1. Create arrayList, add the integer elements in arrayList using asList(). Remove the duplicate values and return an arrayList containing unique values. Implement the logic inside removeDuplicates() method. Test the functionalities using the main () method of the Tester class.

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

//firstQue ki jagah tester likhna hai
public class firstQue {
   public static <T> ArrayList<T> removeDuplicates(ArrayList<T> list)
   {

      // Create a new ArrayList
      ArrayList<T> newList = new ArrayList<T>();

      // Traverse through the first list
      for (T element : list) {

         // If this element is not present in newList
         // then add it
         if (!newList.contains(element)) {
```

```
            newList.add(element);
        }
    }

    // return the new list
    return newList;
}

public static void main(String args[]){
    // Get the ArrayList with duplicate values
    ArrayList<Integer> list = new ArrayList<>(Arrays.asList(10,20,10,15,40));

    // Print the Arraylist
    System.out.println("ArrayList with duplicates: "
            + list);

    // Remove duplicates
    ArrayList<Integer>
            newList = removeDuplicates(list);

    // Print the ArrayList with duplicates removed
    System.out.println("ArrayList with duplicates removed: "
```

+ newList);

}

}

2.Given two lists, concatenate the second list in reverse order to the end of the first list and return the concatenated list. Implement the logic inside concatenateLists() method.

listOne = Hello 102 200.8 25

listTwo = 150 40.8 welcome A 0

output: Hello 102 200.8 25 A welcome 40.8 150

```
import java.util.*;
public class Main
{

  static void concentrateList(){
     List<String>list1=new ArrayList<String>();
     list1.add("Hello");
     list1.add("102");
```

```
      list1.add("200.8");
      list1.add("25");
      System.out.println("listone:"+list1);

      List<String>list2=new ArrayList<String>();
      list2.add("150");
      list2.add("40.8");
      list2.add("welcome");
      list2.add("A");
      System.out.println("listtwo:"+list2);

      Collections.reverse(list2);
      System.out.println("Reverse listtwo"+list2);
     StringBuffer sb = new StringBuffer();
     for(String s : list1)
     {
        sb.append(s);
        sb.append(" ");
     }
          for(String s : list2)
     {
        sb.append(s);
        sb.append(" ");
    }

     String str = sb.toString();
     System.out.println(str);
  }
  public static void main(String[]args){
     concentrateList();
}
}
```

3. **Find and return the average salary, number of salaries greater than the average salary and number of salaries lesser than the average salary from the salary array passed to the findDetails() method. Method should return a double array containing average salary in index position 0, number of salaries greater than the average salary in index position 1 and number of salaries lesser than the average salary in index position 2. Implement the logic inside findDetails() method. Test the functionalities using the main method of the Tester class. sample Input: {23500.0 , 25080.0 , 28760.0 , 22340.0 , 19890.0}**

output: Average salary: 23914.0

Number of salaries greater than the average salary: 2.0

Number of salaries lesser than the average salary: 3.0

```
import java.util.*;
public class Tester3 {
    public static double[] lab(){
        int s1 = 23500;
        int s2 = 25080;
        int s3 = 28760;
        int s4 = 22340;
        int s5 = 19890;
        int [] salaries = {s1,s2,s3,s4,s5};
        int sum = 0;
        for(int i=0; i< salaries.length;i++){
            sum = sum +salaries[i];
        }
        double average = sum/ (salaries.length);
        //first value

        int lower_than_avg = 0;
        int higher_than_avg = 0;
        for(int i=0; i< salaries.length;i++){
            if (salaries[i]<average){
                lower_than_avg = lower_than_avg + 1;
            } else if (salaries[i]>average) {
```

```
            higher_than_avg = higher_than_avg + 1;
          }
        }
return new double[0];
}
}
```

4. **Write a Java Program to-**

    b. **Check that given number is palindrome or not**

    c. **Check that given number is odd or even**

    d. **Print reverse of a number**

```
package Practicals;

import java.util.Scanner;


public class Number {

   public static void main(String[] args) {

      Scanner input = new Scanner(System.in);

      int a = input.nextInt();

      Armstrong(a);

      Pallindrome(a);

      EvenorOdd(a);

      Reverse(a);

   }
```

```
 static int Armstrong(int number){
    int num = number , originalnumber , remainder , result=0;
    originalnumber = number;
     System.out.println(originalnumber);
    while(originalnumber !=0){
        remainder = originalnumber % 10;
        result += Math.pow(remainder,3);
        originalnumber /= 10;
    }
    if(result == number){
         System.out.println(number+ " is an Armstrong no.");
    }
    else {
        System.out.println(number + " is not an Armstrong no.");
    }
    return number;
 }
static int Pallindrome (int number){
    int   remainder , originalnumber = number , reversednumber=0;
     while (originalnumber!=0){
         remainder = originalnumber% 10;
         reversednumber = reversednumber * 10 + remainder;
         originalnumber /=10;
```

```
    }
    if(reversednumber == number){
        System.out.println(number + " is a Pallindrome no.");
    }
    else {
        System.out.println(number + " is not a Pallindrome no.");
    }
    return number;
}

static int EvenorOdd(int number){
    int num = number;
    if(num %2==0){
        System.out.println(num +" is Even");
    }
    else {
        System.out.println(num + " is ODD");
    }
    return number;
}
static int Reverse(int number)
{
    int num = number, reversednumber=0, remainder;
```

```
    while(num != 0 ){
        remainder = num % 10;
        reversednumber = reversednumber * 10 + remainder;
        num /=10;
    }
    System.out.println("Reverse number : "+reversednumber);
    return number;
  }
}
```

5. **An educational institution provides stipends for post-graduate students every year. For calculating the stipend, the institution has fixed a base amount of $100 which is provided to all the students. The students who perform exceptionally well during the academics get an extra amount based on their performance. You need to help the institution in developing an application for calculating the stipend by implementing the class using info given below. Note: STIPEND is a final variable.**

calculateTotalStipend():-Calculate and return the total stipend amount based on the aggregate marks of the student using the below table. Implement the getter and setter methods appropriately.

i) Aggregate marks>=85 and <90 then bonus stipend amt is $10

ii) Aggregate marks>=90 and <95 then bonus stipend amt is $15

iii) Aggregate marks>=95 and <100 then bonus stipend amt is $20

sample input: student ID:1212
aggregate marks :93
output: the final stipend of 1212 is $115.0

