LAB FILE OF

Advance Java Programming

CIE-306P

Faculty name: Ms. Zameer Fatima

Student name: Anshu Kumar Pathak
Roll No.: 004148227222
Semester: 6th
Group: FSD – II B

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Advance Java Programming

PRACTICAL RECORD

PAPER CODE : CIE-306P

Name of the student. :Anshu Kumar Pathak

University Roll No : 004148227222

Branch : CST-1

Group : 6 FSD -I

INDEX

Exp. no	Experiment Name	Date of performance	R1 (3)	R2 (3)	R3 (3)	R4 (3)	R5 (3)	Total Marks (15)	Signature

Lab 1

Practical 1(a)

AIM: Write a program to reverse the elements of a given 2*2 array. Four integer numbers need to be passed as Command-Line arguments.

CODE:

```
public class Demo { public static void
        main(String args[]) { int arr[][] = new
        int[2][2]; int n = 0;

        for (int i = 0; i < arr.length; i++) { for (int
        j = 0; j < arr[i].length; j++) { arr[i][j] =
        Integer.parseInt(args[n]); n++;
        }
        }

        System.out.println("\nOriginal Array");
        for (int i = 0; i < arr.length; i++) { for
        (int j = 0; j < arr[i].length; j++) {
                System.out.print(arr[i][j] + " ");
        }
        System.out.println();
        }

        for (int i = 0; i < arr.length; i++) { for (int j = 0; j < arr[i].length;
        j++) { if (i * arr.length + j >= (arr.length * arr[i].length) / 2) break;
        int elem = arr[i][j]; arr[i][j] = arr[arr.length - i - 1][arr[i].length - j -
        1]; arr[arr.length - i - 1][arr[i].length - j - 1] = elem;
        }
        }

        System.out.println("\nReversed Array");
        for (int i = 0; i < arr.length; i++) { for
        (int j = 0; j < arr[i].length; j++) {
                System.out.print(arr[i][j] + " ");
        }
        System.out.println();
        }
        }
}
```

OUTPUT:


Command Prompt
D:\Desktop\LAB WORK 6TH SEM\ADV JAVA\atemp>javac Demo.java
D:\Desktop\LAB WORK 6TH SEM\ADV JAVA\atemp>java Demo 12 15 4 3
Original Array
12 15
4 3
Reversed Array
3 4
15 12
D:\Desktop\LAB WORK 6TH SEM\ADV JAVA\atemp>

Practical 1(b)

AIM: Write a java program to produce the tokens from a given long string.

CODE:

```
import java.util.StringTokenizer;
 import java.util.Scanner;

class demo{

public static void main(String args[]){ Scanner sc = new Scanner(System.in);
System.out.println("Enter the string:"); String s = sc.nextLine();
StringTokenizer st = new StringTokenizer(s, " "); System.out.println("\nTokens: ");
while(st.hasMoreTokens()){

System.out.println(st.nextToken());

}}}
```

OUTPUT:

Practical 1(c)

AIM: Using the concept of method overloading, Write method for calculating the area of triangle, circle and rectangle.

CODE:

```
class demo{

static double area(double l, double b){ return l*b;
} static double area(double s1, double s2, double s3){ double s = (s1 + s2 +

s3)/2;

return Math.sqrt(s*(s-s1)*(s-s2)*(s-s3));
}

static double area(double r){ return Math.PI * r * r;
}
public static void main(String args[]){

System.out.println("Area of Triangle with sides 3, 4, 5 : " + area(3,4,5)); System.out.println("Area of
        Rectangle with length 10 and breadth 5 : " + area(10,5)); System.out.println("Area of Circle
        with radius 3.14 : " + area(3.14));
}
}
```

OUTPUT:

```
PS C:\Users\91876\Desktop\java> javac demo.java
PS C:\Users\91876\Desktop\java> java demo
Area of Triangle with sides 3, 4, 5 : 6.0
Area of Rectangle with length 10 and breadth 5 : 50.0
Area of Circle with radius 3.14 : 30.974846927333928
PS C:\Users\91876\Desktop\java>
```

Practical 1(d)

AIM: Create a class Box that uses a parameterized constructor to initialize the dimensions of a box. The dimensions of the Box are width, height, depth. The class should have a method that can return the volume of the box. Create an object of the Box class and test the functionalities.

CODE:

```
class Box{ double width; double length; double height;

Box(double w, double l, double h){ width = w;
length = l; height = h;
}

void show(){
System.out.println("Dimenstions of box are: "); System.out.println("Width: " + width);
System.out.println("Length: " + length); System.out.println("Height: " + height);
} } class demo{ public static void main(String args[]){ Box b =

new Box(5,4,7); b.show();

}
}
```

OUTPUT:

```
PS C:\Users\91876\Desktop\java> javac demo.java
PS C:\Users\91876\Desktop\java> java demo
Dimenstions of box are:
Width: 5.0
Length: 4.0
Height: 7.0
PS C:\Users\91876\Desktop\java>
```

Practical 1(e)

AIM: Write a program that can count the number of instances created for the class.

