1:
a) Define what is meant by ***Object Oriented Programming*** and describe two ways in which it differs from traditional imperative programming.

Sample Answer
Object-oriented programming may be seen as a collection of cooperating *objects*, as opposed to a traditional view in which a program may be seen as a group of tasks to compute ("subroutines"). In OOP, each object is capable of receiving messages, processing data, and sending messages to other objects.
Each object can be viewed as an independent little machine with a distinct role or responsibility. The actions or "operators" on the objects are closely associated with the object.
For example, in object oriented programming, the data structures tend to carry their own operators around with them (or at least "inherit" them from a similar object or "class"). The traditional approach tends to view and consider data and behavior separately.

b) State whether or not each of the following declarations is legal in JAVA ?

In each case explain your answer.

```
A.   public MyClass {//...}
B.   public protected int myVar;
C.   friendly Button myButton;
D.   Label myLabel;
```

Answer
A – ok
B – wrong, cannot be public and protected
C wrong no friendly modifier
D – OK if Label is a defined class

c)

```
class TClass {
    public static void main (String [] args) {
       int arr[] = new int[args.length];
       float var1 = 0;

       for (int i = 0; i < args.length; i++)
          arr[i] = (new Integer(args[i])).intValue();

       for (int i = 0; i < arr.length; i++)
          var1 = (i%2 == 0) ? 0: arr[i];
       System.out.println( var1/args.length );
    }
 }
```

What would be the output of the above program given the following invocation:

```
> java – classpath . TClass 20 45 60 85 100 205 300 405
```

Explain how you arrived at your answer.

Answer
50.625

The argument strings are each convereted to an integer value.
This program takes the last arg value and if its index is even it assigns 0 to var1 otherwise it assigns the integer equivalent of the last arg value 405 to var1.
It then divides 405 by 8 the no of arguments which yields 50.625

And this is output.

d) Write a program to print out the following:

```
1
1 2
1 2 3
1 2 3 4
……..
1 2 3 4 5 6 7 8 9 10
```

```
// First Program Trianglenos
public class Trianglenos {
   public static void main (String args[]) {
      int i;
      String s1;
      s1 = "1";
      System.out.println( s1 );

      for (i = 2; i < 11; i++) {
          s1 = s1 + " " + i;
        System.out.println( s1 );
      };
   }
}
```

e) In a constructor of a derived class, how do you call the parent's constructor?

Answer
Using super keyword

f) Design an Account Class which has a balance field and three methods

```
i) deposit, ii) withdraw and iii) getBalance
```

(6 marks)

Answer

```
public class Account {
   protected double balance;

   // Constructor to initialize balance
   public Account (double amount) { balance = amount; }

   // Overloaded constructor for empty balance
   public Account () { balance = 0.0; }

   public void deposit (double amount) {
      balance += amount;
   }

   public double withdraw (double amount) {
      // See if amount can be withdrawn
      if (balance >= amount) {
         balance -= amount;
         return amount;
      } else {   // withdraw not allowed
         return 0.0;
      }
   }

    public double getBalance () {
       return balance;
    }
}
```

g) Design and Implement a Savings Account Class which is a subclass of Account and has

i) Extra fields representing a default Interest Rate and a variable Interest Rate

ii) One Constructor which sets the variable interest rate to the default interest rate and the balance to 0 and a second constructor which takes the opening balance and interest rate value as parameters.

ii) Extra method to calculate the monthly interest by multiplying the balance by variable Interest Rate divided by 12; this interest should be added to Account Balance.

Answer

```
class SavingsAccount extends Account {
   //  Default interest rate of 7.95 percent (const)
   private static double defaultInterest = 7.95;

   //  Current interest rate
   private double interestRate;
```

```
    // Overloaded constructor accepting balance and an interest rate
    public SavingsAccount (double amount, double interest) {
        balance = amount;
        interestRate = interest;
    }

    // Overloaded constructor accepting balance with a default interest rate
    public SavingsAccount (double amount) {
        balance = amount;
        interestRate = defaultInterest;
    }

    // Overloaded constructor with empty balance and a default interest rate
    public SavingsAccount () {
        balance = 0.0;
        interestRate = defaultInterest;
    }

     // Add interest to our account
     public void addMonthlyInterest () {
         balance += (balance * interestRate / 100) / 12;
     }
}
```

h) Write a control program which uses the classes in parts b and c to create a Savings account and which

i) Deposits 250 euros

ii) Withdraws 80 euros

iii) Calculates the monthly interest on current Balance

After each action the Account balance should be displayed.