```
public class FinalStudent {
private int STIPEND = 100 ;
private int studentId;
private int aggregateMarks;
double calculateTotalStipend;

public FinalStudent (){
this.STIPEND = STIPEND;
this.studentId = studentId;
this.aggregateMarks = aggregateMarks;
}


public int getStudentId() {
return studentId;
}
public void setStudentId(int studentId) {
this.studentId = studentId ;
}
public int getSTIPEND(){
return STIPEND;
}
public void setSTIPEND(int STIPEND) {
this.STIPEND = STIPEND;
}
```

```
public double calculateTotalStipend() {
if(getaggregateMarks() >= 85 && getaggregateMarks() < 90) {
return STIPEND +10;
}
else if(getaggregateMarks() >= 90 && getaggregateMarks() < 95) {
return STIPEND +15;
}
else if(getaggregateMarks() >= 95 && getaggregateMarks() < 100) {
return STIPEND +20;
}
else return STIPEND ;
}

public int getaggregateMarks() {
return aggregateMarks;
}

public void setaggregateMarks(int aggregateMarks) {
this.aggregateMarks = aggregateMarks;
}
}

class Tester {
```

```
public static void main(String[] args) {
FinalStudent student1 = new FinalStudent() ;
student1.setStudentId(1212);
student1.setaggregateMarks(93);

double totalStipend = student1.calculateTotalStipend();
System.out.println("The final stipend of " + student1.getStudentId()+" is $" + totalStipend);



}
}
```

6. **Create Java GUI using Swing. When clicking on the login button, it shows the credential in the label as shown in the figure.**

```
package exp6;
import javax.swing.*;
import java.awt.event.*;
public class demo {
public static void main(String[] args) {
JFrame f=new JFrame("User login");

final JButton b=new JButton("Login");
b.setBounds(160,100,100, 30);
final JTextField f1 = new JTextField();
```

```
final JPasswordField f2 = new JPasswordField(16);
f1.setBounds(160, 10, 200, 20);
f2.setBounds(160,60,200,20);
JLabel l1 = new JLabel("Username");
JLabel l2 = new JLabel("Password");
final JLabel l3 = new JLabel(" ");
l1.setBounds(10, 10, 100, 14);
l2.setBounds(10, 60, 100, 14);
l3.setBounds(20, 150, 400, 14);
f.add(b);
f.add(f1);//adding button in JFrame
f.add(l1);
f.add(l2);
f.add(f2);
f.add(l3);
b.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
  l3.setText("Username: " + f1.getText() + " Password: " + f2.getText());
    }
});
f.setSize(400,300);
f.setLayout(null);
f.setVisible(true);
}
}
```

7. Create Java GUI using Swing. When click on order button, it shows order details with bill amount in message box as shown in the figure.

```
package exp6;
import javax.swing.*;
import java.awt.event.*;
public class ques7 extends JFrame implements ActionListener{
    JLabel l;
    JCheckBox cb1,cb2,cb3;
    JButton b;
    ques7(){
        l=new JLabel("Food Ordering System");
        l.setBounds(50,50,300,20);
        cb1=new JCheckBox("Pizza @ 100");
        cb1.setBounds(100,100,150,20);
        cb2=new JCheckBox("Burger @ 30");
        cb2.setBounds(100,150,150,20);
        cb3=new JCheckBox("Tea @ 10");
        cb3.setBounds(100,200,150,20);
        b=new JButton("Order");
        b.setBounds(100,250,80,30);
        b.addActionListener(this);
        add(l);add(cb1);add(cb2);add(cb3);add(b);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
```

```
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    public void actionPerformed(ActionEvent e){
        float amount=0;
        String msg="";
        if(cb1.isSelected()){
            amount+=100;
            msg="Pizza: 100\n";
        }
        if(cb2.isSelected()){
            amount+=30;
            msg+="Burger: 30\n";
        }
        if(cb3.isSelected()){
            amount+=10;
            msg+="Tea: 10\n";
        }
        msg+="-----------------\n";
        JOptionPane.showMessageDialog(this,msg+"Total: "+amount);
    }
    public static void main(String[] args) {
        new ques7();
    }
}
```

8. Write a Java Program to count the number of words in a string using HashMap.

Input: Enter String: " This this is is done by Saket Saket ";

{Saket=2, by=1, this=1, This=1, is=2, done=1}

```
import java.util.HashMap;
import java.util.Map;

public class lab_exam8 {
  public static void main(String[] args) {
     String str = "This this is is done by Saket Saket";
     Map<String, Integer> hashMap = new HashMap<>();
     String[] words = str.split(" ");

     for (String word : words) {
        Integer integer = hashMap.get(word);

        if (integer == null)
           hashMap.put(word, 1);

        else {
           hashMap.put(word, integer + 1);
        }
     }
     System.out.println(hashMap);
  }
}
```

9 .a) Write a program to take the input array element and convert all the elements into next prime numbers and display the changed array.

// Java program to implement

```
// the above approach
import java.util.*;
import java.lang.*;

class Main{

// Function to generate all primes

static ArrayList<Integer> SieveOfEratosthenes(int n)
{
 boolean[] prime = new boolean[2 * n + 1];
 Arrays.fill(prime, true);

 for(int p = 2; p * p <= 2 * n; p++)
 {

        // If p is a prime
        if (prime[p] == true)
        {

                // Mark all its multiples
                // as non-prime
                for(int i = p * p;
                                i <= 2 * n; i += p)
                        prime[i] = false;
        }
 }

 ArrayList<Integer> primes = new ArrayList<>();

 // Store all prime numbers
```

```
 for(int p = 2; p <= 2 * n; p++)
        if (prime[p])
            primes.add(p);

 // Return the list of primes
 return primes;
}

// Function to calculate the
// minimum increments to
// convert every array elements
// to a prime
static int minChanges(int[] arr)
{
 int n = arr.length;
 int ans = 0;

 // Extract maximum element
 // of the given array
 int maxi = arr[0];
 for(int i = 1; i < arr.length; i++)
        maxi = Math.max(maxi, arr[i]);

 ArrayList<Integer> primes = SieveOfEratosthenes(maxi);

 for(int i = 0; i < n; i++)
 {

        // Extract the index which has
        // the next greater prime
        int x = -1;
```

```
        for(int j = 0; j < primes.size(); j++)
        {
            if (arr[i] == primes.get(j))
            {
                x = j;
                break;
            }
            else if (arr[i] < primes.get(j))
            {
                x = j;
                break;
            }
        }

        // Store the difference
        // between the prime and
        // the array element
        int minm = Math.abs(primes.get(x) - arr[i]);

        if (x > 1)
        {
            minm = Math.min(minm,
                            Math.abs(primes.get(x - 1) -
                                     arr[i]));
        }
        ans += minm;
    }
    return ans;
}

// Driver code
```

```
    public static void main (String[] args)

     int[] arr = { 4, 25, 13, 6, 20 };

     System.out.println(minChanges(arr));
    }
    }

    // This code is contributed by offbeat
```

b) Write a java program that takes two positive integers as command-line arguments and prints true if either evenly divides the other.