CODE:

```
class Box{ static int objects; static{ objects
= 0; } double width; double length; double
height;

Box(double w, double l, double h){ width = w;
length = l; height = h; objects++;
System.out.println("Nex Box Created");
} } class

demo{

public static void main(String args[]){ Box box1 = new Box(5,4,7);
System.out.println("No. of instances of Box are: " + Box.objects); Box box2 = new Box(5,4,7);
System.out.println("No. of instances of Box are: " + Box.objects);

}
}
```

OUTPUT:

```
PS C:\Users\91876\Desktop\java> javac demo.java
PS C:\Users\91876\Desktop\java> java demo
Nex Box Created
No. of objects of Box are: 1
Nex Box Created
No. of objects of Box are: 2
PS C:\Users\91876\Desktop\java>
```

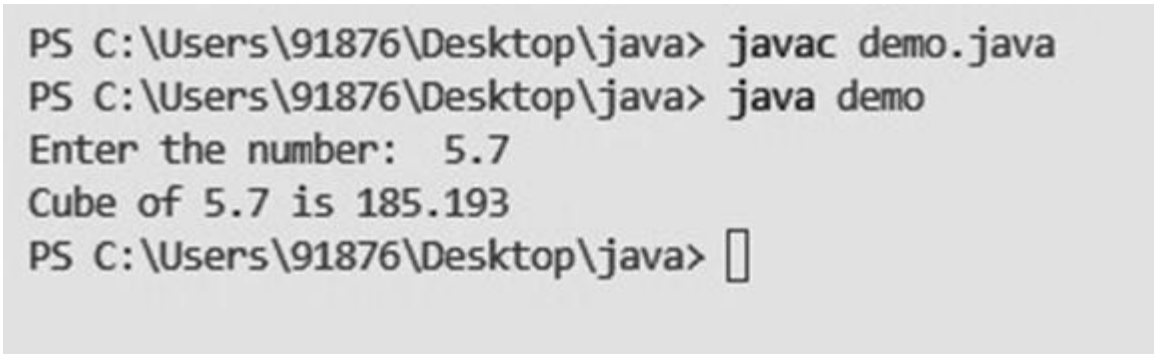
Practical 1(f)

AIM: Java Program to get the cube of a given number using the static method.

CODE:

```
import java.util.Scanner; class demo{ static
double cube(double n){ return n*n*n;
} public static void main(String args[]){ Scanner sc = new Scanner(System.in);
System.out.print("Enter
the number: "); double n = sc.nextDouble();
System.out.println("Cube of " + n + " is " + cube(n));
}
}
```

OUTPUT:



Lab 2

Practical 2(a)

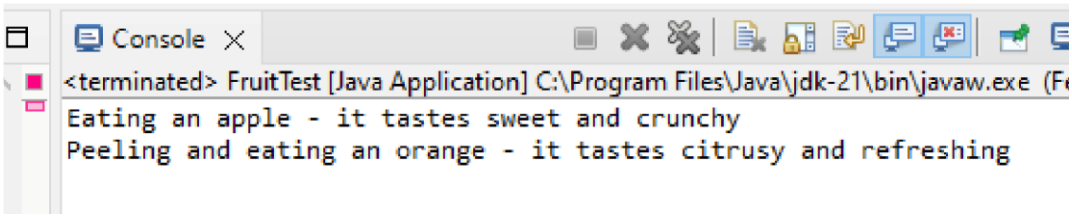
AIM: Create a base class Fruit which has name ,taste and size as its attributes. A method called eat() is created which describes the name of the fruit and its taste. Inherit the same in 2 other class Apple and Orange and override the eat() method to represent each fruit taste. (Method overriding).

CODE:

```
// Base class Fruit class
Fruit { protected String
name; protected String
taste; protected String
size;

    public Fruit(String name, String taste, String size)
        { this.name = name; this.taste = taste; this.size =
        size;
    }

    public void eat() {
        System.out.println("This is a " + name + " and it tastes " + taste);
    }
}

// Subclass Apple class Apple
extends Fruit { //
Parameterized constructor
public Apple(String size) {
        // Calling the constructor of the base class (Fruit)
        super("Apple", "sweet", size);
    }

    // Overriding the eat() method for Apple
    @Override public
    void eat() {
        System.out.println("Eating an apple - it tastes sweet and crunchy");
    }
}

// Subclass Orange class
Orange extends Fruit { //
Parameterized constructor
public Orange(String size) {
        // Calling the constructor of the base class (Fruit)
        super("Orange", "citrusy", size);
    }

    // Overriding the eat() method for Orange
    @Override
    public void eat() {
```

```
        System.out.println("Peeling and eating an orange - it tastes citrusy and refreshing");
} }

public class FruitTest {
    public static void main(String[] args) {
        // Creating objects of Apple and Orange
        Apple apple = new Apple("Medium");
        Orange orange = new Orange("Large");

        // Calling the eat() method for each
        fruit apple.eat(); orange.eat();
    }
}
```

