SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

**(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY, CHENNAI)
COIMBATORE – 641008**

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

21CS302
JAVA PROGRAMMING LABORATORY

CONTINUOUS ASSESSMENT RECORD

Submitted by

Name: ……………………………..……………..

Register No. : ………………………………..………

Degree & Branch: ……………….……………..…

Class & Semester: ……………….…………….…

Academic Year: …………….…………………….……….

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY, CHENNAI)
COIMBATORE – 641008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

21CS302
JAVA PROGRAMMING LABORATORY

Continuous Assessment Record

Submitted by

Name : ……..…….……………………..………… **Register No. : ....……………………**

Class/Semester :…………………………………. **Degree & Branch: ……………………………..**

BONAFIDE CERTIFICATE

This is to certify that this record is the bonafide record of work done by Mr./Ms. ____________________ during the academic year 2022 – 2023.

Faculty In-charge **Head of the Department**

Submitted for the University practical examination held on ______________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY, CHENNAI)
COIMBATORE – 641008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

21CS302
JAVA PROGRAMMING LABORATORY

Record of laboratory work

EVEN SEMESTER - 2022-2023

Name of the Faculty	**Mr.S.Karthikeyan** **Mrs.N.Subhashini**

CONTINUOUS EVALUATION SHEET

REFERENCES RUBRICS TABLE

Criteria	Range of Marks			
	Excellent	Good	Average	Below Average
Objective & Description with sample data (20)	18-20	17-18	15-16	0-14
Formation of Program (30)	27-30	21-26	15-20	0-14
Execution and Testing (30)	27-30	21-26	15-20	0-14
Documentation (10)	9-10	7-8	5-6	0-4
Viva (10)	9-10	7-8	5-6	0-4
Overall Marks	**90-100**	**70-89**	**50-69**	**0-49**

INDEX

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

**(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY, CHENNAI)
COIMBATORE – 641008**

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

21CS302 – JAVA PROGRAMMING LABORATORY

EVEN SEMESTER: 2022-2023

Components	EX1	EX2	EX3	EX4	EX5	EX6	EX7	EX8	EX9
Objective & Algorithm (20 marks)									
Program with Proper Syntax and Structure (30 Marks)									
Compilation and Debugging (30 Marks)									
Documentation (10 Marks)									
Viva (10 Marks)									
TOTAL									
Consolidated Mark (100)									
Faculty Signature									

Exp.No: 1	STRING HANDLING
Date:	

Exp. No: 1 a) Displaying Middle value of a String

AIM:

To write a Java program to display middle value of a String.

Input Format:

Get an input from the user.

Output Format:

Return the middle value of the String.

ALGORITHM:

STEP 1: Start.

STEP 2: An object for Scanner class is declared.

STEP 3: The input is got through the Scanner and stored in a variable.

STEP 4: If the length of the string%2 equals 0 then the two letters in the middle of the string is printed with the help of the substring () method of the String class.

STEP 5: Else the letter in the middle of the string is printed with the help of the charAt () method of the String class.

STEP 6: Stop.

PROGRAM:

```
import java.util.Scanner;
class Main {
        public static void main(String [] args) {
        Scanner s = new Scanner(System.in);
        String str = s.nextLine();
        int l = str.length();
        if(l % 2 == 0) {
                System.out.print(str.substring(l/2 , l/2+1));}
        else {
                System.out.print(str.charAt(l/2));}
        }
}
```

OUTPUT:

Input 1:

Carry

Output 1:

r

Input 2:

Placements

Output 2:

em

RESULT:

Thus, java program to display the middle value of the string has been implemented successfully.

Exp. No: 1 b) **Validating Email Address**

AIM:

To write a Java program to validate domain names of the email address. The fair organizers have listed the accepted domains as "com", "in", "net", and "org". Write a program to validate email addresses that have the above listed domain names.

Create a driver class called Main. In the Main method, obtain the inputs from the console and validate the email address.

Input format:

Input consist of the email address

Output format:

Output prints the email address and in next line whether the email address is valid or not. Refer sample input and output for formatting specifications.

ALGORITHM:

STEP 1: Start

STEP 2: An object for Scanner class is declared.

STEP 3: The input is got through the Scanner and stored in a variable.

STEP 4: The string is checked if it has a valid domain using the contains() method in String class.

STEP 5: If it does “Valid email address” is printed.

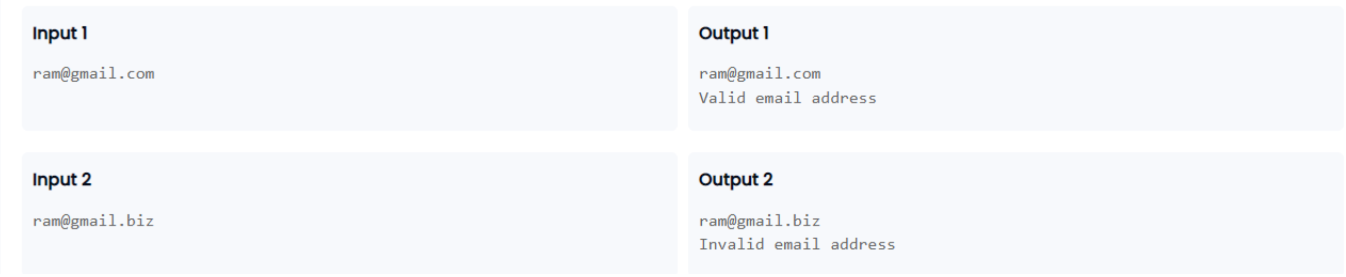
STEP 6: Else “Invalid email address” is printed.

STEP 7: Stop

PROGRAM:

```
import java.util.Scanner;

class main {

        public static void main(String [] args) {

                Scanner sc = new Scanner(System.in);

                String email = sc.nextLine();

                System.out.println(email);

                if(email.contains("com")||email.contains("in")||email.contains("net") ||
                email.contains("org")) {

                        System.out.println("Valid email address"); }

                else {

                        System.out.println("Invalid email address"); }

  } }
```

OUTPUT:



Input 1

```
ram@gmail.com
```

Output 1

```
ram@gmail.com
Valid email address
```

Input 2

```
ram@gmail.biz
```

Output 2

```
ram@gmail.biz
Invalid email address
```

RESULT:

Thus, java program to validate domain names of the email address has been implemented successfully

Exp. No: 1 c) CamelCase notation of a String

AIM:

Camel case (stylized as camelCase or CamelCase) is the practice of writing compound words or phrases such that each word or abbreviation in the middle of the phrase begins with a capital letter, with no intervening spaces or punctuation. Event names should be entered in camel case format. But many users failed to follow this convention. To maintain uniformity, you have to change all the event names into camel case. To Write a Java program to convert event names to camel case format.

Create a driver class called Main. In the Main method, obtain the inputs from the console and print the names of the events in camel case.

Input format

Input consist of the event name.

Output format

Output prints every first letter of the word in uppercase.

ALGORITHM:

```
STEP 1: Start.
STEP 2: An object for Scanner class is declared.
STEP 3: The input is got through the Scanner and stored in a variable.
STEP 4: The string is then converted into a character array.
STEP 5: The first element is converted into upper case using toUpperCase() method.
STEP 6: The array is iterated from the second element.
STEP 7: If whitespace is encountered the character in the next index is converted to upper case.
STEP 8: Then the result array is printed.
STEP 9: Stop
```

PROGRAM:

```
import java.util.*;
class Main {
        public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
```

```
                    char res [] = str.toCharArray();
                    int index = 1,i;
                    res[0] = Character.toUpperCase(res[0]);
                    for( i=1;i<res.length;i++) {
                              if(res[i] == ' ') {
                                        res[i+1] = Character.toUpperCase(res[i+1]);
                                        continue; }
                              else {
                                        res[index++] = res[i]; }
                    }
                    for(i=0;i<index;i++) {
          System.out.print(res[i]); }
   }
}
```

OUTPUT:

Input 1

book sale

Output 1

BookSale

Input 2

camel case

Output 2

CamelCase

RESULT:

Thus, java program to print the names of the events in camel case has been implemented successfully.

Exp.No: 2	FILE HANDLING
Date:	

Exp. No: 2 a) **Minimum element in an Array**

AIM:

To write a Java program to obtain an array and write it into a file. Read the file contents and print the minimum element in the array.

Input Format:

The first line consists of the value of n.

The next input is the array elements.

Output Format:

The output prints the minimum element in the array.

ALGORITHM:

STEP 1: declare an integer variable 'n'.

STEP 2: Read the value of 'n' from the user using a Scanner object.

STEP 3: Create an integer array 'arr' of size 'n'.

STEP 4: Read 'n' integers from the user using a Scanner object, and store them in the 'arr' array.

STEP 5: Create a FileWriter object 'fw' to write the integers to a file named "input.txt".

STEP 6: Iterate through the 'arr' array, and write each integer to the file using the FileWriter object 'fw'.

STEP 7: Close the FileWriter object 'fw'.

STEP 8: Create a FileReader object 'fr' to read the integers from the file "input.txt".

STEP 9: Find the minimum integer value by iterating through the file using a Scanner object and storing the minimum value in a variable 'min'.

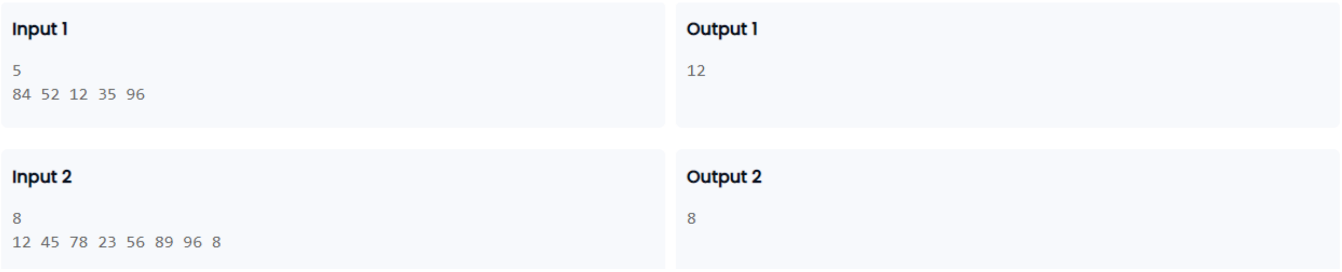
STEP 10: Print the value of 'min' to the console.

PROGRAM:

```
import java.io.*;
import java.util.*;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
 class CreateFile
{
   public static void main(String[] args) throws IOException
```

```
    {
        // Type your code here
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        int []arr=new int[n];
        for(int i=0;i<n;i++){
            arr[i]=s.nextInt();
        }
        FileWriter fw=new FileWriter("input.txt");
        for(int i=0;i<n;i++){
            fw.write(arr[i]+" ");
        }
        fw.close();

        FileReader fr=new FileReader("input.txt");
        Scanner r=new Scanner(fr);
        int min=Integer.MAX_VALUE;
        while(r.hasNextInt()){
            int num=r.nextInt();
            if(num<min){
                min=num;
            }
        }
        r.close();
        System.out.print(min);
    }
}
```

OUTPUT:



Input 1

```
5
84 52 12 35 96
```

Output 1

```
12
```

Input 2

```
8
12 45 78 23 56 89 96 8
```

Output 2

```
8
```

RESULT:

Thus, java program to display the minimum element in an array successfully.

Exp. No: 2 b) Number of digits in a String

AIM:

To write a Java program to obtain a String and write it into a file. Read the file contents and print the number digits present in the file.

Input format:

The input consists of a String.

Output format:

The output print the number of Digits.

ALGORITHM:

```
STEP 1: Create a Scanner object 's' to read input from the user.
STEP 2: Read a string 'str' from the user using the Scanner object 's'.
STEP 3: Count the number of digits in the string 'str' by iterating through its characters using a for loop and checking if each character is a digit using the isDigit() method of the Character class.
STEP 4: Store the count in a variable 'count'.
STEP 5: Write the value of 'count' to a file named "input.txt" using a FileWriter object 'fw'.
STEP 6: Read the contents of the file "input.txt" using a FileReader object 'fr' and a Scanner object 'sc', and store it in a string variable 'str1'.
STEP 7: Print the value of 'str1' to the console.
```

PROGRAM:

```
import java.io.*;
import java.util.*;
class Sample {
  public static void main(String args[]) throws Exception{
      Scanner s=new Scanner(System.in);
      int count=0;
      String str=s.nextLine();
      for(int i=0;i<str.length();i++){
          if(Character.isDigit(str.charAt(i))){
              count++;
```

```
            }
        }
       FileWriter fw=new FileWriter("input.txt");
       fw.write(Integer.toString(count));
       fw.close();
       FileReader fr=new FileReader("input.txt");
       Scanner sc= new Scanner(fr);
       String str1="";
       while(sc.hasNextLine()){
          str1=sc.nextLine();
       }
       System.out.print(str1);
       fr.close();
    }
}
```

OUTPUT:

Input 1:

Phone number : 8976578011

Output 1:

10

Input 2:

Aadhar number: 491861876736

Output 2:

12

RESULT:

Thus, java program to display the number of digits in a string successfully.

Exp. No: 2 c) Number of uppercase and lowercase alphabets in a String

AIM:

To write a Java program to obtain a String and write it into a file. Read the file contents and print the number of uppercase and lowercase letters present in the file.

Input format

The input consists of a String.

Output format

The output print the number of upper case and lower case letters.

ALGORITHM:

STEP 1: Read a string 's' from the user using the Scanner object 'sc'.

STEP 2: Write the string 's' to the file "output.txt" using the FileWriter object 'fw', and then close the FileWriter object.

STEP 3: Create a FileReader object 'fr' to read the contents of the file "output.txt".

STEP 4: Count the number of uppercase and lowercase letters in the file by iterating through its characters using a while loop and checking if each character is an uppercase or lowercase letter using their ASCII values.

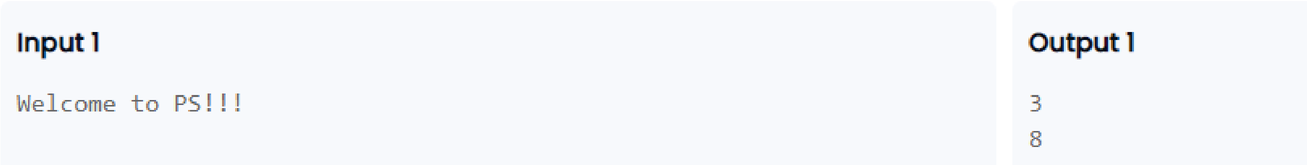
STEP 5: Update the corresponding count variables 'upper' and 'lower' accordingly, and then print their values to the console.

PROGRAM:

```
import java.io.*;
import java.util.*;
class Sample {
  public static void main(String args[]) throws Exception{
      Scanner sc=new Scanner(System.in);
      FileWriter fw=new FileWriter("output.txt");
      String ch;
      int upper=0,lower=0,number=0,special=0;
      String s=sc.nextLine();
      fw.write(s);
      fw.close();
      FileReader fr=null;
      try
      {
            fr = new FileReader("output.txt");
      }
      catch (FileNotFoundException fe)
      {
            System.out.println("File not found");
      }
```

```
             Integer i=fr.read();
             Integer eof=-1;
            ch=i.toString();
            while (ch.charAt(0)!=eof.toString().charAt(0))  {

             if (Integer.parseInt(ch) >= 'A' && Integer.parseInt(ch) <= 'Z')
                upper++;
             else if (Integer.parseInt(ch) >= 'a' && Integer.parseInt(ch) <= 'z')
                lower++;

             i=fr.read();
             ch=i.toString();
             }
        System.out.println(upper);
        System.out.println(lower);}}
```

OUTPUT:



Input 1

```
Welcome to PS!!!
```

Output 1

```
3
8
```

RESULT:

Thus, java program to display the number of upper case and lower case in a string successfully.

Exp.No: 3	SERIALIZATION
Date:	

Exp. No: 3 a)

AIM:

To demonstrate the concepts of Serialization and Deserialization of objects in Java.

Input format:

The first line of user input represents Name of a person, followed by their age.

Output format:

The output prints the received name and age of the person, which has been serialized.

ALGORITHM:

STEP 1: Define a “Person” class that implements Serializable interface.

STEP 2: Set values for the attributes that belong to the “Person” class using Constructor.

STEP 3: In the main method, receive the input using “BufferReader” Class and then create the required object to access the methods and attributes of the “Person” class.

STEP 4: Serialize “Person” object to “person.ser” file using required OutputStreams.

STEP 5: Use exceptions as required to handle errors smoothly.

STEP 6: Deserialize the object using file InputStreams.

PROGRAM:

```
import java.io.*;
class Person implements Serializable {
   private String name;
   private int age;
   public Person(String name, int age) {
      this.name = name;
      this.age = age;
   }
   public String getName() {
      return name;
   }
   public int getAge() {
      return age;
   }
```

```
}

public class SerializationDemo {
    public static void main(String[] args) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String filename = "person.ser";
        System.out.println("Enter person's name:");
        String name = null;
        try {
            name = reader.readLine();
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("Enter person's age:");
        int age = 0;
        try {
            age = Integer.parseInt(reader.readLine());
        } catch (IOException e) {
            e.printStackTrace();
        }
        Person person = new Person(name, age);
        try {
            FileOutputStream file = new FileOutputStream(filename);
            ObjectOutputStream out = new ObjectOutputStream(file);
            out.writeObject(person);
            out.close();
            file.close();
            System.out.println("Object has been serialized");
        }
        catch(IOException ex) {
            System.out.println("IOException is caught");
        }
        try {
            FileInputStream file = new FileInputStream(filename);
            ObjectInputStream in = new ObjectInputStream(file);
            Person newPerson = (Person)in.readObject();
            in.close();
            file.close();
            System.out.println("Object has been deserialized");
            System.out.println("Name: " + newPerson.getName());
            System.out.println("Age: " + newPerson.getAge());
        }
        catch(IOException ex) {
            System.out.println("IOException is caught");
        }
        catch(ClassNotFoundException ex) {
            System.out.println("ClassNotFoundException is caught");
        }
    }
}
```

OUTPUT:

Input:

```
Enter person's name:
John
Enter person's age:
30
```

Output:

```
Object has been serialized
Object has been deserialized
Name: John
Age: 30
```

RESULT:

Thus, the object has been serialized and deserialized successfully.

Exp. No: 3 b)

AIM:

To demonstrate the concepts of Serialization and Deserialization of objects in Java.

ALGORITHM:

STEP 1: A “Dog” class is implemented with a Serializable interface.

STEP 2: The attributes which are marked “transient” and “final” cannot be serialized or deserialized. Hence, other attributes can be done so.

STEP 3: Create an object for “Dog” class in main method.

STEP 4: Serialize the “Dog” object to a file “file.ser” using FileOutputStream and ObjectOutputStream.

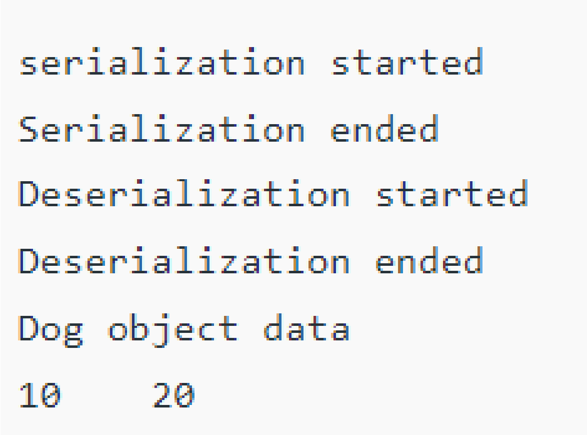
STEP 5: Deserialize the “Dog” object using FileInputStream and ObjectInputStream.

STEP 6: Print the output to the console.

PROGRAM:

```
import java.io.*;
class Dog implements Serializable{
                int i=10;
                transient final int j=20;
}
class Main {
        public static void main (String[] args)throws IOException,ClassNotFoundException
        {
                Dog d1=new Dog();
                System.out.println("serialization started");
                FileOutputStream fos= new FileOutputStream("file.ser");
                ObjectOutputStream oos=new ObjectOutputStream(fos);
                oos.writeObject(d1);
                System.out.println("Serialization ended");
                System.out.println("Deserialization started");
                FileInputStream fis=new FileInputStream("file.ser");
                ObjectInputStream ois=new ObjectInputStream(fis);
                Dog d2=(Dog) ois.readObject();
                System.out.println("Deserialization ended");
                System.out.println("Dog object data");
                System.out.println(d2.i+"\t" +d2.j);
        }
}
```

OUTPUT:



```
serialization started
Serialization ended
Deserialization started
Deserialization ended
Dog object data
10    20
```

RESULT:

Thus, the object has been serialized and deserialized successfully.

Exp.No: 4	JAVA PACKAGES
Date:	

Exp. No: 4 a) Frequency of a string using Java Collections

AIM:

To write a java program that obtains a set of names and a search element and prints its frequency.

Input format

The first line of the input consists of the number of names.
The next input is the user names.
The last input is a user name to be searched.

Output format

The output prints the frequency of the searched element.

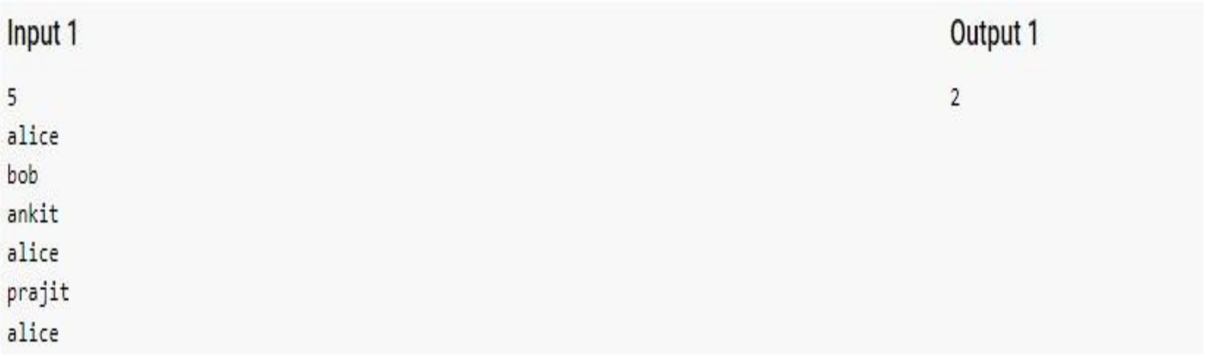
ALGORITHM:

```
STEP 1: Create an empty list.
STEP 2: Ask the user to enter the names and add each name to the list.
STEP 3: Ask the user to enter the search element.
STEP 4: Remove the duplicate elements from the list.
STEP 5: Traverse through the list and count the frequency of the search element.
STEP 6: Print the frequency.
```

PROGRAM:

```
import java.util.*;
class Main{
   public static void main(String args[])
   {
      int num;
      Scanner sc=new Scanner(System.in);
      num=sc.nextInt();
      sc.nextLine();
      ArrayList<String> al=new ArrayList<String>();
      for(int i=0;i<num;i++)
      {
         al.add(sc.nextLine());
      }
      String name=sc.nextLine();
      System.out.println(Collections.frequency(al,name));

      }

}
```

OUTPUT:



```
Input 1                                                Output 1

5                                                      2
alice
bob
ankit
alice
prajit
alice
```

RESULT:

Thus, the program to obtain a set of names and a search element to prints its frequency has been implemented successfully.

Exp. No: 4 b) **Sort strings based on their length using ArrayList and Collections**

AIM:

To input a positive integer N (N > 0), input N strings, and sort the strings in place in the order of increasing length. Print the sorted strings using ArrayList as an implementation of the List interface for storing the individual strings.

Input format

Input number of elements
Input each string on a separate line

Output format

Print the list of strings sorted by their length

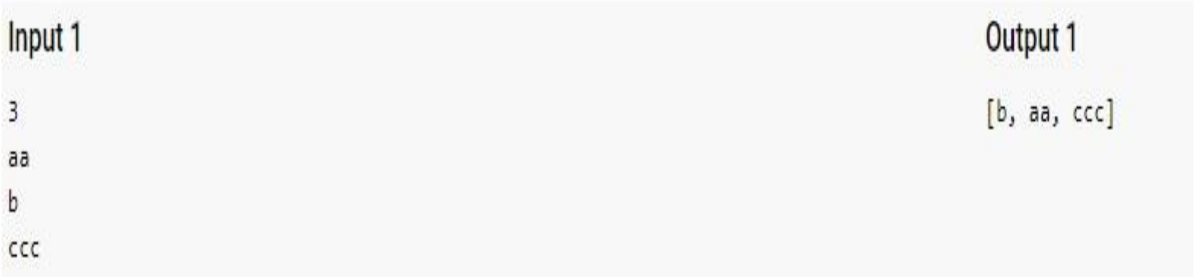
ALGORITHM:

STEP 1: Start the program.
STEP 2: Initialize the string with the input words.
STEP 3: Calculate the string length.
STEP 4: Sort the string array according to the length of words in ascending order with the help of insertion sort.
STEP 5: Print the sorted array.

PROGRAM:

```
import java.util.*;
class Fun
{
        public static void main(String[]args)
        {
                Scanner sc = new Scanner(System.in);
        ArrayList<String> arr = new ArrayList<String>();
                int n = sc.nextInt();
        for (int i=0;i<n;i++)
        {
                        arr.add(i,sc.next());
        }
        Collections.sort(arr,Comparator.comparing(String::length));
        System.out.println(arr);
        }
}
```

OUTPUT:



Input 1

```
3
aa
b
ccc
```

Output 1

```
[b, aa, ccc]
```

RESULT:

Thus, the java program to print the sorted strings using ArrayList has been implemented successfully.

Exp. No: 4 c) **Adding days to a given day**

AIM:

To write a program to find the day and add the day to get exact day of the week.

Input Format:

The input consists of two integers. The first input to get the day. The second input is to get the number of days to be added with the day.

Output Format:

The output prints the day of the first input and then the added value and that day. Refer to the sample input and output for the formatting specifications.\

ALGORITHM:

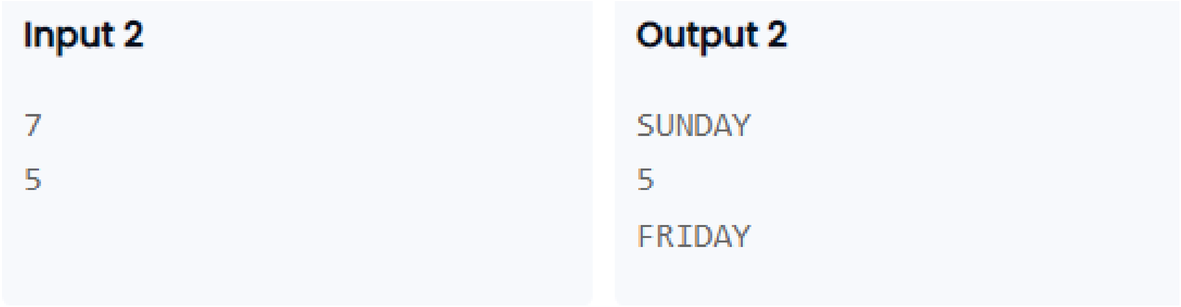
STEP 1: Get the input day and number of days to be added from the user.
STEP 2: Calculate the day of the week using DayOfWeek package imported from java.time package.
STEP 3: Print the Day of the week which was given as input and the added Day of the week to the console.

PROGRAM:

```
import java.util.*;
import java.text.*;
import java.time.DayOfWeek;
class Main{
public static void main(String [] args)
{
   Scanner sc = new Scanner(System.in);
   int x = sc.nextInt();
   int y = sc.nextInt();
   int tot;
   if(x==7||y==7)  {
      tot = (x+y)-7;
      DayOfWeek day1 = DayOfWeek.of(x);
      DayOfWeek day2 = DayOfWeek.of(tot);
      System.out.println(day1);
      System.out.println(tot);
      System.out.println(day2);
   }
      else{
      tot = x+y;
      }
      DayOfWeek day1 = DayOfWeek.of(x);
      DayOfWeek day2 = DayOfWeek.of(tot);
      System.out.println(day1);
```

```
        System.out.println(tot);
        System.out.println(day2);
        }

    }
```

OUTPUT:



Input 2

```
7
5
```

Output 2

```
SUNDAY
5
FRIDAY
```

RESULT:

Thus, the day of the week has been added and printed successfully.

Exp.No: 5	METHODS, CLASSES AND CONSTRUCTORS
Date:	

Exp. No: 5 a) **Counting digits using built-in method**

AIM:

To write a Java program to find the count of all digits of a number using class. In this program, we will read a positive integer number and then calculate the count of all digits using a class.

Input format

The input consists of a number.

Output format

The output prints the count of all digits in the number.

ALGORITHM:

```
STEP 1: Start
STEP 2:An object for Scanner class is declared.
STEP 3: The number input is received by the Scanner and is stored in a String.
STEP 4: The length is calculated by using the length() method for String and it is displayed.
STEP 5: Stop
```

PROGRAM:

```
import java.util.Scanner;
class main {
        public static void main(String ar[]) {
                Scanner s = new Scanner(System.in);
                String number = s.nextLine();
                System.out.print("Count of all digits: " + number.length()); } }
```

OUTPUT:

Input 1:

```
08555
```

Output 1:

```
5
```

Input 2:

```
14
```

Output 2:

```
2
```

RESULT:

Thus, java program to find the count of all digits of a number using class has been implemented successfully.

Exp. No: 5 b) Display Player details using Classes

AIM:

To write a java program to Create a Class named Player with some attributes. Create another Class called Main and write a main method to get the player details in a string separated by a comma. Use String. Split() function to display the details.

Input Format:

Three strings separated by comma.

Output Format:

Displaying the player details separately.

ALGORITHM:

STEP 1: Import the "java.util.Scanner" package to take input from the user.

STEP 2: Create a class "Player" with an instance variable "inp" to store the input provided by the user.

STEP 3: Define a method "display" inside the "Player" class to split the input using "," as a delimiter and print the player's details.

STEP 4: Create a class "Qn3" with the "main" method.

STEP 5: Inside the "main" method, create an object of "Scanner" class to take input from the user.

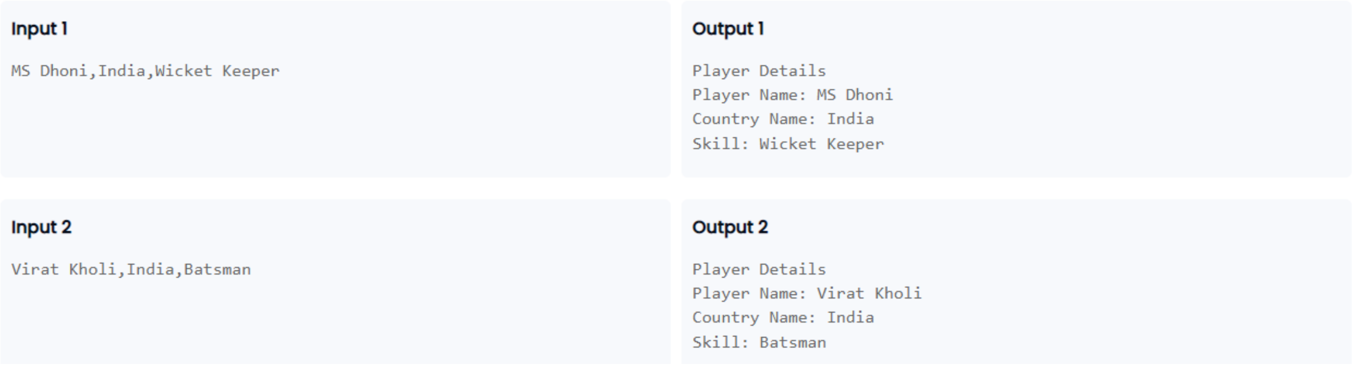
STEP 6: Create an object of the "Player" class, take input from the user, and display the player's details using the "display" method.

PROGRAM:

```
import java.util.Scanner;

class Player{

   String inp;

   public void display()

   {

      String[]el=inp.split(",");

      System.out.println("Player Name: "+el[0]);
```

```
        System.out.println("Country Name: "+el[1]);

        System.out.println("Skill: "+el[2]);

    }

}

class Qn3{

    public static void main(String[] args)

    {

        Scanner s=new Scanner(System.in);

        Player play=new Player();

        play.inp=s.nextLine();

        System.out.println("Player Details");

        play.display();

    }

}
```

OUTPUT:



Input 1

```
MS Dhoni,India,Wicket Keeper
```

Output 1

```
Player Details
Player Name: MS Dhoni
Country Name: India
Skill: Wicket Keeper
```

Input 2

```
Virat Kholi,India,Batsman
```

Output 2

```
Player Details
Player Name: Virat Kholi
Country Name: India
Skill: Batsman
```

RESULT:

The details of the players have been displayed in a structured way, successfully.

Exp. No: 5 c) **Total Money**

AIM:

To write a Java program to Create class money with two attributes:

int rupee int paisa

Include getters, setters, and constructors.

Create the main class and initialize the values for the data members Get two amounts and print their sum.

Input Format:

The input consists of two amounts.
Rupee and Paisa are separated by a space.

Output Format:

The output prints the total sum.

ALGORITHM:

STEP 1: Start

STEP 2: The class Money is declared with two variables Rupee and Paisa.

STEP 3: The setters and getters are declared for that class.

STEP 4: An object for Scanner class is declared.

STEP 5: An array of objects of Money class is declared.

STEP 6: The input is got through the Scanner and stored in the objects using setters.

STEP 7:The values are then got through getters and added.

STEP 8:While the value of paisa is greater than 100 rupee is increased by 1 and 100 is deducted from paisa.

STEP 9:The result value is printed.

STEP 10: Stop

PROGRAM:

```
import java.util.Scanner;
class money {
        int rupee;
        int paisa;
        public void setRupee(int r) {
        this.rupee = r; }
        public void setPaisa(int p) {
     this.paisa = p; }
        public int getRupee() {
```

```
        return rupee; }
        public int getPaisa() {
        return paisa; }
}
class Main {
        public static void main(String [] args) {
                Scanner sc = new Scanner(System.in);
                money [] m = new money[2];
                int i;
        for(i=0;i<2;i++) {
                m[i] = new money();
                m[i].setRupee(sc.nextInt());
                m[i].setPaisa(sc.nextInt()); }
                int r,p;
        r = m[0].getRupee()+m[1].getRupee();
                p = m[0].getPaisa()+m[1].getPaisa();
                if(p>99) {
                r +=1;
                p = p-100; }
        System.out.println(r+"."+p);
  }
}
```

OUTPUT:

Input 1:

```
50 85
42 65
```

Output 1:

```
93.50
```

Input 2:

```
254 45
845 20
```

Output 2:

```
1099.65
```

RESULT:

Thus, java program to get two amounts and print their sum has been implemented successfully.

Exp.No: 6	ABSTRACT CLASS AND INTERFACE
Date:	

Exp.No. 6 a) Write a **Java program to demonstrate Abstract class and interface concepts.**

AIM:

To write a java program to implement an abstract class **Marks** with a method **getPercentage().** Then, create a class called **A** which extends **Marks** and has three attributes named

- **Marks1**
- **Marks2**
- **Marks3**

This class should also have a method named **getPercentage()** that calculates and prints the percentage of the student. Similarly, create another class called **B** that extends **Marks** and has four attributes named

- **Marks1**
- **Marks2**
- **Marks3**
- **Marks4**

This class should also have a method named **getPercentage()** that calculates and prints the percentage of the student. Please note that each mark is out of 100, and you should find the percentage for two students (Student A and Student B) and round off the output to two decimal places.

Input format:

The first line of the input consists of three integers, i.e., the marks scored by student A.
The second line of the input consists of four integers, i.e., the marks scored by student B..

Output format:

The first line prints the percentage of A.
The second line prints the percentage of B.

ALGORITHM:

STEP 1: Start
STEP 2: Define an abstract class "Marks" with an abstract method "getPercentage()".
STEP 3: Define a class "A" that extends "Marks", with three attributes named "Marks1", "Marks2", and "Marks3".
STEP 4 : Define a constructor for class "A" that takes three parameters and initializes the "Marks1", "Marks2", and "Marks3" attributes.
STEP 5: Implement the "getPercentage()" method in class "A" to calculate the average of the three marks and return the percentage.
STEP 6: Define a class "B" that extends "Marks", with four attributes named "Marks1", "Marks2", "Marks3", and "Marks4".
STEP 7: Define a constructor for class "B" that takes four parameters and initializes the "Marks1", "Marks2", "Marks3", and "Marks4" attributes.
STEP 8: Implement the "getPercentage()" method in class "B" to calculate the average of the four marks and return the percentage.
STEP 9: In the main method, create an instance of class "A" and call its "getPercentage()" method to calculate the percentage for Student A.

STEP 10: Create an instance of class "B" and call its "getPercentage()" method to calculate the percentage for Student B.
STEP 11: Round off the output to two decimal places.
STEP 12: Print the percentages for Student A and Student B.
STEP 13: Stop.

PROGRAM:

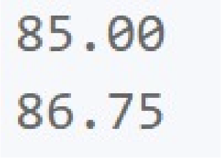
```
import java.util.*;
import java.text.DecimalFormat;

abstract class Marks{
    abstract void s1(double d1, double d2, double d3);
    abstract void s2(double d1, double d2, double d3, double d4);
}

class A extends Marks{
    DecimalFormat d=new DecimalFormat("0.00");
    int m1, m2, m3;

    A(int ma1, int ma2, int ma3){
        m1 = ma1;
        m2 = ma2;
        m3 = ma3;
    }

    public void s1(double d1, double d2, double d3){
        double total = (m1 + m2 + m3) / 3.0;
        System.out.println(d.format(total));
    }

    public void s2(double d1, double d2, double d3, double d4){

    }
}

class B extends Marks{
    DecimalFormat d=new DecimalFormat("0.00");
    int m1, m2, m3, m4;

    B(int ma1, int ma2, int ma3, int ma4){
        m1 = ma1;
        m2 = ma2;
        m3 = ma3;
        m4 = ma4;
    }

    public void s1(double d1, double d2, double d3){
    }

    public void s2(double d1, double d2, double d3, double d4){
        double total = (m1 + m2 + m3 + m4) / 4.0;
        System.out.println(d.format(total));
    }
}

class Main{
    public static void main(String args[]){
        Scanner s = new Scanner(System.in);
        double d1, d2, d3, d4, d5, d6, d7;

        d1 = s.nextDouble();
        d2 = s.nextDouble();
        d3 = s.nextDouble();
        A a = new A((int) d1, (int) d2, (int) d3);
        a.s1(d1, d2, d3);

        d4 = s.nextDouble();
        d5 = s.nextDouble();
        d6 = s.nextDouble();
        d7 = s.nextDouble();
        B b = new B((int) d4, (int) d5, (int) d6, (int) d7);
        b.s2(d4, d5, d6, d7);
    }
}
```

OUTPUT:

Input 1:

```
95 85 75
85 77 92 93
```

Output 1:



```
85.00
86.75
```

RESULT:

Thus, java program to calculate and print the percentage of two students using abstract class and interface has been implemented successfully.

Exp.No. 6 b) Write a Java program to demonstrate Abstract class and interface concepts.

AIM:

Write a java program to create an interface called ShapeCalculator that has a method called calc(int n).

Then, create two classes called

- Square
- Circle that implement the ShapeCalculator interface and implement the calc(int n) method.

Your program should calculate the area and perimeter of both squares and circles.

Input format:

The input to your program will be a single integer that represents the side of the square and the radius of the circle.

Output format:

The output of your program should be the area and perimeter of each shape. The output should be displayed in two lines. The first line should contain the area and perimeter of the square separated by a space, and the second line should contain the details of the circle in a similar format.

Please note that the details of the square should be calculated using integer type while the details of the circle should be calculated using double type.

ALGORITHM:

```
STEP 1: Start
STEP 2: Create an interface called ShapeCalculator with a method called calc(int n).
STEP 3: Create a class called Square that implements the ShapeCalculator interface.
STEP 4 : Implement the calc(int n) method in the Square class to calculate the area and perimeter of the square using the formulae:
area = n * n, perimeter = 4 * n.
STEP 5: Create a class called Circle that implements the ShapeCalculator interface.
STEP 6: Implement the calc(int n) method in the Circle class to calculate the area and perimeter of the circle using the formulae:
area = pi * n * n, perimeter = 2 * pi * n, where pi = 3.14.
STEP 7: Read an integer input from the user representing the side of the square and the radius of the circle.
STEP 8: Create an object of the Square class and call the calc(int n) method passing the integer input as a parameter to calculate the area and perimeter of the square.
STEP 9: Create an object of the Circle class and call the calc(int n) method passing the integer input as a parameter to calculate the area and perimeter of the circle.
STEP 10: Display the calculated area and perimeter of the square and circle in two separate lines, each containing two space-separated values.
STEP 11: Stop.
```

PROGRAM:

```
import java.util.*;
interface ShapeCalculator {
  void calc(int n);
}

class Square implements ShapeCalculator {
   public void calc(int n) {
      int area = n * n;
      int perimeter = 4 * n;
      System.out.println(area + " " + perimeter);
   }
}

class Circle implements ShapeCalculator {
   public void calc(int n) {
      double pi = 3.14;
      double area = pi * n * n;
      double perimeter = 2 * pi * n;
      System.out.printf("%.2f %.2f\n", area, perimeter);
   }
}

class Whitelist implements ShapeCalculator {
   public void calc(int n) { // This class does not perform any calculations
   }
}

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      sc.close();

      Square s = new Square();
      Circle c = new Circle();
      Whitelist w = new Whitelist();

      s.calc(n);
      c.calc(n);
      w.calc(n);
   }
}
```

OUTPUT:

Input 1:	**Output 1:**
8	64 32 200.36 50.24

RESULT:

Thus, java program to calculate and print the area and perimeter of both squares and circles. using abstract class and interface has been implemented successfully.

Exp.No. 6 c) Write a **Java program to demonstrate Abstract class and interface concepts.**

AIM:

Write a java program to count the minimum number of front moves required to sort an array. Your program should create an interface that declares a method. The class should implement this interface.

Input format:

The first line of the input consists of an integer.

The second line of the input consists of an integer.

Output format:

The output of your program should be an integer representing the minimum number of front moves required to sort the array.

ALGORITHM:

```
STEP 1: Start
STEP 2: Define an interface with a method ‘minFrontMoves’ that takes an integer array and its size as input and returns an integer.
STEP 3: Create a class that implements the interface and provides an implementation for the ‘minFrontMoves’ method.
STEP 4 : In the minFrontMoves method, initialize a variable minMoves to 0.
STEP 5: Traverse the array from left to right.
STEP 6: For each element, compare it with the current minimum element found so far.
STEP 7: If the current element is smaller than the minimum element, increment minMoves.
STEP 8: Return minMoves.
STEP 9: Stop.
```

PROGRAM:

```
import java.io.*;
import java.util.Scanner;

interface Move{
   int minmoves(int arr[], int n);
}

class Main implements Move {
   public int minmoves(int arr[], int n) {
      int expectedItem = n;
      for (int i = n - 1; i >= 0; i--) {
         if (arr[i] == expectedItem) {
            expectedItem--;
         }
      }
      System.out.print(expectedItem);
      return expectedItem;
   }

   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int n;
      n = in.nextInt();
      int arr[] = new int[n];
      for (int i = 0; i < n; i++) {
         arr[i] = in.nextInt();
      }
      Main obj = new Main();
      obj.minmoves(arr, n);
   }
}
```

OUTPUT:

Input1:

```
 5
 2 3 1 4 5
```

Output 1:

```
1
```

Input2:

```
7
5 6 3 2 4 1 7
```

Output2:

```
4
```

RESULT:

Thus, java program to count the minimum number of front moves required to sort an array. using abstract class and interface has been implemented successfully.

Exp.No: 7	POLYMORPHISM
Date:	

Exp.No. 7 a) **Write** a **Java program to demonstrate the concept of polymorphism.**

AIM:

Ram is given two or three input as an integer, if he has two integers then add the two numbers. If he has three input, then multiply the three numbers.

Function Header:

public void fun1(int a,int b,int c)
public void fun1(int a,int b)

Input format:

First line represents the number of elements(N) followed by the elements separated by the single space. If the number of the elements exceed 2 or 3, then display message as WRONG INPUT.

Output format:

Display the sum,if there are two integers or Displays product,if there are three integers.

Code constraints

N > 0 and N < 4

ALGORITHM:

STEP 1: Start
STEP 2: Define a Main class .
STEP 3: Define two overloading functions fun1() . public void fun1(int a,int b,int c) – prints the product of integers a , b and c public void fun1(int a,int b) – prints the sum of integers a and b
STEP 4: Create a main function to input numbers and perform the function.
STEP 5: End.

PROGRAM:

```
import java.util.Scanner; class Main
{
   public void fun1(int a,int b,int c)
   {
      System.out.println(a*b*c);
   }

   public void fun1(int a,int b)
   {
      System.out.println(a+b);
```

```
    }

    public static void main(String args[])
    {
      Scanner sc= new Scanner(System.in);
      Main m = new Main();

      int a , b , c ,n;

      n=sc.nextInt();

      switch(n)
      {
          case 2: {
              a=sc.nextInt();
              b=sc.nextInt();
             m.fun1(a,b);
             break;
          }
          case 3:{
              a=sc.nextInt();
              b=sc.nextInt();
              c=sc.nextInt();
              m.fun1(a,b,c);
              break;
         }
         default:{  System.out.println("WRONG INPUT");
        }
      }
    }
}
```

OUTPUT:

Input 1:	**Output 1:**
3 1 2 3	6
Input 2:	**Output 2:**
2 14 56	70

RESULT:

Thus, java program to add three numbers using function overloading has been complied and executed successfully.

Exp.No. 7 b) Write a Java program to demonstrate the concept of polymorphism.

AIM:

To create a class named 'Hello'. Define a method 'sayHello'

1. Create an object obj.
2. Call method 'sayHello' without argument, Output should display 'Hello'.
3. Call method 'sayHello' with one argument, Output should display 'Hello 'argument value''

(Ex: If the argument passed is 'John' Output should display 'Hello John')

Input format:

The input contains a string.

Output format:

The first line of the output should display 'Hello'

The second line of the output should display 'Hello <input>'

ALGORITHM:

```
STEP 1:   Start
STEP 2:   Define a Main class .
STEP 3:   Define two overloading functions sayHello
              public String sayHello() – returns "Hello"
              public String sayHello (String str1)   – returns "Hello" with str1
STEP 4:  Create a main function to input string  .
STEP 5:  Create an object for Main class and call the functions.
STEP 6:  End,
```

PROGRAM:

```
import java.util.Scanner; class Hello{
   public String sayHello()
   {
      return "Hello";
   }

   public String  sayHello(String str1)
   {
      String a=("Hello "+str1);
      return a;
   }
   public static void main(String args[])
   {
      Scanner in=new Scanner(System.in);
```

```
        String str=in.nextLine();
        Hello s = new Hello ();
        System.out.println(s.sayHello());
        System.out.println(s.sayHello(str));
    }
}
```

OUTPUT:

Input 1:

```
John
```

Output 1:

```
Hello
Hello John
```

RESULT:

Thus, java program to print “Hello” using function overloading has been complied and executed successfully.

Exp.No. 7 c) Write a Java program to demonstrate the concept of polymorphism.

AIM:

To write a program by creating a class Bicycle as a base class with the number of gears and speed of the bicycle as integer attributes and create a class called MountainBike, a derived class that extends the Bicycle class with an attribute seat height as an integer. Create a Test class to run the program and obtain the output in the console.

Note: Override toString() method to display the details of the bicycle.

Input format:

To get 3 integers from the user (Number of gears, Speed of bicycle, and Seat height).

Output format:

To display the desired output from the test class.
Code constraints integers only.

ALGORITHM:

```
STEP 1:   Start
STEP 2:   Define a parent class Bicycle with global variables gear and speed. Define a method
toString to print the  values.
STEP 3:    Define child class MountainBike with a class variable height. Override the toString
method in Bicycle and print the gear , speed and height;
STEP 4:   Create a Main class to input the values .
STEP 5:   Create an object for MountainBike class and call the toString method.
STEP 6:    End.
```

PROGRAM:

```
import java.util.Scanner; class Bicycle
{    public int gear;
     public int speed;

   public Bicycle(int gear, int speed)
   {
      this.gear = gear;
      this.speed = speed;
   }
   // toString() method to print info of Bicycle     public String toString()
   {
      return("No of gears are "+gear
             +"\n"
```

```
            + "speed of bicycle is "+speed);
    }
}

class MountainBike extends Bicycle
{
    public int seatHeight;

    public MountainBike(int gear,int speed, int startHeight)
    {
        super(gear, speed);        seatHeight = startHeight;
    }

    public void setHeight(int newValue)
    {
        seatHeight = newValue;
    }

    public String toString()
    {
        return (super.toString()+  "\nseat height is "+seatHeight);
    }
}

class Test
{public static void main(String args[])
    {
         int gear,speed,startHeight;
         Scanner sc=new Scanner(System.in);
         gear=sc.nextInt();         speed=sc.nextInt();         startHeight=sc.nextInt();
        MountainBike mb = new MountainBike(gear,speed,startHeight);
System.out.println(mb.toString());  }}
```

OUTPUT:

Input 1:

```
2 90 40
```

Output 1:

```
No of gears are 2
Speed of bicycle is 90
Seat height is 40
```

RESULT:

Thus, java program to create classes ‘Bike’ and ‘MountainBike’ using function overriding has been complied and executed successfully.

Exp.No: 8	INHERITANCE
Date:	

Exp.No. 8 a) Write a **Java program to concept of Inheritance.**

AIM:

To write a simple program to demonstrate the interface.

Interface details:

Set the value of integer as 10 and then call the display function. In display method, obtain the value of the string and display them. In main method, display the integer value.

Input format:

The input consists of the string.

Output format:

The first line of the output prints the string.
The second line prints the integer value which is set as 10.
Code constraints
Integers and strings only.

ALGORITHM:

STEP 1: Start
STEP 2: An interface in1 is declared with a variable a=10 and display method is declared.
STEP 3: The class testClass is that implements the in1 interface.
STEP 4: Inside the constructor, an integer count is initialized to 0.
STEP 5: The definition of the display method is given to print the details required.
STEP 6: An object for Scanner class is declared.
STEP 7: The values are got through the scanner and is passed to the parametrized constructor of the testClass to create an object.
STEP 8: The display method is called to print the required values.
STEP 9: Stop

PROGRAM:

```
import java.util.Scanner;
interface in1
{
        final int a=10;    void display();
}
class testClass implements in1
{
   public void display()
```

```
    {
        String sr;
        Scanner sc=new Scanner(System.in);
        sr=sc.nextLine();
        System.out.println(sr);
    }

    public static void main (String[] args)
    {
        testClass t = new testClass();
        t.display();
        System.out.println(a);
    }
}
```

OUTPUT:

Input1:

```
spring
```

Output1:

```
spring
10
```

Input2:

```
neet
```

Output2:

```
neet
10
```

RESULT:

Thus java program to demonstrate the interface has been implemented successfully.

Exp.No. 8 b) Write a Java program to concept of inheritance.

AIM:

To write a program, such that you should get N number of elements from the user and compute sum of elements in the odd and even position. Print the elements which has the highest sum.

Note: use interface and inheritance.

Input format:

Input to get the number of values N in first line,next line to get N number of elements followed by single space.

Output format:

Display the output as shown in the sample output.

Code constraints

N (size should be even)

ALGORITHM:

STEP 1: Start
STEP 2: Define an interface Total with a function void printCoins(int ar[],int n)
STEP 3: Define a class Coin that implements the interface.The class implements the printCoin method to find the sum of numbers in odd and even position.Then prints the elements with highest sum.
STEP 4: Create a Main class to input numbers and perform the function.
STEP 5: End.

PROGRAM:

```
import java.util.Scanner;
interface Total{
 void printCoins(int arr[], int n) ;
}
class Coin implements Total
{
          public void printCoins(int arr[], int n) {
                  int oddSum = 0;
                  for (int i = 0; i < n; i += 2)
                           oddSum += arr[i];
                          int evenSum = 0;
                  for (int i = 0; i < n; i += 2)
                          evenSum += arr[i];

int start = ((oddSum > evenSum) ? 0 : 1);
for (int i = start; i < n; i += 2)      System.out.print(arr[i]+" ");
 }}

class Main extends Coin implements Total{
          public static void main(String[] args) {
                     int n1,i;
                     Scanner in=new Scanner(System.in);
                     n1=in.nextInt();
                    int arr1[]=new int[n1];
                     for(i=0;i<n1;i++){
                               arr1[i]=in.nextInt();
          }
           Main obj=new Main();
           if(n1%2==0){     obj.printCoins(arr1,n1);
          }     else{
                              System.out.print("Enter valid number");
    }
}
}
```

OUTPUT:

Input1:

```
4
2 3 1 4
```

Output1:

```
3 4
```

Input2:

```
7
8 9 0 54 3 2
```

Output2:

```
Enter valid number
```

RESULT:

Thus, java program compute sum of elements in the odd and even position and print the elements which has the highest sum has compiled and successfully executed.

Exp.No: 9	EXCEPTION HANDLING
Date:	

Exp.No. 9 a) **Write a Java program to concept of exception handling.**

AIM:

To write a program to obtain two numbers and print their quotient. In case of exception print the same.

Input format:

Given a single line input separated by space.get the Integer N1 and N2

Output format:

Display the quotient if there is no Exception.else print the Exception.

Code constraints Integers only.

ALGORITHM:

STEP 1: Start
STEP 2: Define a class named Main
STEP 3: Input 2 numbers a and b using nextIn. Use a try block to print a/b
STEP 4: Declare a catch block which prints java.lang.ArithmeticException: / by zero in case an exception occurs
STEP 5: End.

PROGRAM:

```
import java.util.*;
class Main{
        public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        try{ System.out.print(a/b); }
        catch(ArithmeticException e)
        {System.out.print("java.lang.ArithmeticException: / by zero"); }
    }
}
```

OUTPUT:

Input1:	**Output1:**
44 2	22

RESULT:

Thus, a java program is created to perform division of two numbers and handle zero division by zero error.

Exp.No. 9 b) Write a Java program to concept of exception handling.

AIM:

To create a class Bank with the following private attributes and create class BankBO with the following method.In this program we will include appropriate getters/setters and add constructors. We will create a driver class called Main. In the Main method, we have to obtain inputs from the user and validate the balance and if there is an exception, handle the exception and prompt the user(Refer I/O) Pass the exception message as "Balance is less than 1000".

Input format:

First line of the input consists of account number
Second line of the input consist of name of the account holder
Third line of the input consists of the account balance

Output format:

Output prints the account details if the balance is greater than 1000 otherwise throws an invalid balance exception.

ALGORITHM:

```
STEP 1: Start
STEP 2: Define a class named Bank which has the values of accno, name and bal
STEP 3: Define a constructor for Bank which initialises the values of accno, name and bal
STEP 4: Define getters and setters for accno, name and bal
STEP 5: Define a function toString which returns the account details as a string
STEP 6: Define a class BankBO
STEP 7: Define a function validate which throws InvalidBalanceException if the balance is less than 1000
STEP 8: Define a class InvalidBalanceException which extends from Exception class and prints "Balance is less than 1000" as a string
STEP 9: Define the class Main and get the account details from the user
STEP 10: Define a try block which calls validate function to check the bank balance
STEP 11: Define a catch block which prints out the error message incase exception occurs
STEP 12: Stop.
```

PROGRAM:

```
import java.io.*;
import java.util.*; class Bank {     private int accno;
        private String name;
        private double bal;
        public Bank() {
                this.accno = 0;
```

```
                 this.name = null;
                 this.bal = (double)0;
    }

 public Bank(int accno, String name,double bal) { this.accno = accno;         this.name =
name;         this.bal = bal;
}

public int getAccno() { return accno;
}

public void setAccno(int accno) {
this.accno = accno;
}

public String getName() { return name;
}

public void setName(String name) { this.name = name;
}

public double getBal() {
return bal;
}

public void setBal(double bal) {
this.bal = bal;
} public String toString() { return accno+" "+name+" "+bal;
}
} class BankBO {
 public void validate(Bank b) throws InvalidBalanceException {  if(b.getBal() < 1000) {
           throw new InvalidBalanceException("Balance is less than 1000");
 }
 } }
class InvalidBalanceException extends Exception { public
InvalidBalanceException(String s) {
System.out.println(s);
} } class Main {
public static void main(String [] args) {
Scanner sc = new Scanner(System.in);
Bank b = new Bank();
b.setAccno(Integer.parseInt(sc.nextLine()));
b.setName(sc.nextLine());
b.setBal(Double.parseDouble(sc.nextLine())); BankBO bbo = new BankBO();
```

```
try {
        bbo.validate(b);      System.out.println(b);
}
catch(Exception e) {
        System.out.println(e);
}
}
}
```

OUTPUT:

```
Input1:                                        Output1:
10001                                          10001 Ankit 5000.0
Ankit
5000

Input2:                                        Output2:
10001                                          Balance is less than 1000
Ankit                                          InvalidBalanceException
500
```

RESULT:

Thus, a java program is created to validate the user’s bank balance using the concept of custom user exceptions.

Exp.No. 9 c) **Write a Java program to concept of exception handling.**

AIM:

In this program we will test out Null pointer exception where we will assign null value to a string and obtain an index position and try to access it and thereby print the exception.

Input format

Input consists of an integer.

Output format

Output prints the null pointer exception.

ALGORITHM:

```
STEP 1: Start
STEP 2: Define a class named Main
STEP 3: Define a try black which a string str assigned to null
STEP 4: Get the value of index from the user
STEP 5: Print the string and the character at that particular index
STEP 6: Define a catch block which prints Null Pointer Exception incase exception arises
STEP 7: Stop.
```

PROGRAM:

```
import java.io.*; import java.util.*; class Main { public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        try { String str = null; int index = Integer.parseInt(sc.nextLine());
        System.out.println(str);System.out.println(str.charAt(index));
        } catch(NullPointerException n) {
                System.out.println(n);
        }
} }
```

OUTPUT:

Input1:	**Output1:**
9	null Java.lang.NullPointerException.

RESULT:

Thus, a java program is created to test out Null Pointer Exception and executed successfully.

Exp.No: 10	LAMBDA EXPRESSIONS
Date:	

Exp. No: 10 a)

AIM:

To write a Java program to print a sentence by lambda expressions without parameters.

ALGORITHM:

```
STEP 1: Define the "Sayable" interface with "say" method having String return type.
STEP 2: Create a public class named "LambdaExpression”.
STEP 3: Within main method,
   Initialize variable "s" of type "Sayable" with Lambda expression.
   The lambda expression returns the String "I have nothing to say."
   Print the result of "say" method using "System.out.println".
STEP 4: End of program
```

PROGRAM:

```
interface Sayable{
        public String say();}
public class LambdaExpression{
        public static void main(String[] args) {
        Sayable s=()->{
                return "I have nothing to say.";  };
        System.out.println(s.say());
 }}
```

OUTPUT :

```
I have nothing to say.
```

RESULT :

Thus, java program to display the sentence by lambda expressions has been implemented successfully.

Exp.No: 10 b)

AIM:
To write a Java program to print a sentence by lambda expressions with one parameter.

Input format :
The first line of the input consists of the name.

Output format :
Display the name with the greeting as per the sample output.

ALGORITHM:

STEP 1 : Import the "java.util" package.

STEP 2 : Define the "Sayable" interface with a method "say" that takes a String parameter and returns a String.

STEP 3 : Create a public class named "LambdaExpression".

STEP 4 : Within main method,

Initialize variable "name" with input taken from the user using Scanner class.

Initialize variable "s1" of type "Sayable" with lambda expression having one parameter.

The lambda expression returns a string that concatenates "Hello, " with the parameter string.

Print the result of "say" method using "System.out.println".

Initialize variable "s2" of type "Sayable" with another lambda expression having one parameter with omitted parentheses.

The lambda expression returns a string that concatenates "Hello, " with the parameter string.

Print the result of "say" method using "System.out.println".

STEP 5 : End of program

PROGRAM :

```
import java.util.*;
interface Sayable{
   public String say(String name);
}
public class LambdaExpression{
   public static void main(String[] args) {
      Scanner sc  = new Scanner(System.in);
```

```
        String name = sc.nextLine();
        // Lambda expression with single parameter.
        Sayable s1=(name)->{
            return "Hello, "+name;  };
        System.out.println(s1.say(name));
        // You can omit function parentheses
        Sayable s2= name ->{
            return "Hello, "+name;  };
        System.out.println(s2.say(name"));
 } }
```

OUTPUT :

```
Input : Nandhini
Output :    Hello, Nandhini
            Hello, Nandhini
```

RESULT :

Thus, java program to display the sentence by lambda expressions has been implemented successfully.

Exp. No: 10 c)

AIM:

To write a Java program to print a sentence by lambda expressions with more than one parameter.

Input format :

The first line of the input consists of two integers separated by a space.

Output format :

Display the sum of the integers as per the sample output.

ALGORITHM:

```
STEP 1 : Import the "java.util" package.
STEP 2 : Define an interface named Addable with a single method named add that takes two int parameters and returns an int.
STEP 3 : Create a public class named LambdaExpression.
STEP 4 : Within the main method, create a new instance of the Scanner class to take user input.
STEP 5 : Read in two integers from the user using sc.nextInt(), and store them in variables a & b.
STEP 6 : Create a variable ad1 of type Addable and set it to a lambda expression with two parameters that adds them together and returns the result.
STEP 7 : Print the result of calling ad1.add(a,b).
STEP 8 : Create a variable ad2 of type Addable and set it to a lambda expression with two int parameters that adds them together and returns the result.
STEP 9 : Print the result of calling ad2.add(a,b).
STEP 10 : End of program
```

PROGRAM :

```java
import java.util.*;
interface Addable{
   int add(int a,int b);  }
public class LambdaExpression{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();

      // Multiple parameters in lambda expression
      Addable ad1=(a,b)->(a+b);
      System.out.println(ad1.add(a,b));
      // Multiple parameters with data type in lambda expression
      Addable ad2=(int a,int b)->(a+b);
```

```
        System.out.println(ad2.add(a,b));
    }
}
```

OUTPUT :

```
            Input : 10 20
            Output :   30
                       30
```

RESULT :

Thus, java program to display the sum of the numbers by lambda expressions has been implemented successfully.

Exp.No: 11	JAVA DATABASE CONNECTIVITY
Date:	

Exp. No: 11 a)

AIM:

To write a java program to connect with MySQL database and check whether the given user name and password are valid. Display Login Successful or Invalid username/password based on the user input.

Use below Database Credentials

Driver Class : com.mysql.jdbc.Driver
Default DB URL : jdbc:mysql://localhost/ri_db
Username : test
Password : test123

Note : Table is already created and set of user name and passwords are inserted.
Table Name : userdetails
Sample User Name and Password

Input format :

User name in first line as string
Password in second line as string

Output format :

Login Successful or Invalid username/password

ALGORITHM:

STEP 1: Start the program.
STEP 2: Load the Driver class STEP 3: Create a connection object.
STEP 4: Create a statement object.
STEP 5: Get the connection, execute the select query as an SQL command and retrieve the result as a set.
STEP 6: Check for username and password in the result set.
STEP 7: Display the result based on the SQL query output.
STEP 8: Close the connection object and the statement object.
STEP 9: End the program.

PROGRAM:

```
import java.util.*;
 import java.sql.*;
class Main{
   public static void main(String args[]) throws ClassNotFoundException,SQLException{
      Scanner sc=new Scanner(System.in);
      String userin=sc.next();
      String passin=sc.next();
```

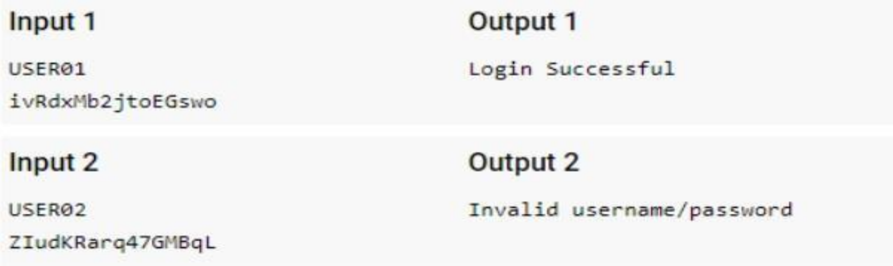
```
        Class.forName("com.mysql.jdbc.Driver");
        Connection x = DriverManager.getConnection("jdbc:mysql://localhost/ri_db","test","test123");
        String q="Select * from userdetails";
        PreparedStatement st=x.prepareStatement(q);
        ResultSet res=st.executeQuery();
        int c=0;
        while(res.next()){
           if(res.getString(1).equals(userin)){
              if(res.getString(2).equals(passin)){
                     c++;
                     break;
              } }
        }
        if(c==1){
           System.out.println("Login Successful"); }
        else{
           System.out.println("Invalid username/password"); }
    }
}
```

OUTPUT:

The first line of the input is the username as string

The second line of the input is the password as string

Output consists of either “Login Successful” or “Invalid username/password”



Input 1

```
USER01
ivRdxMb2jtoEGswo
```

Output 1

```
Login Successful
```

Input 2

```
USER02
ZIudKRarq47GMBqL
```

Output 2

```
Invalid username/password
```

RESULT:

Thus, Java Program to connect with MySQL database and check whether the given username and password are valid has been implemented successfully.

Exp. No: 11 b)

AIM:

To Create a class Stall with the following private attributes using Java DataBase Connection program.

Attributes: id as int name as String and
deposit as Double

Generate appropriate constructors, getters and setters.
Create a class StallBO with the following methods public Hall getStall(int id) - Finds all details of Stall based on stall id and return them.

ALGORITHM:

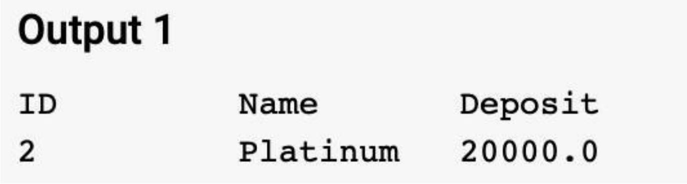
STEP1: Start the program
STEP2: Create a class with the name “STALL”.
STEP3: Load the Driver class
STEP4: Create a Connection object
STEP5: Create a Statement object
STEP6: Get the Connection and execute the create, retrieve, update and delete operations on an SQL Command
STEP7: Close the Connection object, Statement object. st.close(); cn.close();

PROGRAM:

```
import java.sql.*; import java.util.Scanner;
class Stall{
        int id;
        String name;
        Double deposit;
        public Stall() {
                super(); }
        public Stall(int id, String name,Double deposit) {
                super();
                this.id = id;   this.name = name;
                this.deposit = deposit;
} }
class StallBO {
        public Stall getStall(int id) throws Exception {
                String url="jdbc:mysql://localhost/ri_db";
                String username ="test";
                String password ="test123";
                Class.forName("com.mysql.jdbc.Driver");
                String query = "select *from Stall where id="+id;
                Connection con = DriverManager.getConnection(url,username,password);
                Statement st = con.createStatement();
                ResultSet rs =st.executeQuery(query);
                rs.next();
                String name = rs.getString(2);
                Double deposit =rs.getDouble(3);
                Stall s = new Stall();
                s.id=id;
                s.name=name;
                s.deposit=deposit;
                return s;
 } }
class Main {
        public static void main(String args[]) throws Exception {
                StallBO sbo = new StallBO();
                Scanner sc = new Scanner(System.in);
                int id = Integer.parseInt(sc.nextLine());
                Stall S1 = sbo.getStall(id);
                System.out.format("%-10s %-10s %-10s\n","ID","Name","Deposit");
                System.out.format("%-10s %-10s %-10s\n",S1.id,S1.name,S1.deposit);
 } }
```

OUTPUT:

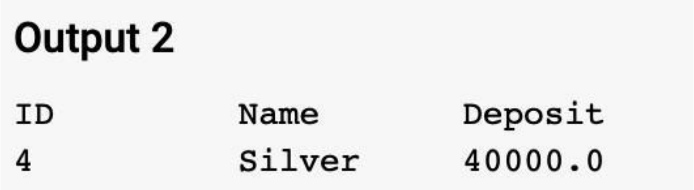
Input 1:



Output 1

```
ID         Name       Deposit
2          Platinum   20000.0
```

Input 2:



Output 2

```
ID         Name       Deposit
4          Silver     40000.0
```

RESULT:

Thus, java program to create a class Stall with the private attributes has been implemented successfully.

Exp. No: 11 c)

AIM:

To write a Java program to connect with MySQL database and insert the given data in the table and display the table contents using ResultSet.

Input format :

The first line of the input consists of the number of students.
The next input consists of the student id, name and average_marks.

Output format :

Display the table contents as shown in the sample output.

ALGORITHM:

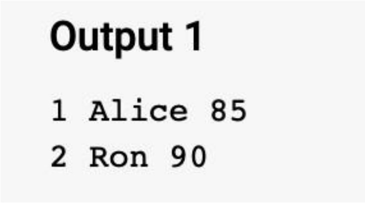
```
STEP1: Start the program
STEP2: Create a class with the name “STALL”.
STEP3: Load the Driver class
STEP4: Create a Connection object
STEP5: Create a Statement object
STEP6: Get the Connection and execute the create, retrieve, update and delete operations on an SQL Command
STEP7: Close the Connection object, Statement object. st.close(); cn.close();
```

PROGRAM:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Scanner;
class RowInsertExample{
        public static void main(String args[]) throws Exception{
                String url="jdbc:mysql://localhost/ri_db";
                String username ="test";
                String password ="test123";
                String query  =  "INSERT INTO STUDENT (id,name,average_marks) VALUES
(?,?,?);";
                Scanner sc =new Scanner(System.in);
                Class.forName("com.mysql.jdbc.Driver");
                Connection con = DriverManager.getConnection(url,username,password);
                PreparedStatement st = con.prepareStatement(query);
                int n =Integer.parseInt(sc.nextLine());
                for(int i=0;i<n;i++) {
                        int id=Integer.parseInt(sc.nextLine());
                        String name =sc.nextLine();
                        int average_marks = Integer.parseInt(sc.nextLine());
                        st.setInt(1,id);
                        st.setString(2,name);
                        st.setInt(3,average_marks);
                        st.executeUpdate(); }
                ResultSet rs =  st.executeQuery("select *from STUDENT");
                while(rs.next()) {
                        System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getInt(3)); }
                st.close();
                con.close();
        } }
```

OUTPUT:

Input 1:



Output 1

```
1 Alice 85
2 Ron 90
```

RESULT:

Thus, java program to connect with MySQL database and insert the given data in the table and display the table contents using ResultSet has been implemented successfully.