```
import java.util.*;
public class EquallyDivide {
   Scanner sc = new Scanner(System.in);
   int a,b;
   public void divide(){
      System.out.println("Enter two numbers ");
      a = sc.nextInt();
      b = sc.nextInt();
      if(a%b==0 || b%a==0){
         System.out.println("TRUE");
      }
      else{
         System.out.println("FALSE");
      }
   }
   public static void main(String[] args) {
      EquallyDivide ob = new EquallyDivide();
      ob.divide();
   }
}
```

10.Write a Java Program to iterate ArrayList using for-loop, iterator, and advance for-loop. Insert 3 Array List. Input : 20 30 40

Output: Iterator Loop: 20 30 40 Advanced For Loop: 20 30 40 for Loop: 20 30 40

```
import java.util.ArrayList;
import java.util.Iterator;

public class lab_exam10 {
  public static void main(String[] args) {
    ArrayList<Integer> a1 = new ArrayList<Integer>();
    a1.add(20);
    a1.add(30);
    a1.add(40);
    System.out.println("Using For Loop:");
    for(int i=0;i< a1.size();i++){
      System.out.print(a1.get(i)+" ");
    }

    System.out.println("\nUsing Iterator:");
    Iterator it = a1.iterator();
    while (it.hasNext())
      System.out.print(it.next() + " ");

    System.out.println("\nUsing Advanced For Loop:");
    for (Integer i : a1){
      System.out.print(i+" ");
    }
  }

}
```

-11.a) Write a program that takes three double values X_0, V_0, and t from the user and prints the value $x_0 + v_0t + g\,t^2 / 2,$, where g is the constant 9.78033. This value is the displacement in meters after t seconds when an object is thrown straight up from initial position x0 at velocity v0 meters per second.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
     Scanner sc=new Scanner(System.in);
     double x=sc.nextDouble();
     double v=sc.nextDouble();
     double t=sc.nextDouble();
     double k= (double) (((9.78033)*t*t)/2);
     System.out.println(x + v*t + k);
  }
}
```

b) Write a program that takes two int values m and d from the command line and prints true if day d of month m is between 3/20 and 6/20, false otherwise.

```
import java.util.Scanner;

public class lab_exam11 {
  public static void main(String[] args) {
     Scanner sc=new Scanner(System.in);
     int m = sc.nextInt();
     int d = sc.nextInt();
     Boolean   res =
(((m==3)&&(d>20&&d<=31))||((m==4)&&(d>=1&&d<=30))||((m==5)&&(d>=1&&d<=31))||((
m==6)&&(d>=1&&d<20      )));
     System.out.println("Result=" + res);
  }
}
```

12. **a) Write a program to check if the two strings entered by the user are anagrams or not. Two words are said to be anagrams if the letters of one word can be rearranged to form the other word. For example, jaxa and ajax are anagrams of each other.**

b) Check whether the string is palindrome without using string methods.

```
import java.util.Scanner;
public class lab_exam12
{
  public static void main(String[] args)
  {
     Scanner sc = new Scanner(System.in);
     System.out.println("Please enter string to check palindrome: ");
     String strInput = sc.nextLine();
     char[] chString = strInput.toCharArray();
     // storing reverse string
     String strReverse = "";
     // reading char by char
     for(int a = chString.length - 1; a >= 0; a--)
     {
        strReverse = strReverse + chString[a];
     }
     System.out.println("Given string: " + strInput);
     System.out.println("Reverse String: " + strReverse);
     if(strInput.equals(strReverse))
     {
        System.out.println("string is palindrome.");
     }
     else
     {
        System.out.println("string is not palindrome.");
     }
  }
}
```

13. Write a Java program to calculate the Factorial of an integer number using both iterative and recursive solutions.

-

```
class lab_exam13{
   public static void main(String args[]){
      int i,fact=1;
      int number=5;
      for(i=1;i<=number;i++){
         fact=fact*i;
      }
      System.out.println("Factorial of "+number+" is: "+fact);
   }
}
```

```
class lab_exam13{
   static int factorial(int n){
      if (n == 0)
         return 1;
      else
         return(n * factorial(n-1));
   }
   public static void main(String args[]){
      int i,fact=1;
      int number=4;
      fact = factorial(number);
      System.out.println("Factorial of "+number+" is: "+fact);
   }
}
```

14. Write a program that reads from the user four integers representing the numerators and denominators of two fractions calculates the results of the two fractions and displays the values of the fractions sum, subtraction, multiplication and division. Display output up to two decimal places.

Sample Input: Enter the numerator and denominator of the first fraction: 6 4

Enter the numerator and denominator of the second fraction: 8 5

Output: The sum is: 3.10

The subtraction is: -0.10

The multiplication is: 2.40

The division is: 0.93

```
import java.util.Scanner;
public class lab_exam14 {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      int n1= sc.nextInt();
      int d1= sc.nextInt();
      int n2= sc.nextInt();
      int d2= sc.nextInt();
      float a=n1*d2;
      float b=n2*d1;
      float c=d1*d2;
      float d=n1*n2;
      float e=d1*d2;
      float sum=(a+b)/c;
      float diff=(a-b)/c;
      float multi=d/e;
      float div=a/b;
      System.out.println("The sum is: "+sum);
      System.out.println("The diff is: "+diff);
      System.out.println("The multiplication is: "+multi);
      System.out.println("The division is: "+div);
   }
}
```

15 . Write a program to implement the following inheritance. Accept data for 5 persons and display the name of employee having salary greater than 5000. Class Name: Person Member variables: Name, age Class Name: Employee Member variables: Designation, salary.

```
class Person{
    String name;
    int age;
    Person(int age, String name) {
        this.name = name;
        this.age = age;
    }
}
class Employee extends Person{
    String designation;
    int salary;
    Employee(String designation, String name, int age, int salary) {
        super(age, name);
        this.designation = designation;
        this.salary = salary;
        if (salary>5000){
            System.out.println(name);
        }
    }
}
public class lab15{
    public static void main (String [] args){
        Employee emp = new Employee("Developer","Naman",19,45000);
        Employee emp1 = new Employee("Manager","Karan",25,4000);
        Employee emp2 = new Employee("Accountant","Akash",26,5000);
        Employee emp3 = new Employee("Developer","Yash",22,6000);
```

```
        Employee emp4 = new Employee("Data Scientist","Sahil",23,2000);
    }
}
```