OUTPUT:



Practical 2(b)

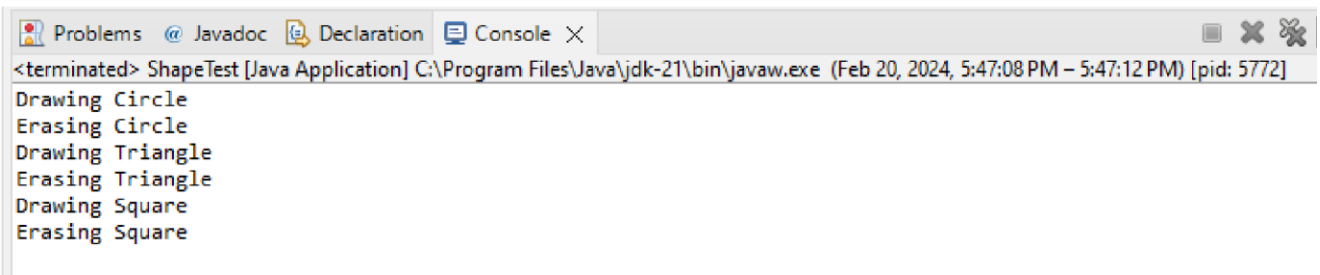
AIM: Write a program to create a class named shape. It should contain 2 methods- draw() and erase() which should print “Drawing Shape” and “Erasing Shape” respectively. For this class we have three sub classes- Circle, Triangle and Square and each class override the parent class functions- draw () and erase (). The draw() method should print “Drawing Circle”, “Drawing Triangle”, “Drawing Square” respectively. The erase() method should print “Erasing Circle”, “Erasing Triangle”, “Erasing Square” respectively. Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling draw() and erase() method using each object. Shape c=new Circle(); Shape t=new Triangle(); Shape s=new Square(); (Polymorphism)

CODE:

```
// Base class Shape
class Shape {
    // Method to draw the shape
    public void draw() {
        System.out.println("Drawing Shape");
    }

    // Method to erase the shape
    public void erase() {
        System.out.println("Erasing Shape");
    }
}

// Subclass Circle class
Circle extends Shape {
    // Overriding the draw() method for Circle
    @Override public
    void draw() {
        System.out.println("Drawing Circle");
    }

    // Overriding the erase() method for Circle
    @Override public
    void erase() {
        System.out.println("Erasing Circle");
    }
}

// Subclass Triangle class
Triangle extends Shape {
    // Overriding the draw() method for Triangle
    @Override public
    void draw() {
        System.out.println("Drawing Triangle");
    }
```

```
    // Overriding the erase() method for Triangle
    @Override public
    void erase() {
        System.out.println("Erasing Triangle");
    }
}

// Subclass Square class
Square extends Shape {
    // Overriding the draw() method for Square
    @Override public
    void draw() {
        System.out.println("Drawing Square");
    }

    // Overriding the erase() method for Square
    @Override public
    void erase() {
        System.out.println("Erasing Square");
    }
}

public class ShapeTest {
    public static void main(String[] args) {
        // Creating objects of Circle, Triangle, and Square
        Shape c = new Circle();
        Shape t = new Triangle();
        Shape s = new Square();

        c.draw();
        c.erase();

        t.draw();
        t.erase();

        s.draw();
        s.erase();
    }
}
```

OUTPUT:



Practical 2(c)

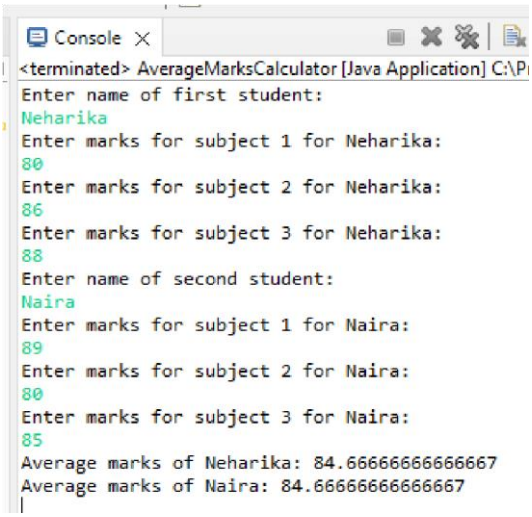
AIM: Write a Program to take care of Number Format Exception if user enters values other than integer for calculating average marks of 2 students. The name of the students and marks in 3 subjects are taken from the user while executing the program. In the same Program write your own Exception classes to take care of Negative values and values out of range (i.e. other than in the range of 0-100)

