Implement the Java classes to automate the process described above.

```
class PassA
{
    public static void main(String [] args)
    {
        PassA p = new PassA();
        p.start();
    }

    void start()
    {
        long [] a1 = {3,4,5};
        long [] a2 = fix(a1);
        System.out.print(a1[0] + a1[1] + a1[2] + " ");
        System.out.println(a2[0] + a2[1] + a2[2]);
    }

    long [] fix(long [] a3)
    {
        a3[1] = 7;
        return a3;
    }
}
```

```
class Test
{
    public static void main(String [] args)
    {
        Test p = new Test();
        p.start();
    }

    void start()
    {
        boolean b1 = false;
        boolean b2 = fix(b1);
        System.out.println(b1 + " " + b2);
    }

    boolean fix(boolean b1)
    {
        b1 = true;
        return b1;
    }
}
```

```
class PassS
{
    public static void main(String [] args)
    {
        PassS p = new PassS();
        p.start();
    }

    void start()
    {
        String s1 = "slip";
        String s2 = fix(s1);
        System.out.println(s1 + " " + s2);
    }

    String fix(String s1)
    {
        s1 = s1 + "stream";
        System.out.print(s1 + " ");
        return "stream";
    }
}
```