16. Implementing "Multiple Inheritance". Create a two interfaces Account containing methods set () and display () And interface Person containing methods store () and disp(). Derive a class Customer from Person and Account. Accept the name, account number, balance and display all the information related to account along with the interest.

```
// Interface Account
public interface Account {
  void set();
  void display();
}

// Interface Person
public interface Person {
  void store();
  void disp();
}

// Class Customer
public class Customer implements Person, Account {
  private String name;
  private int accountNumber;
  private double balance;

  // Constructor to initialize name, account number, and balance
  public Customer(String name, int accountNumber, double balance) {
```

```
    this.name = name;
    this.accountNumber = accountNumber;
    this.balance = balance;
  }

  // Method to set the account information
  public void set() {
    // Code to set the account information
  }

  // Method to display the account information
  public void display() {
    System.out.println("Name: " + name);
    System.out.println("Account Number: " + accountNumber);
    System.out.println("Balance: " + balance);
    // Code to calculate and display the interest
  }

  // Method to store the person's information
  public void store() {
    // Code to store the person's information
  }

  // Method to display the person's information
  public void disp() {
    // Code to display the person's information
  }
}
Customer c = new Customer("John Smith", 123456, 1000.0);
c.set();
c.display();
```

```
c.store();
c.disp();
```

17. Write a program to read 10 strings from console and then print the sorted strings on console (Use String Class).2) combine two string 3) reverse first string and display it.

```
import java.util.Scanner;
public class lab_exam {
  public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     String str;
     String arr1[]=new String[10];
     for(int i=0;i<10;i++){
        System.out.println("Enter String: "+i);
        str=sc.next();
        arr1[i]=str;
        char arr[] = str.toCharArray();
        char temp;
        for(int a=0;a<arr.length;a++){
           for (int b=a+1;b<arr.length;b++){
              if (arr[b] < arr[a]) {
                 temp = arr[a];
                 arr[a] = arr[b];
                 arr[b] = temp;
              }
           }
        }
        System.out.println(arr);
     }
     System.out.println(arr1[0]+" "+arr1[1]);
     String nstr="";
     char ch;
     for (int i=0; i<arr1[0].length(); i++)
     {
        ch= arr1[0].charAt(i);
        nstr= ch+nstr;
     }
```

```
        System.out.println("Reversed First String: "+ nstr);

    }
}
```

18. Write a program, to implement the following hierarchy. Displays information of each class the rectangle represents the classes. The classes Movie and MusicVideo inherit all the members of the class VideoTape.

```
class VideoTape {
    private String title;
    private int length;

    public VideoTape(String title, int length) {
        this.title = title;
        this.length = length;
    }

    public String getTitle() {
        return title;
    }

    public int getLength() {
        return length;
    }

    @Override
    public String toString() {
        return "VideoTape: " + title + " (" + length + " minutes)";
    }
}

class Movie extends VideoTape {
    private String rating;

    public Movie(String title, int length, String rating) {
        super(title, length);
        this.rating = rating;
    }
```

```
    public String getRating() {
        return rating;
    }

    @Override
    public String toString() {
        return "Movie: " + getTitle() + " (" + getLength() + " minutes, rated " + rating + ")";
    }
}

class MusicVideo extends VideoTape {
    private String artist;

    public MusicVideo(String title, int length, String artist) {
        super(title, length);
        this.artist = artist;
    }

    public String getArtist() {
        return artist;
    }

    @Override
    public String toString() {
        return "MusicVideo: " + getTitle() + " (" + getLength() + " minutes, by " + artist + ")";
    }
}

// Test the classes
VideoTape tape1 = new VideoTape("The Secret Life of Pets", 90);
System.out.println(tape1);  // prints "VideoTape: The Secret Life of Pets (90 minutes)"

Movie movie1 = new Movie("Jurassic Park", 127, "PG-13");
System.out.println(movie1);  // prints "Movie: Jurassic Park (127 minutes, rated PG-13)"

MusicVideo musicVideo1 = new MusicVideo("Roar", 3, "Katy Perry");
System.out.println(musicVideo1);  // prints "MusicVideo: Roar (3 minutes, by Katy Perry)"
```

19. Create two arrayLists ,add the String elements in arrayList using add().

1) **Add one arraylist into the other from index 1 using appropriate method.**
2) **Print the two added list.**
3) **Print the index of "Are".**
4) **Remove the 4th element from arraylist1**
5) **Replace 4 element of arraylist2 as "U"**

Test the functionalities using the main() method of the Tester class.

Sample Input: str1 - ("Hello", "How", "Are", "You") str2 - ("Hi" , "How", "Are" ,You")

Sample Output: [Hello, Hi, How, Are, You, How, Are, You]

```
import java.util.*;
public class lab_exam19 {
   public static void main(String[] args) {
      ArrayList<String> str1=new ArrayList<>();
      str1.add("Hello");
      str1.add("How");
      str1.add("Are");
      str1.add("You");

      ArrayList<String> str2=new ArrayList<>();
      str2.add("Hi");
      str2.add("How");
      str2.add("Are");
      str2.add("You");

      ArrayList<String> str3=new ArrayList<>();
      str3.addAll(str1);
      str3.addAll(1,str2);
      System.out.println(str3);

      System.out.println("The index of Are:"+str3.indexOf("Are"));
```

```
        str1.remove(3);
        System.out.println(str1);

        str2.set(3,"U");
        System.out.println(str2);
    }
}
```

20. Create a new class Order in the Java project with the instance variables and methods mentioned below. orderId: int , orderedFoods: String, totalPrice: double, status: String , calculateTotalPrice(int unitPrice): double. Calculate the total price by applying a service charge of 5% on the food item ordered and store it in the instance variable totalPrice. Return the calculated total price. Create an object of the Order class, initialize the instance variables using parameterized constructor, invoke the calculateTotalPrice() method and display the values of the instance variables in the main() method of the Tester class. Use static block to print status "Ordered".