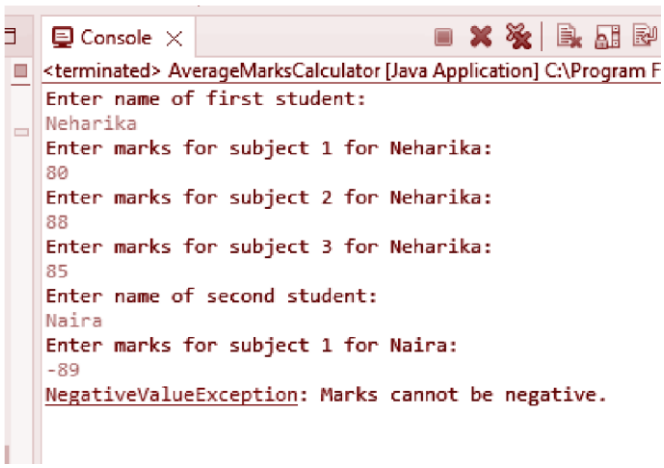
CODE:

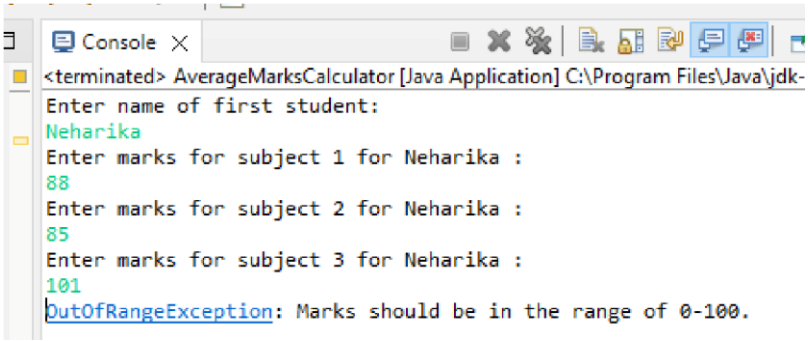
```
import java.util.Scanner;

class NegativeValueException extends Exception {
   public NegativeValueException(String message) {
   super(message);
   }}

class OutOfRangeException extends Exception {
   public OutOfRangeException(String message) {
   super(message);
   }}

public class AverageMarksCalculator { public
   static void main(String[] args) { Scanner
   scanner = new Scanner(System.in);

      try {
         System.out.println("Enter name of first student:");
         String student1Name = scanner.nextLine();

         int[] student1Marks = new int[3];
         for (int i = 0; i < 3; i++) {
            System.out.println("Enter marks for subject " + (i + 1) + " for " + student1Name +
            ":"); int mark = Integer.parseInt(scanner.nextLine()); validateMark(mark);
            student1Marks[i] = mark;
         }

         System.out.println("Enter name of second student:");
         String student2Name = scanner.nextLine();

         int[] student2Marks = new int[3];
         for (int i = 0; i < 3; i++) {
            System.out.println("Enter marks for subject " + (i + 1) + " for " + student2Name +
            ":"); int mark = Integer.parseInt(scanner.nextLine()); validateMark(mark);
            student2Marks[i] = mark;
         } double averageMarksStudent1 =
         calculateAverage(student1Marks); double averageMarksStudent2
         = calculateAverage(student2Marks);

         System.out.println("Average marks of " + student1Name + ": " + averageMarksStudent1);
         System.out.println("Average marks of " + student2Name + ": " + averageMarksStudent2);
```

```
    } catch (NumberFormatException e) {
        System.out.println("NumberFormatException: Please enter valid integer values for marks.");
    } catch (NegativeValueException e) {
        System.out.println("NegativeValueException: Marks cannot be negative.");
    } catch (OutOfRangeException e) {
        System.out.println("OutOfRangeException: Marks should be in the range of 0-100.");
    }
}

private static double calculateAverage(int[] marks) {
    double sum = 0; for
    (int mark : marks) {
        sum += mark;
    } return sum /
    marks.length;
}

private static void validateMark(int mark) throws NegativeValueException, OutOfRangeException {
    if (mark < 0) { throw new NegativeValueException("Negative marks
        not allowed.");
    }
    if (mark < 0 || mark > 100) {
        throw new OutOfRangeException("Marks should be in the range of 0-100.");
    }
}}
```

OUTPUT:







