Practical: 1

Aim: Set up Java Programming development environment by using

i. Command Prompt

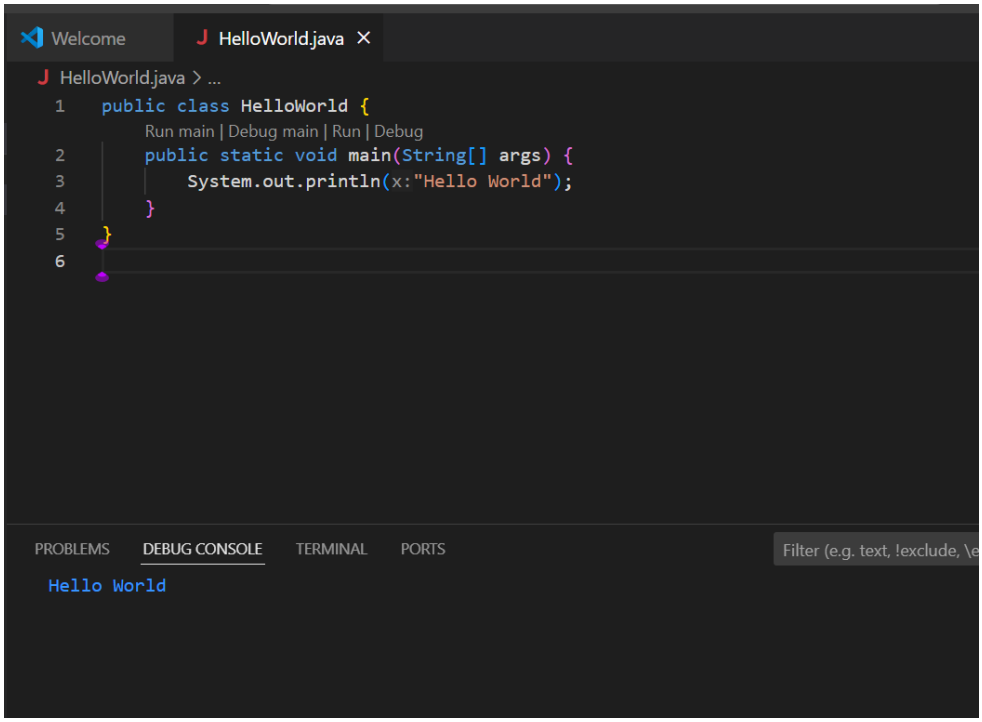
ii. Any IDE like Eclipse, Notepad++, JCreater etc.

1. And Test Java Programming development environment by implementing a small program.

Source Code:

```
public class HelloWord {
   public static void main(String[] args) {
      System.out.println("Hello World");
   }
}
```

Output:



Practical: 2

Aim: Implementing the Operations of stack and queue using package and interface.

Source Code: 1 : Queue

```
import java.util.LinkedList;
import java.util.Queue;
public class QueueDemo {
    public static void main(String[] args){
        Queue<String> queue = new LinkedList<>();
        queue.add("JAVA");
        queue.add("DBMS");
        queue.add("CN");
        queue.add("OS");
        System.out.println("Queue: " + queue);
        String front = queue.remove();
        System.out.println("Removed Element: " + front);
        System.out.println("Queue after removal: " + queue);;
        queue.add("date");
        String peeked = queue.peek();
        System.out.println("Peeked element: " + peeked);
        System.out.println("Queue after peek: " + queue);
    }
}
```

Output:

```
Queue: [JAVA, DBMS, CN, OS]
Removed Element: JAVA
Queue after removal: [DBMS, CN, OS]
Peeked element: DBMS
Queue after peek: [DBMS, CN, OS, date]
```

Source Code: 2 : Stack

```
import java.util.Stack;
public class Stack1 {
    public static void main(String[] args){
        Stack<Integer> s = new Stack<>();
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);
        System.out.println("Data Stored: ");
        while(!s.isEmpty()){
            System.out.println(s.pop());
        }
    }
}
```

Output:

```
Data Stored:
4
3
2
1
```

Practical: 3

Aim: Write a program to implement an object oriented system and multithreaded processes as per needs and specifications.