Sample Output: Order Details:

Order Id: 101

Ordered Food: Burger

Order status: Ordered

Total Price: 35

```
class Order1{
    private int orderId;
    private String orderedFoods;
    private double totalPrice;
    private String status;

    public Order1(int orderId, String orderedFoods) {
        this.orderId = orderId;
        this.orderedFoods = orderedFoods;
```

```
  }

  public int getOrderId() {
     return orderId;
  }

  public String getOrderedFoods() {
     return orderedFoods;
  }

  double calculateTotalPrice(int unitPrice){
     totalPrice = (unitPrice+(0.05*unitPrice));
     return totalPrice;
  }
  static {
     System.out.println("Order Status: Ordered");
  }
}

public class lab_exam20 {
  public static void main(String[] args) {
     System.out.println("Order Details");
     Order1 order1=new Order1(101,"Burger");
     System.out.println("Order ID:" + order1.getOrderId());
     System.out.println("Ordered Food: "+order1.getOrderedFoods());
     System.out.println("Total Price "+order1.calculateTotalPrice(33));

  }
}
```

21.Create GUI using Swing. As shown in the figure. Write a program to print the text entered in the text field-1 into text field-2 after button click. Display entered text in label below button also.

```
import javax.swing.*;
import java.awt.event.*;
public class GUI3 {
   public static void main(String args[]){
```

```
        JFrame f = new JFrame("Text");
        JTextField t = new JTextField(" ");
        JTextField t2 = new JTextField(" ");
        t.setBounds(120,20,150,25);
        t2.setBounds(120,60,150,25);
        JButton b = new JButton("SUBMIT");
        JLabel l = new JLabel(" ");
        b.setBounds(135,100, 100, 20);
        l.setBounds(170,150,150,25);
        f.add(t);
        f.add(t2);
        f.add(b);
        f.add(l);
        f.setSize(400,300);
        f.setLayout(null);
        f.setVisible(true);
        b.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                t2.setText(t.getText());
                l.setText(t.getText());
            }
        });

    }
}
```

22. Write a program to handle the custom exception when the number entered by user through keyboard is odd. Use throw and throws keywords. Exception name is OddNumberException. If the number is odd print message "Number is Odd" else print "Number is Even".

```
import java.util.Scanner;
class OddNumberException extends Exception{
    @Override
    public String getMessage(){
        return "Number is Odd";
    }
```

```
}

public class lab_exam22 {
  public static void main(String[] args) throws OddNumberException{
     Scanner sc=new Scanner(System.in);
     OddNumberException oddNumberException = new OddNumberException();
     System.out.println("Enter a number");
     int a= sc.nextInt();
     if(a%2!=0){
        try {
           throw new OddNumberException();
        }
        catch (Exception e){
           System.out.println(oddNumberException.getMessage());
        }
     }
     else {
        System.out.println("Number is Even");
     }
  }
}
```

23. Calculate and return the sum of all the even numbers present in the numbers array passed to the method calculateSumOfEvenNumbers. Implement the logic inside calculateSumOfEvenNumbers() method. Test the functionalities using the main() method of the Tester class.

```
public class lab_exam23 {
  static int calculateSumOfEvenNumbers(int array[]){
     int sum=0;
     for(int i=0;i<array.length;i++){
        if(array[i]%2==0){

           sum=sum+array[i];
        }
     }
     return sum;
```

```
  }
  public static void main(String[] args) {
     int arr[]={1,2,3,4,5,6,7,8,9,10};
     System.out.println(calculateSumOfEvenNumbers(arr));

  }
}
```

24. Two classes - Camera and DigitalCamera are provided to you. DigitalCamera extends Camera class. Both classes have their parameterized constructors.

Camera Class: private String brand; private double cost;

DigitalCamera Class: private int memory;

Create a instance of child class and display the output as shown below.

Sample Output: Nikon, 100$, 16GB

```
class Camera{
  private String brand;
  private double cost;

  public Camera(String brand, double cost) {
     this.brand = brand;
     this.cost = cost;
  }
}

class DigitalCamera extends Camera
  {  private int memory;
  public DigitalCamera(String brand, double cost, int memory) {
     super(brand, cost);
     this.memory = memory;
     System.out.println("The brand is: "+brand);
     System.out.println("The cost is: "+cost+"$");
     System.out.println("The memory is: "+memory+"GB");
```

```
  }
}


public class lab_exam24 {
  public static void main(String[] args) {
    DigitalCamera digitalCamera = new DigitalCamera("Nikon",100,16);
  }
}
```