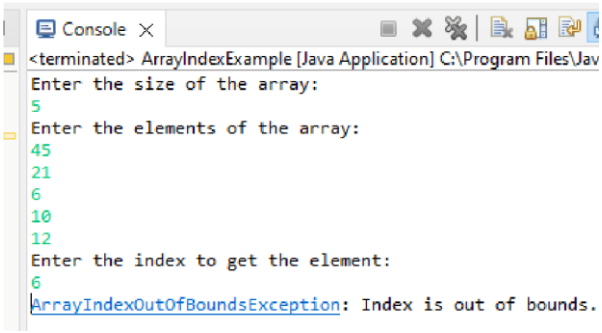
Practical 2(d)

AIM: Write a program that takes as input the size of the array and the elements in the array. The program then asks the user to enter a particular index and prints the element at that index. Index starts from zero.This program may generate Array Index Out Of Bounds Exception or NumberFormatException . Use exception handling mechanisms to handle this exception.

CODE:

```
import java.util.Scanner;

public class ArrayIndexExample { public static
   void main(String[] args) { Scanner scanner =
   new Scanner(System.in);

      try {
         System.out.println("Enter the size of the array:");
         int size = Integer.parseInt(scanner.nextLine());

         int[] array = new int[size];
         System.out.println("Enter the elements of the array:");
         for (int i = 0; i < size; i++) {
            array[i] = Integer.parseInt(scanner.nextLine());
         }

         System.out.println("Enter the index to get the

         element:"); int index =

         Integer.parseInt(scanner.nextLine()); int element =

         getElementAtIndex(array, index);

         System.out.println("Element at index " + index + " is: " + element);

      } catch (NumberFormatException e) {
         System.out.println("NumberFormatException: Please enter valid integer values.");
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("ArrayIndexOutOfBoundsException: Index is out of bounds.");
      } }
   private static int getElementAtIndex(int[] array, int index) throws ArrayIndexOutOfBoundsException {
      if (index < 0 || index >= array.length) { throw new ArrayIndexOutOfBoundsException("Index is out
      of bounds.");
      } return
      array[index];
   }}
```

OUTPUT:


Console
<terminated> ArrayIndexExample [Java Application] C:\Program Files\Jav
Enter the size of the array:
5
Enter the elements of the array:
45
21
6
10
12
Enter the index to get the element:
6
ArrayIndexOutOfBoundsException: Index is out of bounds.

Lab 3

Practical 3(a)

AIM: Write a Java program to demonstrate the concept of socket programming.

CODE:

Server.java

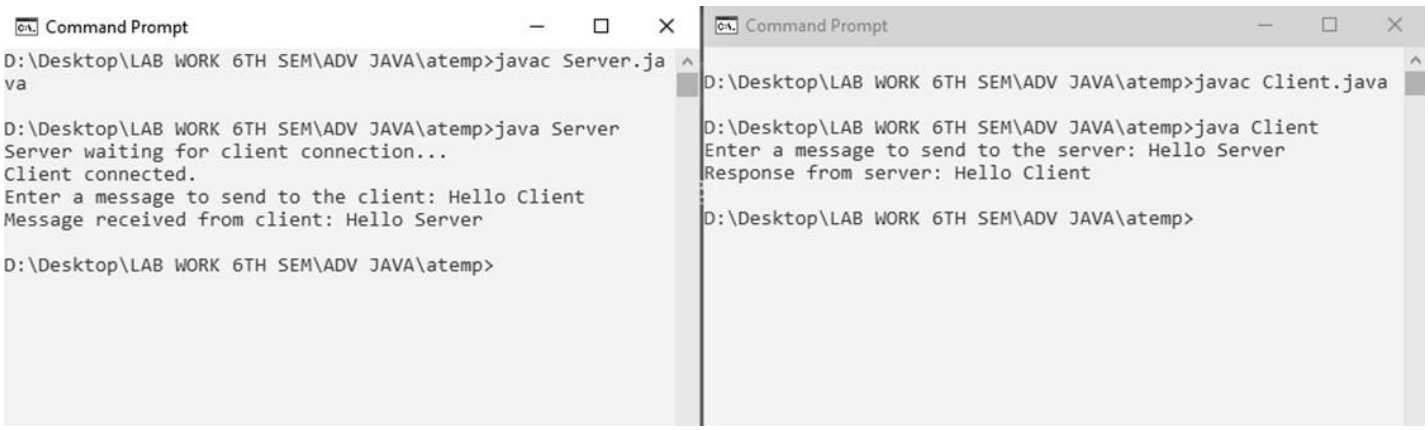
```
import java.io.BufferedReader;
import
java.io.InputStreamReader;
import java.io.PrintWriter; import
java.net.ServerSocket; import
java.net.Socket;

public class Server { public static void
   main(String[] args) { try {
         ServerSocket serverSocket = new ServerSocket(1234);
         System.out.println("Server waiting for client connection...");

         Socket clientSocket = serverSocket.accept();
         System.out.println("Client connected.");

                                          BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
         PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);

         BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
         System.out.print("Enter a message to send to the client: ");
         String serverMessage = consoleReader.readLine();
         writer.println(serverMessage);

         String clientMessage = reader.readLine();
         System.out.println("Message received from client: " + clientMessage);

         clientSocket.close();
         serverSocket.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Client.java