Source Code:

```
class Task implements Runnable {
    private final String name;
    public Task(String name) {
        this.name = name;
    }
    public void run() {
        System.out.println(Thread.currentThread().getName() + " is executing " + name);
        try {
            Thread.sleep(1000); // Simulate task duration
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
public class Main {
    public static void main(String[] args) {
        // Create and start multiple threads (multithreading)
        for (int i = 1; i <= 5; i++) {
            Thread thread = new Thread(new Task("Task-" + i), "Worker-" + i);
            thread.start();
        }
    }
}
```

Output:

```
Worker-4 is executing Task-4
Worker-3 is executing Task-3
Worker-1 is executing Task-1
Worker-5 is executing Task-5
Worker-2 is executing Task-2
```

Practical: 4

Aim: Write a program to implement student information in a file and perform the operations on it.

Source Code:

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

class Student {
    private String rollNo;
    private String name;
    private int age;

    public Student(String rollNo, String name, int age) {
        this.rollNo = rollNo;
        this.name = name;
        this.age = age;
    }

    public String toFileString() {
        return rollNo + "," + name + "," + age;
    }

    public static Student fromFileString(String line) {
        try {
            String[] parts = line.split(",");
            if (parts.length != 3) return null;
            return new Student(parts[0], parts[1], Integer.parseInt(parts[2]));
        } catch (Exception e) {
            // In case of corrupted or invalid data
            return null;
        }
```

```
    }

    public String getRollNo() {
        return rollNo;
    }

    @Override
    public String toString() {
        return "Roll No: " + rollNo + ", Name: " + name + ", Age: " + age;
    }
}

public class StudentApp {
    static final String FILE_NAME = "students.txt";

    public static void addStudent(Student s) throws IOException {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(FILE_NAME, true))) {
            bw.write(s.toFileString());
            bw.newLine();
        }
    }

    public static void showAll() throws IOException {
        File file = new File(FILE_NAME);
        if (!file.exists()) {
            System.out.println("No records found.");
            return;
        }

        boolean found = false;
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
```

```
        while ((line = br.readLine()) != null) {
            Student s = Student.fromFileString(line);
            if (s != null) {
                System.out.println(s);
                found = true;
            }
        }
    }

    if (!found) {
        System.out.println("No valid student records to display.");
    }
}

public static void searchStudent(String rollNo) throws IOException {
    boolean found = false;
    try (BufferedReader br = new BufferedReader(new FileReader(FILE_NAME))) {
        String line;
        while ((line = br.readLine()) != null) {
            Student s = Student.fromFileString(line);
            if (s != null && s.getRollNo().equalsIgnoreCase(rollNo)) {
                System.out.println("Found: " + s);
                found = true;
                break;
            }
        }
    }
    if (!found) System.out.println("Student not found.");
}

public static void main(String[] args) throws IOException {
    Scanner sc = new Scanner(System.in);
```

```
while (true) {
    System.out.println("\n===== Student Information System =====");
    System.out.println("1. Add Student");
    System.out.println("2. Show All Students");
    System.out.println("3. Search Student by Roll No");
    System.out.println("4. Exit");
    System.out.print("Enter your choice: ");

    int choice;
    try {
        choice = Integer.parseInt(sc.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Please enter a valid number.");
        continue;
    }

    switch (choice) {
        case 1:
            System.out.print("Enter Roll No: ");
            String roll = sc.nextLine();
            System.out.print("Enter Name: ");
            String name = sc.nextLine();
            System.out.print("Enter Age: ");
            int age;
            try {
                age = Integer.parseInt(sc.nextLine());
            } catch (NumberFormatException e) {
                System.out.println("Invalid age input.");
                break;
            }
            addStudent(new Student(roll, name, age));
            System.out.println("Student added successfully.");
```

```
                break;

            case 2:
                showAll();
                break;

            case 3:
                System.out.print("Enter Roll No to search: ");
                String search = sc.nextLine();
                searchStudent(search);
                break;

            case 4:
                System.out.println("Exiting the program. Goodbye!");
                return;

            default:
                System.out.println("Invalid option. Please try again.");
        }
    }
  }
}
```

Text File: student.txt

169, Yash, 21

164, Vishal, 22

166, Pranali, 21

Output:

```
===== Student Information System =====
1. Add Student
2. Show All Students
3. Search Student by Roll No
4. Exit
Enter your choice: 1
Enter Roll No: 169
Enter Name: Yash
Enter Age: 21
Student added successfully.
```

```
===== Student Information System =====
1. Add Student
2. Show All Students
3. Search Student by Roll No
4. Exit
Enter your choice: 1
Enter Roll No: 164
Enter Name: Vishal
Enter Age: 22
Student added successfully.
```

```
===== Student Information System =====
1. Add Student
2. Show All Students
3. Search Student by Roll No
4. Exit
Enter your choice: 1
Enter Roll No: 166
Enter Name: Pranali
Enter Age: 21
Student added successfully.
```

```
===== Student Information System =====
1. Add Student
2. Show All Students
3. Search Student by Roll No
4. Exit
Enter your choice: 2
Roll No: 169, Name: Yash, Age: 21
Roll No: 164, Name: Vishal, Age: 22
Roll No: 166, Name: Pranali, Age: 21
```

```
===== Student Information System =====
1. Add Student
2. Show All Students
3. Search Student by Roll No
4. Exit
Enter your choice: 3
Enter Roll No to search: 169
Found: Roll No: 169, Name: Yash, Age: 21
```

```
===== Student Information System =====
1. Add Student
2. Show All Students
3. Search Student by Roll No
4. Exit
Enter your choice: 4
Exiting the program. Goodbye!
PS E:\java\practical journal>
```

Practical: 5

Aim: Working with shape motion by Applet programming.

Source Code:

<u>Practical5.java</u> :

```
import java.applet.Applet;
import java.awt.*;
public class Practical5 extends Applet implements Runnable {
    int x = 0;
    Thread t;
    Image bufferImage;
    Graphics bufferGraphics;

    public void init() {
        setSize(1200, 600); // Bigger window
        setBackground(Color.WHITE);
        bufferImage = createImage(getWidth(), getHeight());
        bufferGraphics = bufferImage.getGraphics();
    }

    public void start() {
        t = new Thread(this);
        t.start();
    }
    public void run() {
        while (true) {
            x += 10; // Faster motion since we have more space
            if (x > getWidth()) {
                x = -150; // Start from left off-screen
            }
            repaint();
            try {
                Thread.sleep(30);
```

```
            } catch (InterruptedException e) {
                break;
            }
        }
    }
    public void update(Graphics g) {
        paint(g);
    }
    public void paint(Graphics g) {
        bufferGraphics.setColor(getBackground());
        bufferGraphics.fillRect(0, 0, getWidth(), getHeight());

        bufferGraphics.setColor(Color.BLUE);
        bufferGraphics.fillOval(x, 225, 150, 150); // Bigger circle, adjusted Y

        g.drawImage(bufferImage, 0, 0, this);
    }
    public void stop() {
        t = null;
    }
}
```

Practical5.html :

```
<html>
    <body>
        <applet code="Practical5.class" width="400" height="200">
        </applet>
    </body>
</html>
```

Cmd :

```
appletviewer Practical5.html
```

Output:

Practical: 6

Aim: Write a program to design Registration process form using Applet and AWT components.

Source Code:

<u>Practical6.java</u> :

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Practical6 extends Applet implements ActionListener {
    Label nameLabel, genderLabel, courseLabel, outputLabel;
    TextField nameField;
    CheckboxGroup genderGroup;
    Checkbox male, female;
    Choice courseChoice;
    Button submitButton;
    public void init() {
        setLayout(null); // We'll manually set positions
        setBackground(Color.LIGHT_GRAY);
        nameLabel = new Label("Name:");
        nameLabel.setBounds(50, 50, 80, 25);
        add(nameLabel);
        nameField = new TextField();
        nameField.setBounds(150, 50, 200, 25);
        add(nameField);

        genderLabel = new Label("Gender:");
        genderLabel.setBounds(50, 100, 80, 25);
        add(genderLabel);

        genderGroup = new CheckboxGroup();
        male = new Checkbox("Male", genderGroup, false);
        female = new Checkbox("Female", genderGroup, false);
```

```
        male.setBounds(150, 100, 80, 25);
        female.setBounds(240, 100, 80, 25);
        add(male);
        add(female);

        courseLabel = new Label("Course:");
        courseLabel.setBounds(50, 150, 80, 25);
        add(courseLabel);

        courseChoice = new Choice();
        courseChoice.add("Computer Science");
        courseChoice.add("Information Technology");
        courseChoice.add("Electronics");
        courseChoice.add("Mechanical");
        courseChoice.setBounds(150, 150, 200, 25);
        add(courseChoice);

        submitButton = new Button("Submit");
        submitButton.setBounds(150, 200, 100, 30);
        submitButton.addActionListener(this);
        add(submitButton);

        outputLabel = new Label("");
        outputLabel.setBounds(50, 250, 400, 25);
        add(outputLabel);
    }
    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String gender = genderGroup.getSelectedCheckbox() != null ?
genderGroup.getSelectedCheckbox().getLabel() : "Not selected";
        String course = courseChoice.getSelectedItem();
        outputLabel.setText("Registered: " + name + " | " + gender + " | " + course);
```

```
    }
}
```