25. Create Calculator GUI using Swing. Add buttons for numbers 0-9 , operators +, -, x, /, =, AC. Display output of addition and subtraction operations in textbox.

```
import java.awt.event.*;

import javax.swing.*;

class calculator extends JFrame implements ActionListener {

    // create a frame

    static JFrame f;

    // create a textfield

    static JTextField l,l2;


    // store operator and operands

    String s0, s1, s2;


    // default constructor

    calculator()

    {

        s0 = s1 = s2 = "";

    }
```

```
// main function
public static void main(String args[])
{
    // create a frame
    f = new JFrame("calculator");
    l2 = new JTextField("Calculator");



    // create a object of class
    calculator c = new calculator();

    // create a textfield
    l = new JTextField(16);

    // set the textfield to non editable
    l.setEditable(false);

    // create number buttons and some operators
    JButton b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, ba, bs, bd, bm, be, beq, beq1;

    // create number buttons
    b0 = new JButton("0");
    b1 = new JButton("1");
```

```
b2 = new JButton("2");
b3 = new JButton("3");
b4 = new JButton("4");
b5 = new JButton("5");
b6 = new JButton("6");
b7 = new JButton("7");
b8 = new JButton("8");
b9 = new JButton("9");

// equals button
beq1 = new JButton("=");

// create operator buttons
ba = new JButton("+");
bs = new JButton("-");
bd = new JButton("/");
bm = new JButton("*");
beq = new JButton("C");

// create . button
be = new JButton(".");
// create a panel
JPanel p = new JPanel();

// add action listeners
```

```
bm.addActionListener(c);
bd.addActionListener(c);
bs.addActionListener(c);
ba.addActionListener(c);
b9.addActionListener(c);
b8.addActionListener(c);
b7.addActionListener(c);
b6.addActionListener(c);
b5.addActionListener(c);
b4.addActionListener(c);
b3.addActionListener(c);
b2.addActionListener(c);
b1.addActionListener(c);
b0.addActionListener(c);
be.addActionListener(c);
beq.addActionListener(c);
beq1.addActionListener(c);


// add elements to panel
p.add(l2);
p.add(l);
p.add(b1);
p.add(b2);
p.add(b3);
```

```
    p.add(b4);
    p.add(b5);
    p.add(b6);
    p.add(b7);
    p.add(b8);
    p.add(b9);
    p.add(b0);
    p.add(ba);
    p.add(bs);
    p.add(bd);
    p.add(bm);
    p.add(beq);
    p.add(beq1);

    // set Background of panel



    // add panel to frame
    f.add(p);

    f.setSize(200, 250);
    f.setVisible(true);
    f.setLayout(null);
}
public void actionPerformed(ActionEvent e)
```

```
{
    String s = e.getActionCommand();

    // if the value is a number
    if ((s.charAt(0) >= '0' && s.charAt(0) <= '9') || s.charAt(0) == '.') {
        // if operand is present then add to second no
        if (!s1.equals(""))
            s2 = s2 + s;
        else
            s0 = s0 + s;

        // set the value of text
        l.setText(s0 + s1 + s2);
    }
    else if (s.charAt(0) == 'C') {
        // clear the one letter
        s0 = s1 = s2 = "";

        // set the value of text
        l.setText(s0 + s1 + s2);
    }
    else if (s.charAt(0) == '=') {
        double te;
        // store the value in 1st
        if (s1.equals("+"))
```

```
        te = (Double.parseDouble(s0) + Double.parseDouble(s2));
    else if (s1.equals("-"))
        te = (Double.parseDouble(s0) - Double.parseDouble(s2));
    else if (s1.equals("/"))
        te = (Double.parseDouble(s0) / Double.parseDouble(s2));
    else
        te = (Double.parseDouble(s0) * Double.parseDouble(s2));

    // set the value of text
    l.setText(s0 + s1 + s2 + "=" + te);

    // convert it to string
    s0 = Double.toString(te);

    s1 = s2 = "";
}
else {
    // if there was no operand
    if (s1.equals("") || s2.equals(""))
        s1 = s;
        // else evaluate
    else {
        double te;

        // store the value in 1st
```

```
            if (s1.equals("+"))
                te = (Double.parseDouble(s0) + Double.parseDouble(s2));
            else if (s1.equals("-"))
                te = (Double.parseDouble(s0) - Double.parseDouble(s2));
            else if (s1.equals("/"))
                te = (Double.parseDouble(s0) / Double.parseDouble(s2));
            else
                te = (Double.parseDouble(s0) * Double.parseDouble(s2));

            // convert it to string
            s0 = Double.toString(te);
            // place the operator
            s1 = s;
 // make the operand bla
  s2 = "";
          }
          // set the value of text
          l.setText(s0 + s1 + s2);
    }
    }
}
```

26.1) Write a Java program to find the maximum and minimum value of an array

```
4public class lab_exam26 {
   public static void main(String[] args) {
```

```
        int array[]= {23, 92, 56, 39, 93};
        int max = 0;

        for(int i=0; i<array.length; i++ ) {
            if(array[i]>max) {
                max = array[i];
            }
        }
        int min = array[0];

        for(int i=0; i<array.length; i++ ) {
            if(array[i]<min) {
                min = array[i];
            }
        }
        System.out.println("Maximum value: "+max);
        System.out.println("Minimum value: "+min);
    }
}
```

26. 2) Write a Java program to find the second largest element in an array.

```
import java.util.Arrays;
```

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {5, 2, 7, 1, 8, 3};

        // Sort the array in non-descending order
        Arrays.sort(arr);

        // Return the second last element of the sorted array
        System.out.println("Second largest element: " + arr[arr.length - 2]);
    }
}
```

26. 3) Take 10 integer inputs from user and store them in an array. Now, copy all the elements in another array but in reverse order.

```
import java.util.Scanner;

public class lab_exam26 {
   public static void main(String[] args) {
       Scanner sc=new Scanner(System.in);
       int arr[] =new int[10];
```

```
    System.out.println("Enter 10 numbers");
    for(int i=0;i<10;i++){
        arr[i]=sc.nextInt();
    }

    int arr1[]=new int[10];
    for (int j=0;j<10;j++){
        arr1[j]=arr[10-j-1];
    }

    for(int k=0;k<10;k++){
        System.out.print(arr1[k]);
    }
  }
}
```

27 . 1) public class SubarraySum {

```
    /* Returns true if the there is a
subarray of arr[] with a sum equal to
       'sum' otherwise returns false.
Also, prints the result */
    void subArraySum(int arr[], int n, int sum)
    {
        // Pick a starting point
```

```
        for (int i = 0; i < n; i++) {
            int currentSum = arr[i];

            if (currentSum == sum) {
                System.out.println("Sum found at indexe "
                                   + i);
                return;
            }
            else {
                // Try all subarrays starting with 'i'
                for (int j = i + 1; j < n; j++) {
                    currentSum += arr[j];

                    if (currentSum == sum) {
                        System.out.println(
                            "Sum found between indexes " + i
                            + " and " + j);
                        return;
                    }
                }
            }
```

```
        }

        System.out.println("No subarray found");

        return;

    }


    public static void main(String[] args)

    {

        SubarraySum arraysum = new SubarraySum();

        int arr[] = { 15, 2, 4, 8, 9, 5, 10, 23 };

        int n = arr.length;

        int sum = 23;

        arraysum.subArraySum(arr, n, sum);

    }

}
```

// Java program to count occurrences

```
// of an element


class Main

{
```

```
    // Returns number of times x occurs in arr[0..n-1]
    static int countOccurrences(intarr[], int n, int x)
    {
        int res = 0;
        for (int i=0; i<n; i++)
            if (x == arr[i])
              res++;
        return res;
    }

    public static void main(String args[])
    {
        int arr[] = {1, 2, 2, 2, 2, 3, 4, 7 ,8 ,8 };
        int n = arr.length;
        int x = 2;
        System.out.println(countOccurrences(arr, n, x));
    }
}
```

28. Create a GUI that has two buttons on it. When clicked, each button creates a new window. The first button creates a window that has a single button on it. Pressing that button will change the window's background color back and forth between red and green. The second button creates a window that has two buttons. Pressing the first button of these two buttons will change the window's background color to black. Pressing the second button will change the background color to white. Make sure that your windows will just dispose of themselves when they are closed.

```
            this.getContentPane().setBackground(Color.RED);

            isRed = true;

          }

          this.setForeground(Color.RED);

        } else

          System.out.println("Error in  red/green events");

      }

  }

// The second kind of window

  private class BlackWhiteWindow extends JFrame

        implements ActionListener {

    private JButton blackButton;

    private JButton whiteButton;

    public BlackWhiteWindow(){

      super();

      setSize(FLOAT_WIDTH, FLOAT_HEIGHT);
```

```
        setTitle("A Black/White Window");
        setLayout(new BorderLayout());
        setForeground(Color.GRAY);
        blackButton = new JButton("Black");
        blackButton.addActionListener(this);
        add(blackButton, BorderLayout.NORTH);
        whiteButton = new JButton("White");
        whiteButton.addActionListener(this);
        add(whiteButton, BorderLayout.SOUTH);
        FloatingWindowHandler listener = new FloatingWindowHandler(this);
        addWindowListener(listener);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("White")) {
//System.out.println("White button pressed");
//System.out.println("Change applied to " + this);
            this.getContentPane().setBackground(Color.WHITE);
        } else if (e.getActionCommand().equals("Black")) {
//System.out.println("White button pressed");
//System.out.println("Change applied to " + this);
```

```
            this.getContentPane().setBackground(Color.BLACK);
        }  else
            System.out.println("Error in  Black/White events");
      }

  }

}
```

28. Create a GUI that has two buttons on it. When clicked, each button creates a new window. The first button creates a window that has a single button on it. Pressing that button will change the window's background color back and forth between red and green. The second button creates a window that has two buttons. Pressing the first button of these two buttons will change the window's background color to black. Pressing the second button will change the background color to white. Make sure that your windows will just dispose of themselves when they are closed.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.*;
public class Exercise6
{import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.*;
public class Exercise6
{
  public static void main(String[] args) {
     MultiWindow window1 = new MultiWindow();
     window1.setVisible(true);
  }
}

class MultiWindow extends JFrame implements ActionListener {

  public static final int WIDTH = 400;
  public static final int HEIGHT = 300;
  public static final int FLOAT_WIDTH = 500;
  public static final int FLOAT_HEIGHT = 150;
```

```
    private JButton floatOneButton;
    private JButton floatTwoButton;
    /**
     * Creates a new instance of MultiWindow
     */
    public MultiWindow() {
        super();
// This window is sized so that it will not be hidden by the floating window
// that is created
        setSize(WIDTH, HEIGHT);
        setTitle("Multiple windows Demo");
        setLayout(new BorderLayout());
        floatOneButton = new JButton("Create Window Type 1");
        floatTwoButton = new JButton("Create Window Type 2");
        floatOneButton.addActionListener(this);
        floatTwoButton.addActionListener(this);
        getContentPane().add(floatOneButton, BorderLayout.EAST);
        getContentPane().add(floatTwoButton, BorderLayout.WEST);
//setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
//        WindowDestroyer listener = new WindowDestroyer();
//
```

```
//        addWindowListener(listener);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==floatOneButton) {
            RedGreenWindow x = new RedGreenWindow();
            x.setVisible(true);
        } else if (e.getSource()==floatTwoButton) {
            BlackWhiteWindow x = new  BlackWhiteWindow();
            x.setVisible(true);
        } else
            System.out.println("Error in interface.");
    }
    private class FloatingWindowHandler extends WindowAdapter {
        private JFrame myWindow;
        public FloatingWindowHandler(JFrame aWindow){
            myWindow = aWindow;
        }
        public void windowClosing(WindowEvent e) {
            myWindow.dispose();
        }
```

```
  }
// The first kind of window
  private class RedGreenWindow extends JFrame
          implements ActionListener {
      private JButton colorButton;
      private boolean isRed = true;
      public RedGreenWindow(){
          super();
          setSize(FLOAT_WIDTH, FLOAT_HEIGHT);
          setTitle("A Red/Green Window");
          setLayout(new BorderLayout());
          setForeground(Color.RED);
          colorButton = new JButton("Change");
          colorButton.addActionListener(this);
          add(colorButton, BorderLayout.WEST);
          FloatingWindowHandler listener = new FloatingWindowHandler(this);
          addWindowListener(listener);
      }
      public void actionPerformed(ActionEvent e) {
          if (e.getActionCommand().equals("Change")) {
```

```
//System.out.println("Change button pressed");

//System.out.println("Change applied to " + this);

            if(isRed){

               this.getContentPane().setBackground(Color.GREEN);

               isRed = false;

            } else {

   public static void main(String[] args) {
      MultiWindow window1 = new MultiWindow();
      window1.setVisible(true);
   }
}

class MultiWindow extends JFrame implements ActionListener {

   public static final int WIDTH = 400;
   public static final int HEIGHT = 300;
   public static final int FLOAT_WIDTH = 500;
   public static final int FLOAT_HEIGHT = 150;

   private JButton floatOneButton;

   private JButton floatTwoButton;

   /**

    * Creates a new instance of MultiWindow

    */

   public MultiWindow() {

      super();

// This window is sized so that it will not be hidden by the floating window
```

```
// that is created
        setSize(WIDTH, HEIGHT);
        setTitle("Multiple windows Demo");
        setLayout(new BorderLayout());
        floatOneButton = new JButton("Create Window Type 1");
        floatTwoButton = new JButton("Create Window Type 2");
        floatOneButton.addActionListener(this);
        floatTwoButton.addActionListener(this);
        getContentPane().add(floatOneButton, BorderLayout.EAST);
        getContentPane().add(floatTwoButton, BorderLayout.WEST);
//setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
//        WindowDestroyer listener = new WindowDestroyer();
//
//        addWindowListener(listener);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==floatOneButton) {
            RedGreenWindow x = new RedGreenWindow();
            x.setVisible(true);
        } else if (e.getSource()==floatTwoButton) {
            BlackWhiteWindow x = new  BlackWhiteWindow();
```

```
            x.setVisible(true);
        } else
            System.out.println("Error in interface.");
    }
    private class FloatingWindowHandler extends WindowAdapter {
        private JFrame myWindow;
        public FloatingWindowHandler(JFrame aWindow){
            myWindow = aWindow;
        }
        public void windowClosing(WindowEvent e) {
            myWindow.dispose();
        }
    }
// The first kind of window
    private class RedGreenWindow extends JFrame
            implements ActionListener {
        private JButton colorButton;
        private boolean isRed = true;
        public RedGreenWindow(){
            super();
```

```
        setSize(FLOAT_WIDTH, FLOAT_HEIGHT);
        setTitle("A Red/Green Window");
        setLayout(new BorderLayout());
        setForeground(Color.RED);
        colorButton = new JButton("Change");
        colorButton.addActionListener(this);
        add(colorButton, BorderLayout.WEST);
        FloatingWindowHandler listener = new FloatingWindowHandler(this);
        addWindowListener(listener);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Change")) {
//System.out.println("Change button pressed");
//System.out.println("Change applied to " + this);
            if(isRed){
                this.getContentPane().setBackground(Color.GREEN);
                isRed = false;
            } else {
                this.getContentPane().setBackground(Color.RED);
                isRed = true;
```

```
            }
            this.setForeground(Color.RED);
        } else
            System.out.println("Error in  red/green events");
    }
  }
// The second kind of window
  private class BlackWhiteWindow extends JFrame
        implements ActionListener {
    private JButton blackButton;
    private JButton whiteButton;
    public BlackWhiteWindow(){
        super();
        setSize(FLOAT_WIDTH, FLOAT_HEIGHT);
        setTitle("A Black/White Window");
        setLayout(new BorderLayout());
        setForeground(Color.GRAY);
        blackButton = new JButton("Black");
        blackButton.addActionListener(this);
        add(blackButton, BorderLayout.NORTH);
```

```
        whiteButton = new JButton("White");
        whiteButton.addActionListener(this);
        add(whiteButton, BorderLayout.SOUTH);
        FloatingWindowHandler listener = new FloatingWindowHandler(this);
        addWindowListener(listener);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("White")) {
//System.out.println("White button pressed");
//System.out.println("Change applied to " + this);
            this.getContentPane().setBackground(Color.WHITE);
        } else if (e.getActionCommand().equals("Black")) {
//System.out.println("White button pressed");
//System.out.println("Change applied to " + this);
            this.getContentPane().setBackground(Color.BLACK);
        }  else
            System.out.println("Error in  Black/White events");
    }
  }
}
```

29. **Create an abstract class 'Bank' with an abstract method 'getBalance'. $100, $150 and $200 are deposited in banks A, B and C respectively. 'BankA', 'BankB' and 'BankC' are subclasses of class 'Bank', each having a method named 'getBalance'. Call this method by creating an object of each of the three classes.**

```
abstract class Bank{
  abstract void getBalance();
}

class BankA extends Bank{
  public void getBalance(){
     System.out.println("The balance in bank A is 100");
  }
}

class BankB extends Bank{
  public void getBalance(){
     System.out.println("The balance in bank B is 150");
  }
}

class BankC extends Bank{
  public void getBalance(){
     System.out.println("The balance in bank C is 200");
  }
}

public class lab_exam29 {
  public static void main(String[] args) {
     BankA bankA = new BankA();
     bankA.getBalance();

     BankB bankB=new BankB();
     bankB.getBalance();

     BankC bankC = new BankC();
     bankC.getBalance();
```

```
  }
}
```

30 .All the banks operating in India are controlled by RBI. RBI has set a well defined guideline (e.g. minimum interest rate, minimum balance allowed, maximum withdrawal limit etc) which all banks must follow. For example, suppose RBI has set minimum interest rate applicable to a saving bank account to be 4% annually; however, banks are free to use 4% interest rate or to set any rates above it.

Write a JAVA program to implement bank functionality in the above scenario and demonstrate the dynamic polymorphism concept. Note: Create few classes namely Customer, Account, RBI (Base Class) and few derived classes (SBI, ICICI, PNB etc). Assume and implement required member variables and functions in each class.

Hint: Class Customer {
//Personal Details ... // Few functions ...
}
Class Account {
// Account Detail ... // Few functions ...
}
Class RBI { Customer c; //hasA relationship Account a;
//hasA relationship ..
Public double GetInterestRate() { }
Public double GetWithdrawalLimit() { }
}
Class SBI: public RBI {
//Use RBI functionality or define own functionality.
}
Class ICICI: public RBI { //Use RBI functionality or define own functionality. }

```
// Customer class to store personal details of the customer

class Customer {

  private String name;

  private String address;

  private long phoneNumber;

  private String email;
```

```
public Customer(String name, String address, long phoneNumber, String email) {
  this.name = name;
  this.address = address;
  this.phoneNumber = phoneNumber;
  this.email = email;
}

public String getName() {
  return this.name;
}

public String getAddress() {
  return this.address;
}

public long getPhoneNumber() {
  return this.phoneNumber;
}
```

```
  public String getEmail() {
    return this.email;
  }
}

// Account class to store account details and perform account-related actions
class Account {
  private long accountNumber;
  private double balance;
  private Customer customer;

  public Account(long accountNumber, double balance, Customer customer) {
    this.accountNumber = accountNumber;
    this.balance = balance;
    this.customer = customer;
  }

  public long getAccountNumber() {
    return this.accountNumber;
  }
```

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

public Customer getCustomer() {
  return this.customer;
}

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

public void withdraw(double amount) {
  if (amount > this.balance) {
    System.out.println("Insufficient balance");
  } else {
    this.balance -= amount;
  }
}
```

```
}

// RBI class to set guidelines for banks and perform RBI-related actions
class RBI {
  protected double minimumInterestRate;
  protected double minimumBalance;
  protected double maximumWithdrawalLimit;

      public RBI(double minimumInterestRate, double minimumBalance, double maximumWithdrawalLimit) {
    this.minimumInterestRate = minimumInterestRate;
    this.minimumBalance = minimumBalance;
    this.maximumWithdrawalLimit = maximumWithdrawalLimit;
  }

  public double getMinimumInterestRate() {
    return this.minimumInterestRate;
  }

  public double getMinimumBalance() {
```

```
    return this.minimumBalance;
  }

  public double getMaximumWithdrawalLimit() {
    return this.maximumWithdrawalLimit;
  }

  public double calculateInterest(double balance) {
    return balance * this.minimumInterestRate;
  }
}

// SBI class that extends the RBI class and performs SBI-specific actions
class SBI extends RBI {
        public SBI(double minimumInterestRate, double minimumBalance, double
maximumWithdrawalLimit) {
    super(minimumInterestRate, minimumBalance, maximumWithdrawalLimit);
  }

  public void applyForLoan(double loanAmount) {
```

```
    System.out.println("Loan application submitted for SBI");
  }
}

// ICICI class that extends the RBI class and performs ICICI-specific actions
class ICICI extends RBI {
    public ICICI(double minimumInterestRate, double minimumBalance, double maximumWithdrawalLimit) {
    super(
```