```
import java.io.BufferedReader;
import
java.io.InputStreamReader;
```

```
import java.io.PrintWriter; import
java.net.Socket; public class
Client {
   public static void main(String[] args) {
      try {
         Socket socket = new Socket("localhost", 1234);

         BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
         PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

         BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
         System.out.print("Enter a message to send to the server: ");
         String clientMessage = consoleReader.readLine();
         writer.println(clientMessage);

         String serverResponse = reader.readLine();
         System.out.println("Response from server: " + serverResponse);

         socket.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

OUTPUT:



Practical 3(b)

AIM: Implement Datagram UDP socket programming in java.

CODE:

Server.java

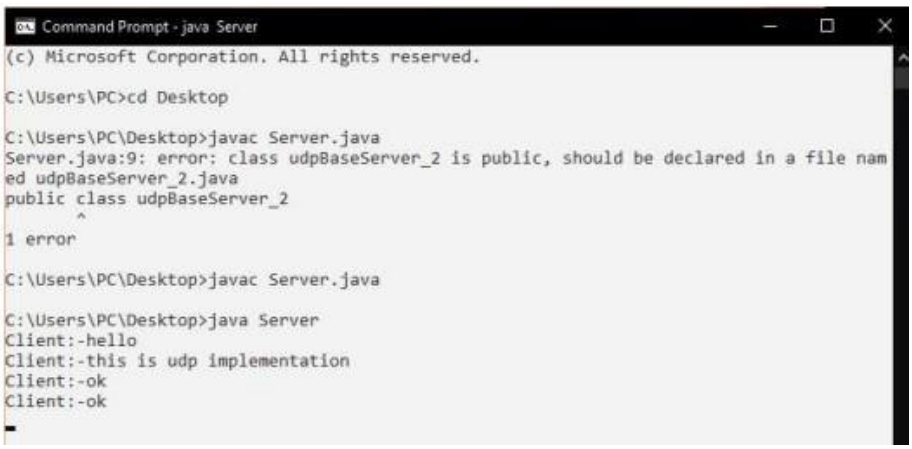
```
import java.io.BufferedReader;
import
java.io.InputStreamReader;
import java.io.PrintWriter; import
java.net.ServerSocket; import
java.net.Socket;

public class Server { public static void
   main(String[] args) { try {
         ServerSocket serverSocket = new ServerSocket(1234);
         System.out.println("Server waiting for client connection...");

         Socket clientSocket = serverSocket.accept();
         System.out.println("Client connected.");

                                          BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
         PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);

         BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
         System.out.print("Enter a message to send to the client: ");
         String serverMessage = consoleReader.readLine();
         writer.println(serverMessage);

         String clientMessage = reader.readLine();
         System.out.println("Message received from client: " + clientMessage);

         clientSocket.close();
         serverSocket.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

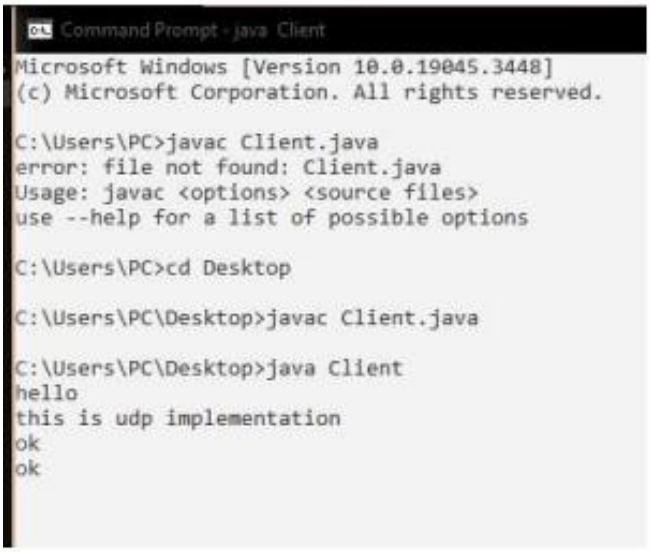
Client.java

```
import java.io.BufferedReader;
import
```

```
java.io.InputStreamReader;
import java.io.PrintWriter; import
java.net.Socket;
public class Client { public static void
   main(String[] args) { try {
         Socket socket = new Socket("localhost", 1234);

         BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
         PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

         BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
         System.out.print("Enter a message to send to the server: ");
         String clientMessage = consoleReader.readLine();
         writer.println(clientMessage);

         String serverResponse = reader.readLine();
         System.out.println("Response from server: " + serverResponse);

         socket.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

OUTPUT:





Practical 3(c)

AIM: Implement Socket programming for TCP in Java Server and Client Sockets.