Answer

```
/** SavingsAccountDemo * Demonstration of SavingsAccount class */
class SavingsAccountDemo {
    public static void main (String[] args) {
        // Create an empty SavingsAccount
        SavingsAccount sa = new SavingsAccount();

        // Deposit money
        sa.deposit(250.00);

        // Print current balance
        System.out.println( "Current balance " + sa.getBalance() );

        // Withdraw money
        sa.withdraw( 80.00 );

        // Print remaining balance
        System.out.println( "Remaining balance " + sa.getBalance() );

        sa.addMonthlyInterest();
        System.out.println( "Remaining balance " + sa.getBalance() );
    }
}
```

3: a) Given the following abstract Singer Class:

```
public abstract class Singer {
   private String nameOfSinger;

   // This abstract method is to output the kind of songs the singer sings

   public abstract void sings ();

   public String getSingerName () {
      return nameOfSinger;
   }

   public void setSingerName (String name) {
      nameOfSinger = name;
   }
}
```

Create three subclasses of Singer which describe Jazz Singers, Pop Singers and Opera Singers.

(6 marks)

Answer

```
public class PopSinger extends Singer {
   public void sings () {
      System.out.println( "Sings Pop Songs" );
   }
}

public class JazzSinger extends Singer {
   public void sings () {
      System.out.println( "Sings Jazz songs" );
   }
}


public class OperaSinger extends Singer {
   public void sings () {
      System.out.println( "Sings Classical Songs" );
   }
}
```

b) Write a control program which demonstrates the use of the classes defined in part a above

Answer

```
public class UseSingers {
   public static void main (String[] args) {
      JazzSinger myJazzSinger = new JazzSinger();
      PopSinger myPopSinger = new PopSinger();
```

```
      OperaSinger myOperaSinger = new OperaSinger();
      myJazzSinger.setSingerName( "Murphy" );
      myPopSinger.setSingerName( "Elsie" );
      myOperaSinger.setSingerName( "Sammy" );
      System.out.print( myJazzSinger.getSingerName() + " says " );
      myJazzSinger.sings();
      System.out.print( myPopSinger.getSingerName() + " says " );
      myPopSinger.sings();
      System.out.print( myOperaSinger.getSingerName() + " says " );
      myOperaSinger.sings();
   }
}
```

c) Define what is meant by a java interface;

Answer
An **interface** in the Java programming language is an abstract type that is used to specify an interface (in the generic sense of the term) that classes must implement. Interfaces are declared using the **interface** keyword and may only contain method signatures and constant declarations (variable declarations which are declared to be both static and final).

d)
i) Define a java interface called Professional which has one method signature called work.

Answer

```
public interface Professional {
   public void work();
}
```

ii) Define a User Class called ProfessionalJazzSinger which extends the JazzSinger class defined in a above and which implements the Professional interface.
This class is to have an extra field to represent hours spent training and methods to get and set this.

Answer

```
public class ProfessionalJazzSinger
extends JazzSinger implements Professional {
   private int hoursOfTraining;

   public void setHoursOfTraining (int hrs) {
      hoursOfTraining = hrs;
   }

   public int getHoursOfTraining () {
      return hoursOfTraining;
   }
```

```
   public void work () {
      System.out.println( "I am a JazzSinger who works" );
      System.out.println( "I have " + hoursOfTraining +
         " hours of professional training!" );
   }
}
```

e) Write a control program which demonstrates the use of the ProfessionalJazzSingers class defined in part e above.

Answer

```
public class DemoProfessionalJazzSingers {
   public static void main (String[] args) {
      ProfessionalJazzSinger dizzy = new ProfessionalJazzSinger();
      ProfessionalJazzSinger jazzy = new ProfessionalJazzSinger();
      dizzy.setSingerName( "Simon" );
      jazzy.setSingerName( "Sophie" );
      dizzy.setHoursOfTraining( 40 );
      jazzy.setHoursOfTraining( 300 );

      System.out.println( dizzy.getSingerName() + " says " );
      dizzy.sings();
      dizzy.work();
      System.out.println();

      System.out.println( jazzy.getSingerName() + " says " );
      jazzy.sings();
      jazzy.work();
   }
}

public class DisplayJazzSinger {
   public static void main (String[] args) {
      JazzSinger myJazzSinger = new JazzSinger();
      String JazzSingerString = myJazzSinger.toString();
      System.out.println( JazzSingerString );
   }
}
```

4:

a) ***Courses*** is an array which contains the following DIT course codes:

```
{ DT211, DT249, DT228, DT210 }
```

Write a java program which takes a course code as input and checks to see if that code is in the ***Courses*** array.

Answer

```
import javax.swing.*;
public class SearchList {
   public static void main (String[] args) {
      String[] deptName = { “DT211”, “DT249”, “DT228”, “DT210” };
      boolean codeWasFound = false;
      String code =
         JOptionPane.showInputDialog( null, "Enter a department name" );
      for (int x = 0; x < deptName.length; ++x)
         if (code.equals( deptName[x] ))
            codeWasFound = true;
      if (codeWasFound)
         JOptionPane.showMessageDialog(
            null, code + " was found in the list" );
      else
         JOptionPane.showMessageDialog(
            null, code + " was not found in the list" );
      System.exit( 0 );
   }
}
```

b) Compare and contrast vectors and arrays in java

Answer

- Vectors are much like arrays in that Vectors are
- *homogeneous* - vectors typically store only one kind of element;
- *indexed* - vectors permit clients to access values using integers that indicate position;
- *efficient* - vectors provide fast access to values.

Vectors differ from arrays in that they are *expandable* - you can continue to add elements to a vector, even if the vector has grown beyond its original size.

c) Define a class Person which has fields , first name and surname and defines a toString method.

(5 marks)

Answer

```
public class Person {
   private String firstName;
   Private String surname;

   public Person (String firstName, String surname) {
      this.firstName = firstName;
      this.surname = surname;
   }

   public  String toString () {
      return firstName + " " + surname;
   }
}
```

f) Write a java class which uses vectors to represent a crowd of people as defined by the Person class in part c above

```
import java.util.*;
class Crowd {
    private Vector people;

    public Crowd () {
       people = new Vector();
    }

    public Crowd (int numPersons) {
        people = new Vector( numPersons );
    }

    public boolean add (Person someone) {
        return people.add( someone );
    }

    Person get (int index) {
        return (Person)people.get( index );
    }

    public int size () {
        return people.size();
    }

    public int capacity () {
        return people.capacity();
    }

    public Iterator iterator () {
        return people.iterator();
    }
}
```

g) Write a control program which uses the crowd class and which does the following:

i) Adds five people to the crowd
ii) Outputs the 3rd person in the crowd
iii) Removes the 2nd person from the crowd
iv) Outputs the size of the crowd.

Answer

```
public class CrowdDemo {
   public static void main (String[] args) {
      Crowd crowd1 = new Crowd()
      Person p1 = new Person( "JR", "Ewing" );
      crowd1.add( p1 );
      Person p2 = new Person( "Bobby", "Ewing" );
      crowd1.add( p2 );
      Person p3 = new Person( "SueEllen", "Ewing" );
```

```
        crowd1.add( p3 );
        Person p4 = new Person( “Granny” , “Ewing” );
        crowd1.add( p4 );
        Person p5 = new Person( “Chuck” , “Ewing” );
        crowd1.add( p5 );
        Person p6 = crowd1.get( 2 );
        System.out.println( p6.toString() );
        crowd1.remove( 1 );
        System.out.println( “size is “ + crowd1.size() );
    }
}
```