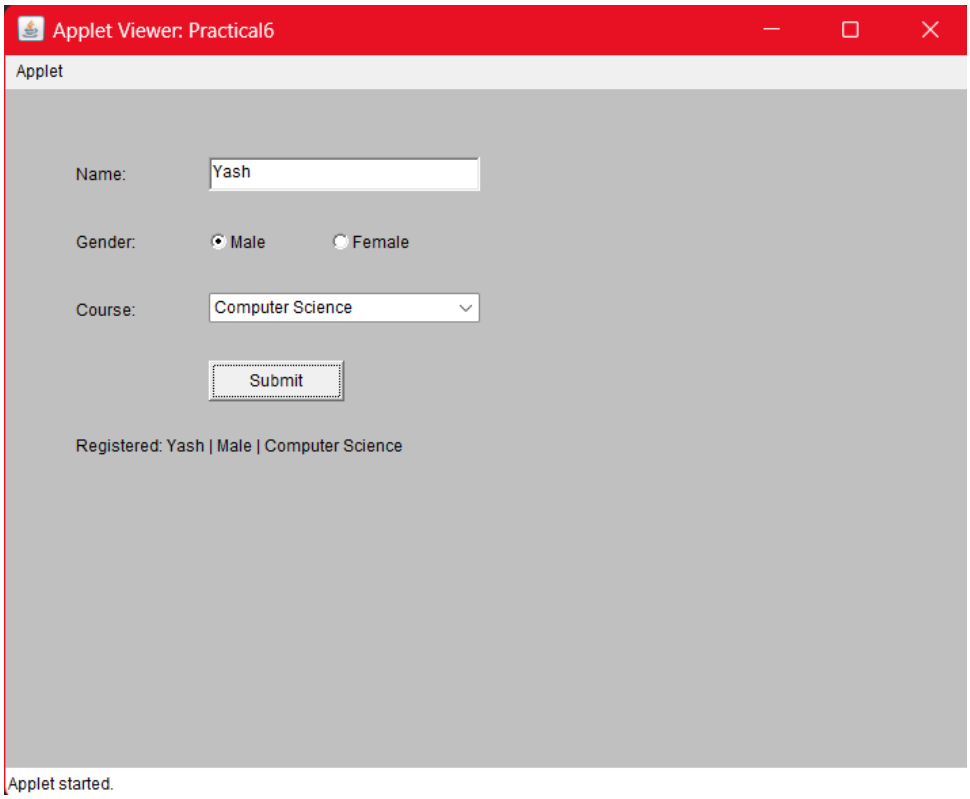
<u>Practical6.html</u> :

```
<html>
  <body>
    <applet code="Practical6" width="500" height="500"></applet>
  </body>
</html>
```

<u>Cmd</u> :

appletviewer Practical6.html

OutPut:



Practical: 7

Aim : Write a program to connect to any database and to execute the SQL query operation on command prompt.

MySQL :

```
CREATE DATABASE testdb;

USE testdb;

CREATE TABLE students (

   id INT PRIMARY KEY,

   name VARCHAR(50),

   age INT );

insert into students value(14, "Rutuja", 22), (66, "Pranali", 21), (69, "Yash", 21);
```

```
mysql> insert into students value(14, "Rutuja", 22),
    -> (66, "Pranali", 21),
    -> (69, "Yash", 21);
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

DBQueryExample.java :

```
import java.sql.*;

public class DBQueryExample {

   public static void main(String[] args) {

      String url = "jdbc:mysql://localhost:3306/testdb";

       String user = "javauser";

       String password = "javapass";

       try {

          Class.forName("com.mysql.cj.jdbc.Driver");

          Connection conn = DriverManager.getConnection(url, user, password);

          System.out.println("Connected to database!");

          Statement stmt = conn.createStatement();

          ResultSet rs = stmt.executeQuery("SELECT * FROM students");

          while (rs.next()) {

             int id = rs.getInt("id");

             String name = rs.getString("name");
```

```
            int age = rs.getInt("age");
            System.out.println("ID: " + id + ", Name: " + name + ", Age: " + age);
        }
        rs.close();
        stmt.close();
        conn.close();
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }  }  }
```

<u>CMD</u> :

E:

cd JavaProgram

javac -cp .;mysql-connector-j-9.2.0.jar DBQueryExample.java

java -cp .;mysql-connector-j-9.2.0.jar DBQueryExample

```
E:\JavaProgram>javac -cp .;mysql-connector-j-9.2.0.jar DBQueryExample.java

E:\JavaProgram>java -cp .;mysql-connector-j-9.2.0.jar DBQueryExample
Connected to database!
ID: 14, Name: Rutuja, Age: 22
ID: 66, Name: Pranali, Age: 21
ID: 69, Name: Yash, Age: 21
```

Practical: 8

Aim : Write a program to connect to any database and to execute the SQL query operation using GUI Interface

MySQL :

```
CREATE DATABASE testdb;

USE testdb;

CREATE TABLE students (id INT PRIMARY KEY, name VARCHAR(50),  age INT );

insert into students value(14, "Rutuja", 22), (66, "Pranali", 21), (69, "Yash", 21);
```

```
mysql> insert into students value(14, "Rutuja", 22),
    -> (66, "Pranali", 21),
    -> (69, "Yash", 21);
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

DatabaseGUI.java :

```
import javax.swing.*;

import javax.swing.table.DefaultTableModel;

import java.awt.*;

import java.sql.*;

public class DatabaseGUI extends JFrame {

   private JTable table;

   private JButton loadButton;

   public DatabaseGUI() {

      setTitle("Database Query GUI");

      setSize(600, 400);

      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      setLayout(new BorderLayout());

      loadButton = new JButton("Load Data");

      table = new JTable();

      add(loadButton, BorderLayout.NORTH);

      add(new JScrollPane(table), BorderLayout.CENTER);

      loadButton.addActionListener(e -> loadDataFromDatabase());       }

   private void loadDataFromDatabase() {

      String url = "jdbc:mysql://localhost:3306/testdb";
```

```
        String user = "javauser";       // Replace with your MySQL user
        String password = "javapass";    // Replace with your MySQL password
        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            String query = "SELECT * FROM students";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            ResultSetMetaData metaData = rs.getMetaData();
            int columns = metaData.getColumnCount();
            DefaultTableModel model = new DefaultTableModel();
            for (int i = 1; i <= columns; i++) {
                model.addColumn(metaData.getColumnName(i));          }
            while (rs.next()) {
                Object[] row = new Object[columns];
                for (int i = 1; i <= columns; i++) {
                    row[i - 1] = rs.getObject(i);    }
                model.addRow(row);       }
            table.setModel(model);     }
    catch (SQLException ex) {
      JOptionPane.showMessageDialog(this, "Database error: " + ex.getMessage());       }  }
       public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            DatabaseGUI gui = new DatabaseGUI();
            gui.setVisible(true);        });  }   }
```

CMD :

```
javac -cp .;mysql-connector-j-9.2.0.jar DatabaseGUI.java
java -cp .;mysql-connector-j-9.2.0.jar DatabaseGUI
```

Load Data

id	name	age
14	Rutuja	22
66	Pranali	21
69	Yash	21

Practical: 9

Aim : Write a program to demonstrate socket programming. E.g. send hello world to server from client.

Server.java :

```
import java.io.*;
import java.net.*;
public class Server {
   public static void main(String[] args) {
      int port = 1234;
      try (ServerSocket serverSocket = new ServerSocket(port)) {
         System.out.println("Server started. Waiting for client...");
         Socket clientSocket = serverSocket.accept();
         System.out.println("Client connected!");
 BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
         String message = in.readLine();
         System.out.println("Received from client: " + message);

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

Client.java :

```
import java.io.*;
import java.net.*;
public class Client {
   public static void main(String[] args) {
      String host = "localhost";
      int port = 1234;
      try (Socket socket = new Socket(host, port)) {
         PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
         out.println("Hello, World");
         System.out.println("Message sent to server.");
```

```
    } catch (IOException e) {
        e.printStackTrace();    }    }   }
```

<u>CMD</u> :

java Server

```
E:\socketProgramming>java Server
Server started. Waiting for client...
```

<u>New cmd</u> :

java Client

```
E:\socketProgramming>java Client
Message sent to server.
```

```
Client connected!
Received from client: Hello, World
```

Practical: 10

Aim: Write a Servlet code to demonstrate GET and POST methods with suitable example.

Source Code:

<u>Practical10</u> :

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Practical10 extends HttpServlet {
   public void doGet(HttpServletRequest request, HttpServletResponse response)
         throws IOException, ServletException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<h2>Hello from GET method!</h2>");
      out.println("<form method='POST'>");
      out.println("Name: <input type='text' name='username'/>");
      out.println("<input type='submit' value='Send via POST'/>");
      out.println("</form>");    }
   public void doPost(HttpServletRequest request, HttpServletResponse response)
         throws IOException, ServletException {
      String name = request.getParameter("username");
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<h2>Hello, " + name + " (from POST method)!</h2>");    }   }
```

Folder Structure in Tomcat

```
apache-tomcat-9.0.102
        └── webapps
                └── MyApp
                        ├── Practical10.java (compiled separately)
                        └── WEB-INF
                                ├── web.xml
```

└── classes

└── Practical10.class

<u>Web.xml</u> :

Located at: webapps/MyApp/WEB-INF/web.xml

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
        version="3.1">
    <servlet>
        <servlet-name>Practical10</servlet-name>
        <servlet-class>Practical10</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Practical10</servlet-name>
        <url-pattern>/prac10</url-pattern>
    </servlet-mapping>
</web-app>
```

Compile Servlet

<u>Open CMD and navigate to where your .java file is saved, then run:</u>

javac -cp "C:\apache-tomcat-9.0.102\lib\servlet-api.jar" Practical10.java

<u>Copy the generated Practical10.class into:</u>

C:\apache-tomcat-9.0.102\webapps\MyApp\WEB-INF\classes\

Run and Access the Servlet

1. <u>Start Tomcat by running startup.bat from:</u>

   C:\apache-tomcat-9.0.102\bin

2. <u>Open browser and go to:</u>

http://localhost:8080/MyApp/prac10

Output :

Practical: 11

Aim: Write a program to demonstrate the use of JSP.

Source Code:

```
apache-tomcat-9.0.102
        └── webapps
                └── MyJspApp
                        ├── index.jsp
                        └── greet.jsp
```

<u>Index.jsp</u> :

```
<html>
<head>
   <title>JSP Greeting Form</title>
</head>
<body>
   <h2>Enter Your Name</h2>
   <form action="greet.jsp" method="post">
      Name: <input type="text" name="username" />
      <input type="submit" value="Greet Me" />
   </form>
</body>
</html>
```

<u>Greet.jsp</u> :

```
<%
   String name = request.getParameter("username");
%>
<html>
<head>
   <title>Greeting Result</title>
</head>
```

```
<body>
    <h2>Hello, <%= name %>! Welcome to JSP.</h2>
</body>
</html>
```

Cmd :

http://localhost:8080/MyJspApp/index.jsp

Output :

Practical: 12

Aim: Write a java program to print a pyramid number pattern as follows:

0
101
21012
3210123
432101234
54321012345
6543210123456
765432101234567
87654321012345678
9876543210123456789

Source code :

```
public class Practical12{
   public static void main(String[] args) {
      int rows = 10;
      for (int i = 0; i < rows; i++) {
         // Print leading spaces
         for (int s = 0; s < rows - i - 1; s++) {
            System.out.print("  ");     }
         // Print decreasing numbers from i to 0
         for (int j = i; j >= 0; j--) {
            System.out.print(j + " ");    }
         // Print increasing numbers from 1 to i
         for (int j = 1; j <= i; j++) {
            System.out.print(j + " ");     }
         System.out.println();   }}}
```

Output :

```
                    0
                  1 0 1
                2 1 0 1 2
              3 2 1 0 1 2 3
            4 3 2 1 0 1 2 3 4
          5 4 3 2 1 0 1 2 3 4 5
        6 5 4 3 2 1 0 1 2 3 4 5 6
      7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
    8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8
  9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9
PS E:\java\practical journal>
```