CODE:

Server.java

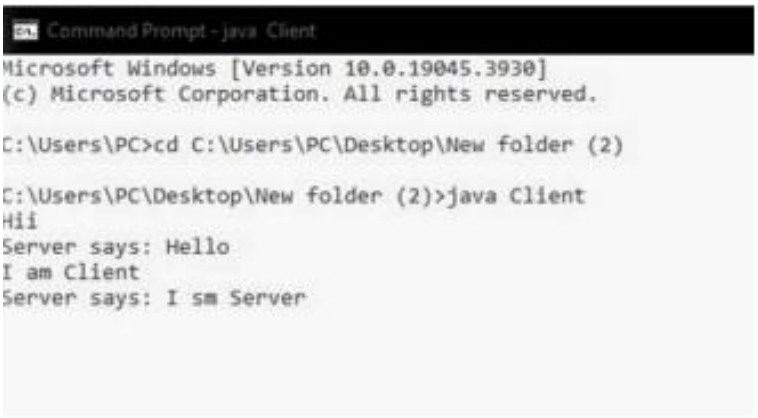
```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import
java.io.InputStreamReader;
import java.net.*; public class
Server {
public static void main(String arg[]) throws Exception {
ServerSocket ss = new ServerSocket(8080);
Socket s = ss.accept();
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str ="",str2="";
while(!str.equals("stop")){ str
= din.readUTF();
System.out.println("Client says: "+str);
str2 = br.readLine();
dout.writeUTF(str2); dout.flush();
}
dout.close();
din.close();
s.close(); ss.close();
}
}
```

Client.java

```
import java.net.*;
import java.io.*;
public class Client {
public static void main(String arg[]) throws Exception {
Socket s = new Socket("localhost", 8080);
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str ="",str2="";
while(!str.equals("stop")){
str = br.readLine();
dout.writeUTF(str);
dout.flush(); str2 =
din.readUTF();
System.out.println("Server says: "+str2);
}
```

```
dout.close();
din.close(); s.close();
}
}
```

OUTPUT:



Lab 4

Practical 4(a)

AIM: Implement a program Java Bean to represent person details.

CODE:

Person.java

```
package lab4; import
java.io.Serializable;

public class Person implements Serializable {
   private String firstName;
   private String lastName;
   private String email;

   // Default constructor
   public Person() {
   }

   public Person(String firstName, String lastName, String email)
      { this.firstName = firstName; this.lastName = lastName;
      this.email = email;
   }
   // Getter and Setter methods
   public String getFirstName() {
   return firstName;
   } public void setFirstName(String
   firstName) { this.firstName = firstName;
   }

   public String getLastName() {
      return lastName;
   } public void setLastName(String
   lastName) { this.lastName = lastName;
   }

   public String getEmail() {
      return email;
   } public void setEmail(String
   email) { this.email = email;
   }

   // Override toString method for a meaningful representation of the object
   @Override public String
   toString() {
      return "Person [firstName=" + firstName + ", lastName=" + lastName + ", email=" + email + "]";
   }
}
```

Main.java

```
package lab4;
public class Main {
   public static void main(String[] args) {

      // Create a new Person object
      Person person = new Person();

      // Set properties using setter methods
      person.setFirstName("Anshu");

       person.setLastName("Pathak");
      person.setEmail("anshu@gmail.com");

      // Retrieve and print values using getter methods
      System.out.println("First Name: " + person.getFirstName());
      System.out.println("Last Name: " + person.getLastName());
      System.out.println("Email: " + person.getEmail());

      // Print the entire object using toString method
      System.out.println(person);
   }
}
```

OUTPUT:

```
First Name : Anshu
Last Name : Pathak
Email : anshu@gmail.com
Person [FirstName = Anshu, LastName = Pathak ,email=anshu@gmail.com ]
```

Practical 4(b)

AIM: Write a program in java to demonstrate encapsulation in java beans.

CODE:

Car.java

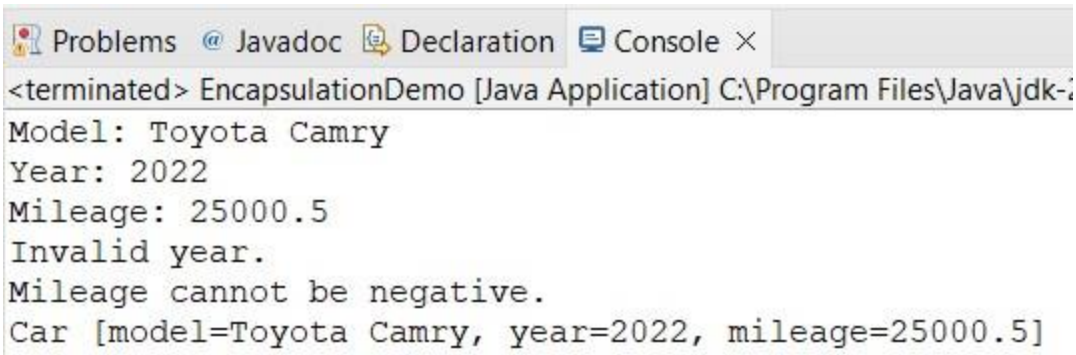
```
package lab4; import java.io.Serializable;
public class Car implements Serializable
{
  private String model;
  private int year; private
  double mileage; //
  Default constructor
  public Car() {
  }
  // Parameterized constructor public Car(String
  model, int year, double mileage) {
     this.model = model;
     this.year = year;
     this.mileage = mileage;
  }
  // Getter and Setter methods
  public String getModel() {
     return model; } public void
  setModel(String model) {
     this.model = model;
  }

  public int getYear() {
     return year;
  }
  public void setYear(int year) {
     if (year > 0) {
        this.year = year;
     } else {
        System.out.println("Invalid year.");
     } } public double
  getMileage() {
     return mileage;
  } public void setMileage(double mileage) { // Adding
  a check for the mileage to be non-negative if (mileage
  >= 0) {
        this.mileage = mileage;
     } else {
        System.out.println("Mileage cannot be negative.");
     }
  }
  @Override public String
  toString() { return "Car
```

```
  [model=" + model + ",
  year=" + year + ",
  mileage=" + mileage +
  "]"; }
}
```

EncapsulationDemo.java

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

      // Create a new Car object Car
      car = new Car();
      car.setModel("Toyota
      Camry"); car.setYear(2022);
      car.setMileage(25000.5);

      System.out.println("Model: " + car.getModel());
      System.out.println("Year: " + car.getYear());
      System.out.println("Mileage: " + car.getMileage());

      // Try to set an invalid year and negative mileage (will be
      ignored) car.setYear(0); car.setMileage(-500.75);

      // Print the entire object using toString method
      System.out.println(car);
  }
}
```

OUTPUT:



Lab 5

Practical 5(a)

AIM: Write a JSP program to display your name.

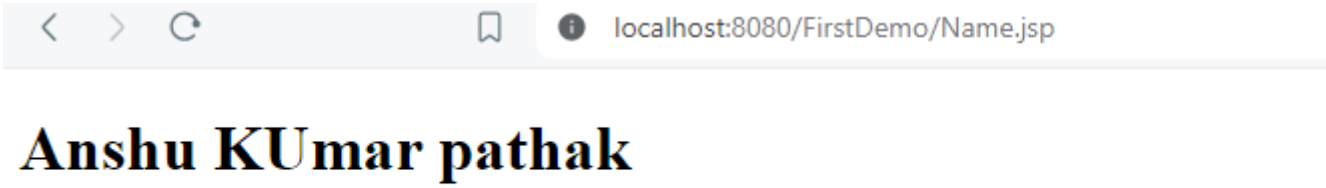
PROCEDURE:

1. Create a new file with a .jsp extension, here it is “index.jsp.”
2. Write JSP code to display your name and save the file.
3. Start your Tomcat server by right clicking configured Tomcat Server.
4. Now your JSP code will be visible on the web browser.

CODE: **index.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> First JSP </title>
</head>
<body>
<h1> Anshu KUmar pathak</h1>
</body>
</html>
```

OUTPUT:



Practical 5(b)

AIM: Write a JSP program to implement form data validation.

CODE: **formValidation.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Form Data Validation</title>
  <script> function validateForm() { var name =
    document.forms["myForm"]["name"].value; var email =
    document.forms["myForm"]["email"].value; if (name ===
    "" || email === "") { alert("Both name and email must be
    filled out"); return false;
      } return
      true;
    }
  </script>
</head>
<body>
  <h2>Form Data Validation</h2>
      <form name="myForm" action="processFormData.jsp" onsubmit="return validateForm()"
method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br><br>
    <label for="email">Email:</label>
    <input type="text" id="email" name="email"><br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

processFormData.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Form Data Processing</title>
</head>
<body>
  <h2>Form Data Processing</h2>
  <%-- Retrieve and display form data --%>
  <p>Name: <%= request.getParameter("name") %></p>
  <p>Email: <%= request.getParameter("email") %></p>
</body>
</html>
```

OUTPUT: