

# **MAHARSHI DAYANAND UNIVERSITY, ROHTAK**



**Department of Computer Science & Applications**

## **Practical Assignment-1**

**(Session 2023-24)**

**BASED ON**

**SOFTWARE LAB – 3 : 20MCA22C1**

**Submitted By:**

**Juhi**

**MCA - 1<sup>st</sup> year**

**Roll No. : 23170**

**Submitted To:**

**Dr.Gopal Singh**

**(Associate Professor)**

# INDEX

Sr. No.	PROGRAMS	PAGE NO.	REMARKS
1.	Write a program in Java to implement print to develop a window using an Applet.	1,2	
2.	Write a program to generate Form using HTML & JAVASCRIPT.	3 to 7	
3.	Write a program to implement Event and AWT components. a) Button b) Checkbox	8 to 12	
4.	Write a program to implement Swing components. a) Button b) Table c) Tree d) Checkbox Pane	13 to 19	
5.	Write a program to implement Swing components. a) Tabbed Pane b) Scroll Pane	20 to 23	
6.	Write a program in Java to implement all the phases of life cycle of Servlet.	24,25	
7.	Write a program in Java to show implement DHTML and CSS with javascript.	26 to 28	
8.	How is role of server side is different from client side in a typical website? Clear using an example.	29 to 33	

9.	Write a program in Java to using JSP which accept two integer numbers from user and display the result.	34,35	
10.	Write a program in Java using POST and GET method in Swing.	36 to 39	
11.	Write a Javascript Program to check number entered is an Armstrong number or not.	40 to 42	
12.	Write a Javascript program to create a Login Form and validate it.	43 to 46	
13.	Write a program to implement Event and AWT components. a) CANVAS b) SCROLLBAR	47,48	
14.	Write a program using JSP to implement the Scripting Elements.	49 to 51	
15.	Write a program using JSP to implement any five Implicit Objects.	52 to 54	

# **1. Write a program to develop a window using an Applet.**

## **RectangleWindowApplet.java:-**

:

```
import java.applet.Applet;
import java.awt.*;
public class RectangleWindow extends Applet {
    public void paint(Graphics g) {

        // Set font style and font name
        Font font = new Font("Arial", Font.BOLD |
                               Font.ITALIC, 18);
        g.setFont(font);

        // Draw rectangle
        g.setColor(Color.BLUE);
        g.fillRect(50, 50, 300, 200);

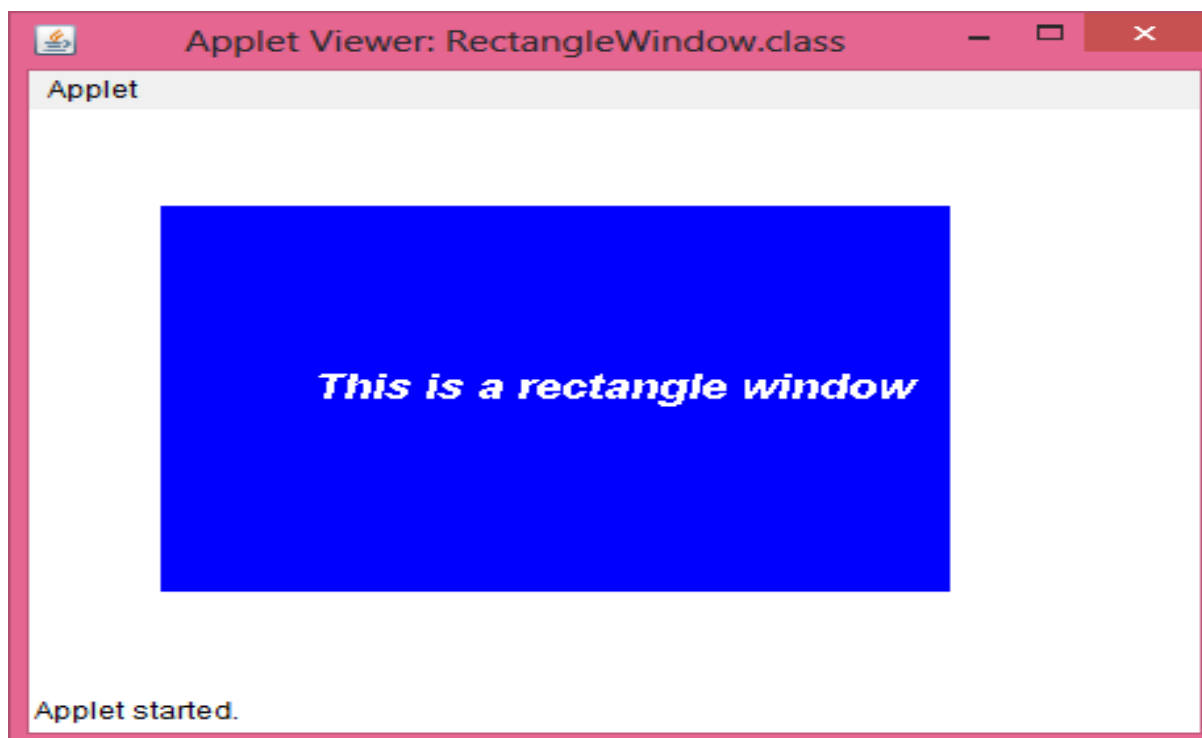
        // Set font color
        g.setColor(Color.WHITE);

        // Draw text
        String text = "This is a rectangle window";
        FontMetrics fm = g.getFontMetrics();
        int textWidth = fm.stringWidth(text);
        int x = (getWidth() - textWidth) / 2;
        // Center the text horizontally
        int y = getHeight() / 2;
        // Center the text vertically
        g.drawString(text, x, y);
    }
}
```

## RectangleWindowApplet.html:-

```
<!DOCTYPE html>
<html>
<head>
<title>Rectangle Window Applet</title>
</head>
<body>
  <h1>Rectangle Window Applet Example</h1>
  <hr>
  <applet code="RectangleWindow.class" width="400"
          height="300">
    </applet>
</body>
</html>
```

## Output:-



## **2. Write a program to generate Form using HTML & JAVASCRIPT.**

### **Form.html:-**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>College Student Details Form</title>
<style>

body {
  font-family: Arial, sans-serif;
  background:linear-gradient(45deg,dodgerblue,white,purple);
  margin: 0;
  padding: 20px;
  box-sizing: border-box;
}
.form-container {
  background-color: #fff;
  padding: 30px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  max-width: 500px;
  margin: auto;
}
h2 {
  color: #007bff;
  text-align: center;
  margin-bottom: 30px;
}
.form-group {
```

```

    margin-bottom: 20px;
}
.form-group label {
    font-weight: bold;
    display: block;
    margin-bottom: 5px;
    color: #555;
}
.form-group input[type="text"], .form-group select {
    width: calc(100% - 22px);
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
    transition: border-color 0.3s;
    box-sizing: border-box;
    font-size: 16px;
}
.form-group input[type="text"]:focus, .form-group select:focus {
    border-color: #007bff;
    outline: none;
}
.form-group button {
    background-color: #007bff;
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
    margin-right: 10px;
    font-size: 16px;
}
.form-group button[type="reset"] {
    background-color: #dc3545;
}
.form-group button:hover {
    background-color: #0056b3;
}

```

```

    }
    #address{
        width:12.7cm;
        font-size: 14px;
    }
    @media screen and (max-width: 600px) {
        .form-group input[type="text"], .form-group select {
            width: 100%;
        }
    }
</style>
</head>
<body>

<div class="form-container">
    <h2>College Student Details Form</h2>
    <div class="form-group">
        <label for="name">Name:</label>
        <input type="text" id="name" placeholder="Enter your name">
    </div>
    <div class="form-group">
        <label for="age">Age:</label>
        <input type="text" id="age" placeholder="Enter your age">
    </div>
    <div class="form-group">
        <label for="gender">Gender:</label>
        <select id="gender">
            <option value="male">Select</option>
            <option value="male">Male</option>
            <option value="female">Female</option>
            <option value="other">Other</option>
        </select>
    </div>
    <div class="form-group">
        <label for="department">Department:</label>
        <input type="text" id="department" placeholder="Enter your
            department">
    </div>

```



```

</div>
<div class="form-group">
  <label for="year">Year:</label>
  <input type="text" id="year" placeholder="Enter your year of
    study">
</div>
<div class="form-group">
  <label for="address">Address:</label>
  <textarea id="address" placeholder="Enter your
    address"></textarea>
</div>
<div class="form-group">
  <button type="button" onclick="submitForm()">Submit</button>
  <button type="reset" onclick="reset()">Reset </button>
</div>
</div>
</body>
</html>

```

### **Form.js:-**

```

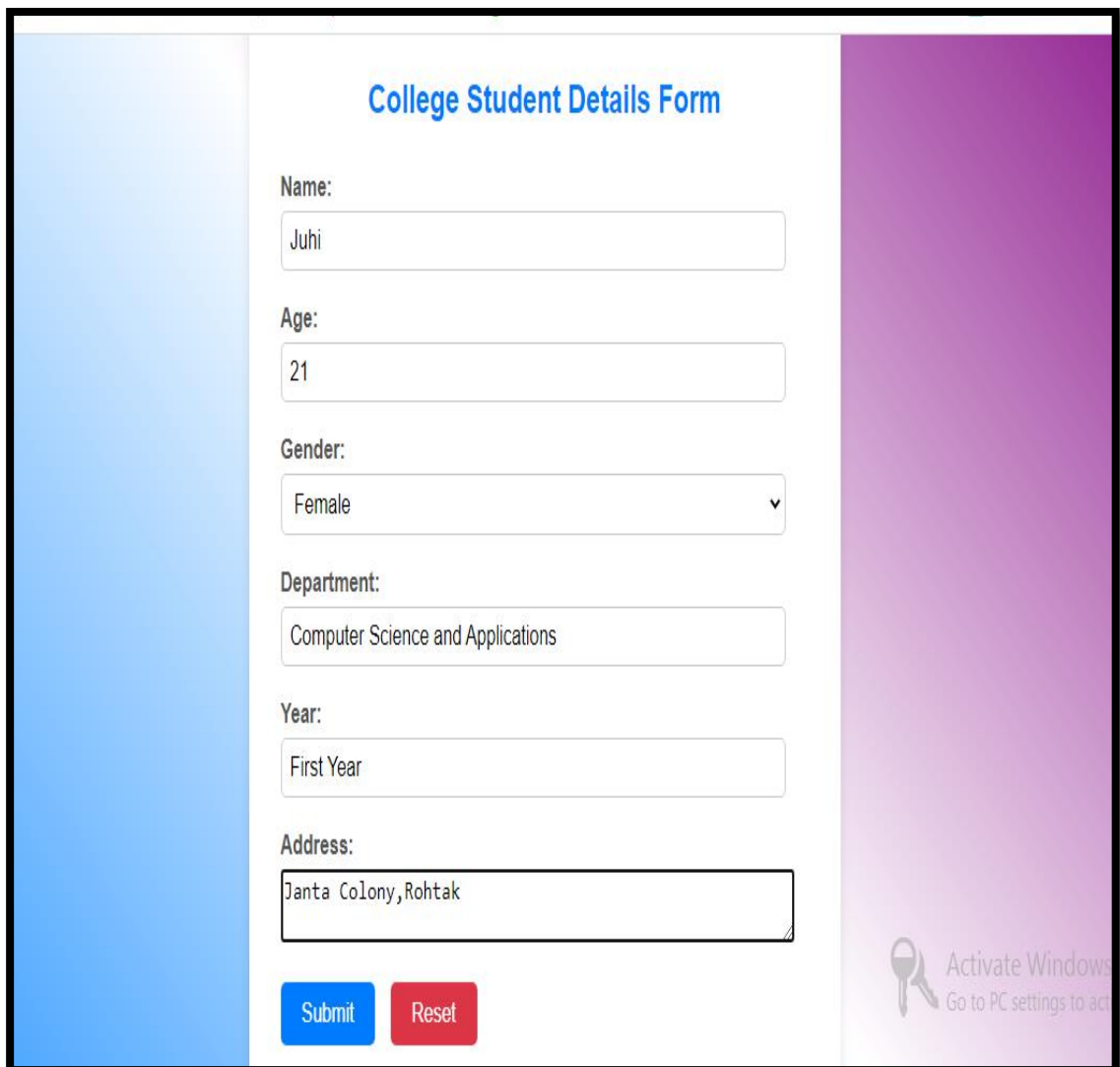
<script>
function submitForm() {
  var name = document.getElementById('name').value;
  var age = document.getElementById('age').value;
  var gender = document.getElementById('gender').value;
  var department= document.getElementById('department').
    value;
  var year = document.getElementById('year').value;
  var address = document.getElementById('address').value;
  alert('Student details submitted:\nName: ' + name + '\nAge: ' +
    age + '\nGender: ' + gender + '\nDepartment: ' +
    department + '\nYear: ' + year + '\nAddress: ' + address);
}

function reset(){
  document.getElementById('name').value=" ";

```

```
document.getElementById('age').value=" ";  
document.getElementById('gender').value=" ";  
document.getElementById('department').value=" ";  
document.getElementById('year').value=" ";  
document.getElementById('address').value=" ";  
}  
</script>
```

## Output:-



The screenshot displays a web form titled "College Student Details Form". The form contains the following fields and values:

- Name: Juhi
- Age: 21
- Gender: Female (dropdown menu)
- Department: Computer Science and Applications
- Year: First Year
- Address: Janta Colony, Rohtak

At the bottom of the form, there are two buttons: "Submit" (blue) and "Reset" (red). The form is set against a background with a blue gradient on the left and a purple gradient on the right. A Windows watermark is visible in the bottom right corner.

### 3. Write a program to implement Event and AWT components.

#### a) Button:-

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

public class ButtonExample extends Frame implements ActionListener {
    private Button button;

    public ButtonExample() {
        // Create a frame
        super("Button Example");

        // Create a button
        button = new Button("Click Me");

        // Set the layout
        setLayout(new FlowLayout());

        // Add button to the frame
        add(button);

        // Register action listener for the button
        button.addActionListener(this);

        // Set frame properties
        setSize(300, 200);
        setVisible(true);

        // Handle window closing event
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
```

```

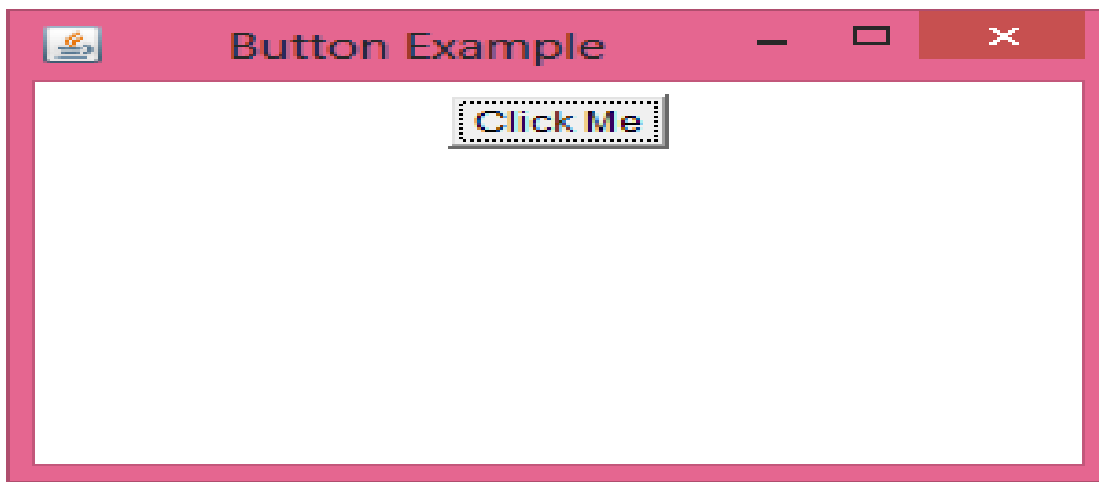
        dispose(); // Release resources
    }
});
}

// Action listener implementation
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button) {
        // Display some text in a dialog box
        Dialog dialog = new Dialog(this, "Button Clicked", true);
        dialog.setLayout(new FlowLayout());
        Label label = new Label("Button Clicked!");
        dialog.add(label);
        dialog.setSize(200, 100);
        dialog.setVisible(true);
    }
}

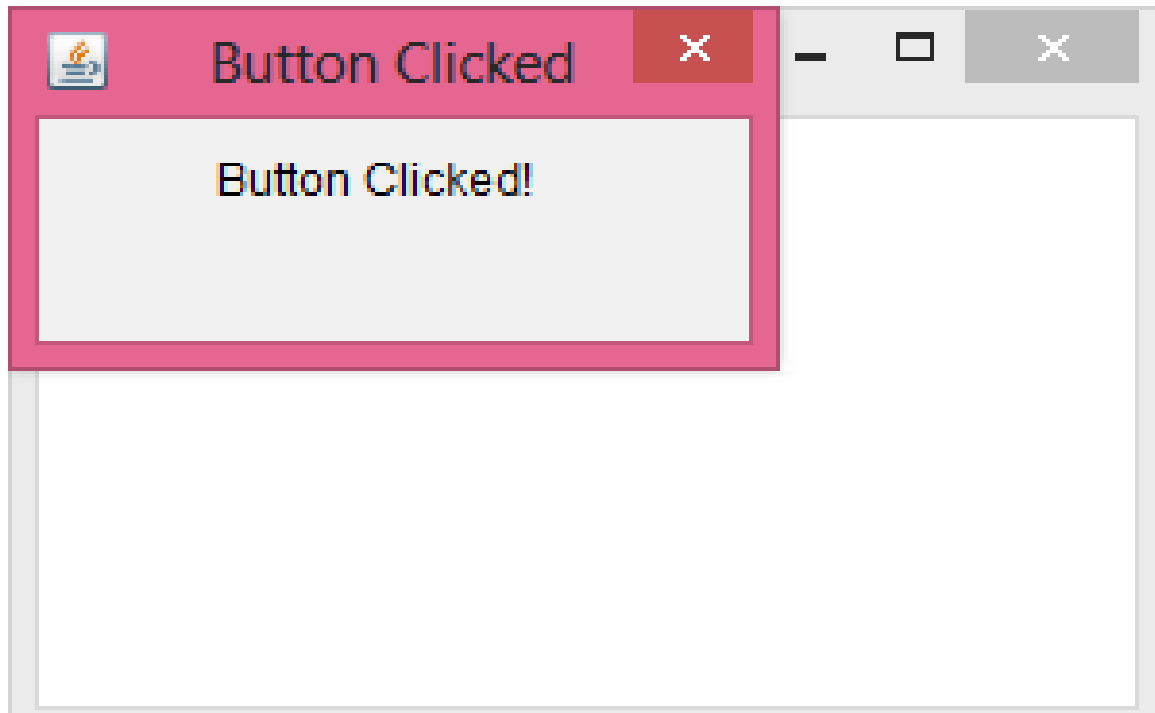
public static void main(String[] args) {
    new ButtonExample(); // Create an instance of ButtonExample
}
}

```

## **Output:-**



## After Button Clicked:-



### **b) Checkbox:-**

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

public class CheckboxExample extends Frame implements ItemListener {
    private Checkbox checkbox;

    public CheckboxExample() {
        // Create a frame
        super("Checkbox Example");

        // Create a checkbox
        checkbox = new Checkbox("Check Me");

        // Set the layout
```

```

        setLayout(new FlowLayout());

        // Add checkbox to the frame
        add(checkbox);


        // Register item listener for the checkbox
        checkbox.addItemListener(this);

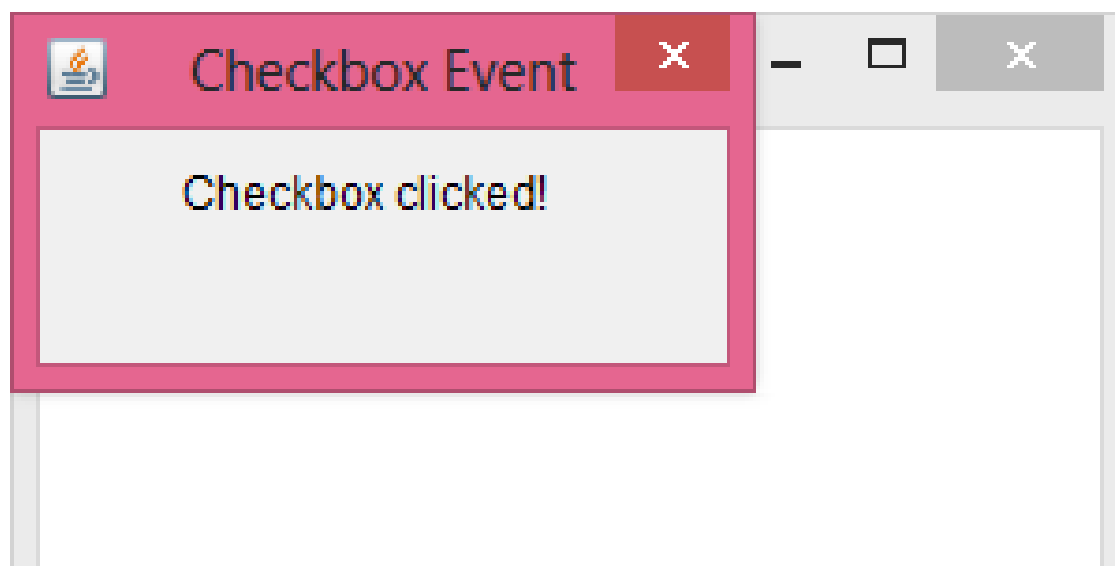
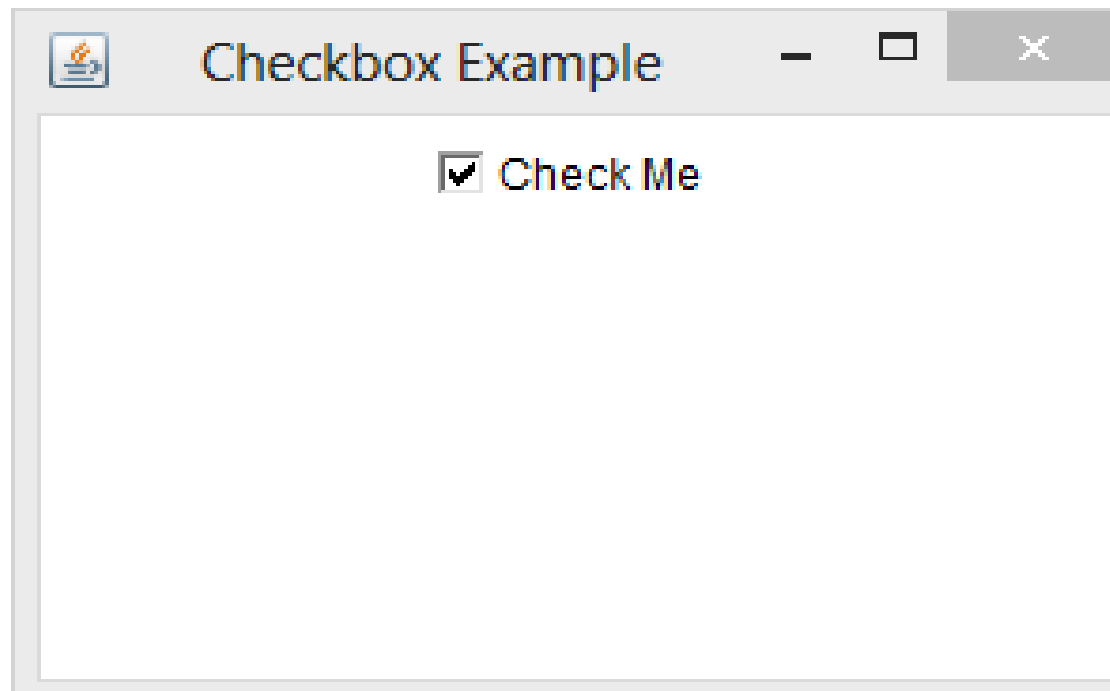

        // Set frame properties
        setSize(300, 200);
        setVisible(true);


        // Handle window closing event
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose(); // Release resources
            }
        });
    }


    // Item listener implementation
    public void itemStateChanged(ItemEvent e) {
        if (e.getSource() == checkbox) {
            if (checkbox.getState()) {
                // Display message in a dialog box
                Dialog dialog = new Dialog(this, "Checkbox Event", true);
                dialog.setLayout(new FlowLayout());
                Label label = new Label("Checkbox clicked!");
                dialog.add(label);
                dialog.setSize(200, 100);
                dialog.setVisible(true);
            }
        }
    }
}

```

## Output:-



#### 4. . Write a program to implement swing components.

##### a) Button:-

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

public class ButtonExample {
    public static void main(String[] args) {
        // Create a JFrame
        JFrame frame = new JFrame("Button Example");

        // Create a JButton
        JButton button = new JButton("Click Me");

        // Add action listener to the button
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null, "Button Clicked!");
            }
        });

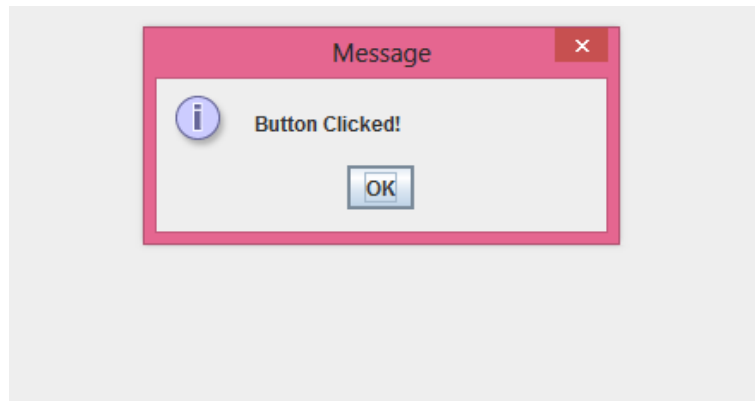
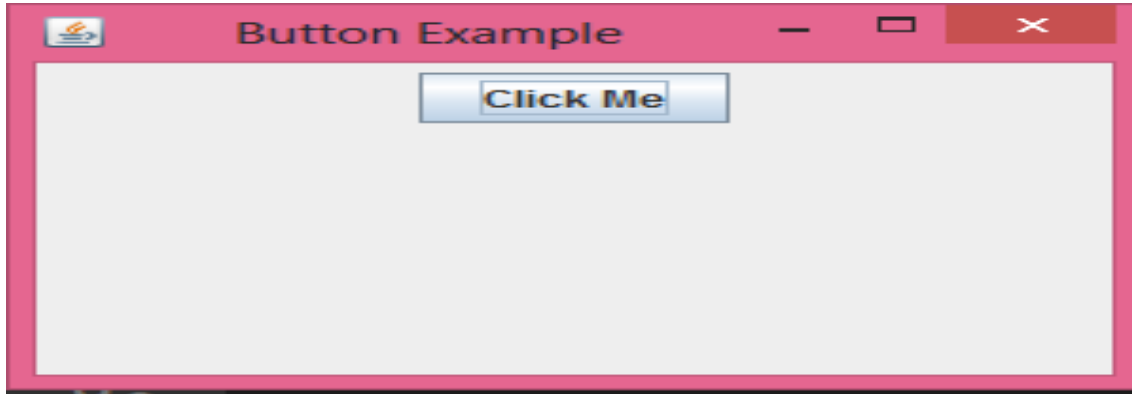
        // Set layout for the frame
        frame.setLayout(new java.awt.FlowLayout());

        // Add button to the frame
        frame.add(button);

        // Set frame properties
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



## Output:-



## b) Table:-

```
import javax.swing.*.*;
import javax.swing.table.DefaultTableModel;

public class TableExample {
    public static void main(String[] args) {
        // Create a JFrame
        JFrame frame = new JFrame("Table Example");
```

```

// Create a table model
DefaultTableModel model = new DefaultTableModel();

// Add columns to the model
model.addColumn("ID");
model.addColumn("Name");
model.addColumn("Age");

// Add rows to the model
model.addRow(new Object[]{"1", "John Doe", "30"});
model.addRow(new Object[]{"2", "Jane Smith", "25"});
model.addRow(new Object[]{"3", "Tom Brown", "35"});

// Create a JTable with the model
JTable table = new JTable(model);

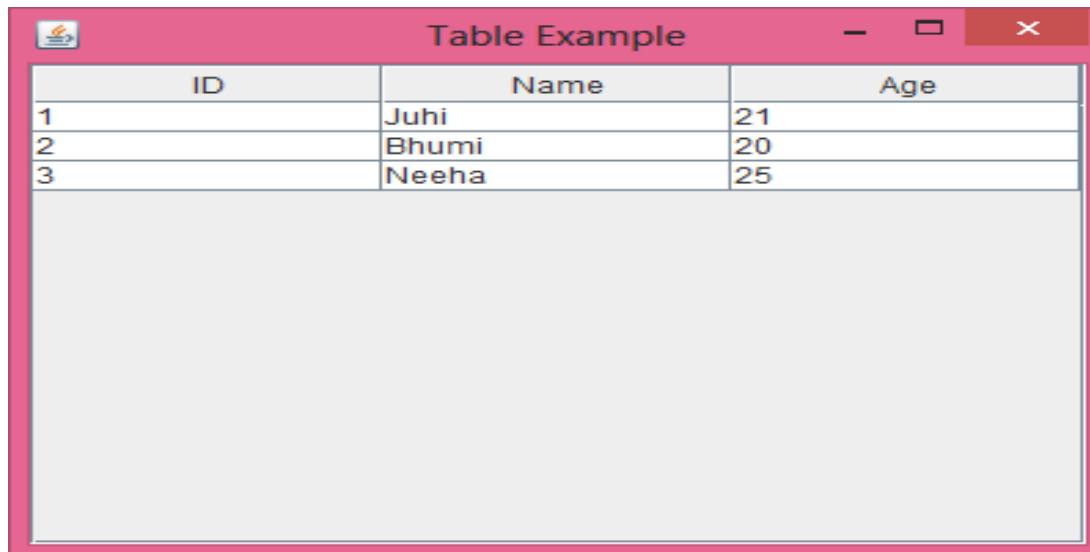
// Create a JScrollPane to hold the table
JScrollPane scrollPane = new JScrollPane(table);

// Add scroll pane to the frame
frame.add(scrollPane);

// Set frame properties
frame.setSize(400, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
}
}

```

## Output:-



ID	Name	Age
1	Juhi	21
2	Bhumi	20
3	Neeha	25

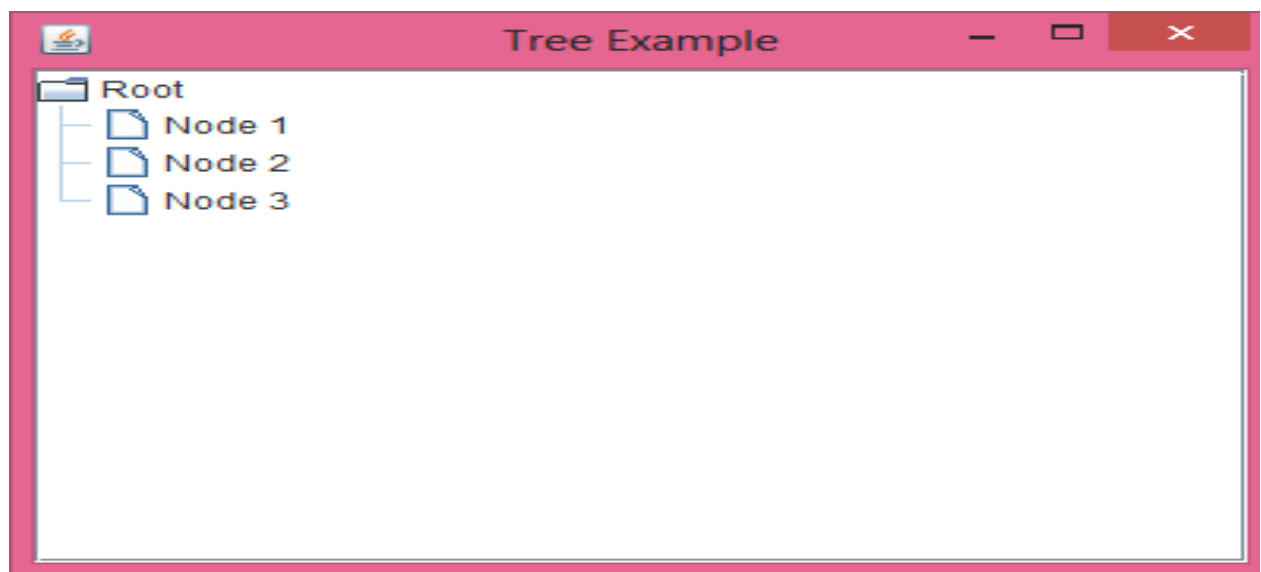
## c) Tree:-

```
import javax.swing.*;
import javax.swing.tree.DefaultMutableTreeNode;
public class TreeExample {
    public static void main(String[] args) {
        // Create a JFrame
        JFrame frame = new JFrame("Tree Example");
        // Create tree nodes
        DefaultMutableTreeNode rootNode = new
        DefaultMutableTreeNode("Root")
        DefaultMutableTreeNode node1 = new
        DefaultMutableTreeNode("Node 1");
        DefaultMutableTreeNode node2 = new
        DefaultMutableTreeNode("Node 2");
        DefaultMutableTreeNode node3 = new
        DefaultMutableTreeNode("Node 3");
```

```
// Add child nodes to the root node
rootNode.add(node1);
rootNode.add(node2);
rootNode.add(node3);

// Create a JTree with the root node
JTree tree = new JTree(rootNode);
// Create a JScrollPane to hold the tree
JScrollPane scrollPane = new JScrollPane(tree);
// Add scroll pane to the frame
frame.add(scrollPane);
// Set frame properties
frame.setSize(400, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
}
}
```

## **Output:-**



## **d) CheckedPane:-**

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.util.ArrayList;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;

public class CheckboxPane extends JFrame {
    private ArrayList<JCheckBox> boxes = new ArrayList<JCheckBox>();
    JSplitPanesplitPane;
    private JPanelleftPanel;
    private JPanelrightPanel;
    public CheckboxPane() {
        leftPanel = new JPanel(new GridBagLayout());
        rightPanel = new JPanel(new GridBagLayout()) {
            @Override
            public Dimension getPreferredSize() {
                return new Dimension(200, 200);
            }
        };

        splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, leftPanel, new
        JScrollPane(rightPanel));
        leftPanel.setBackground(Color.BLUE);
        add(splitPane);
        addBoxes();
    }
}
```

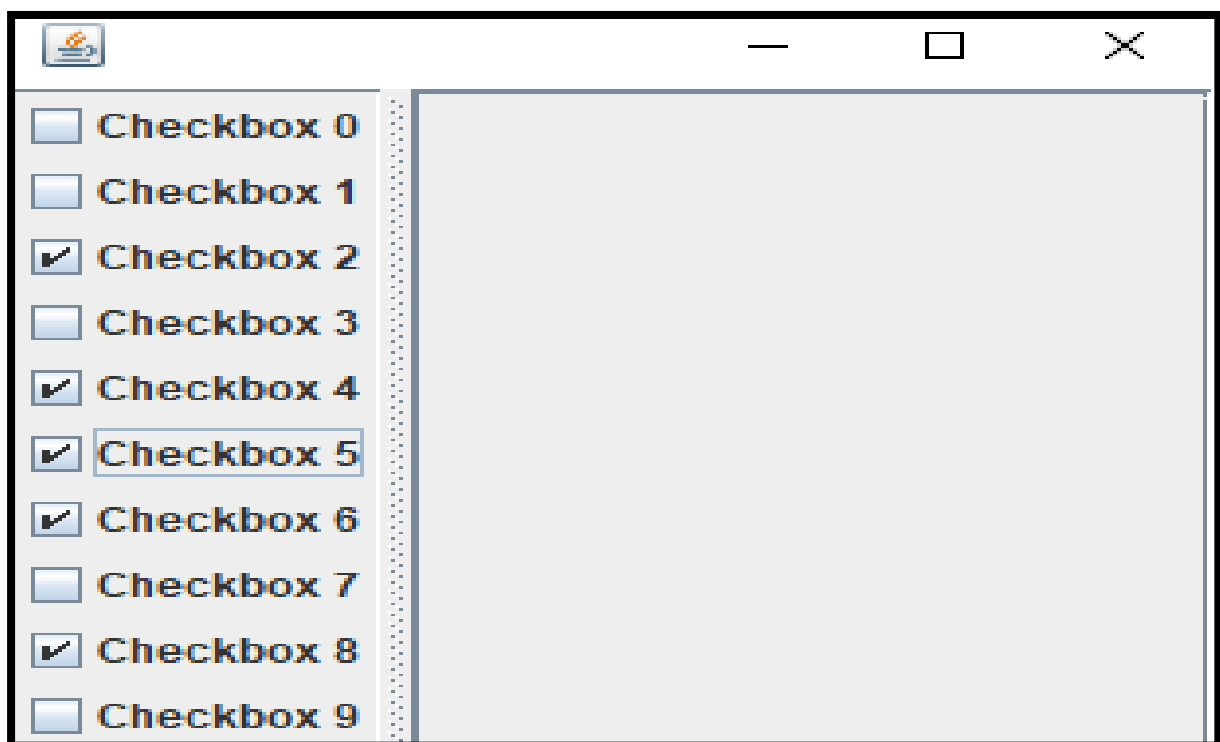
```

void addBoxes() {
    int i = 0;
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.gridwidth = GridBagConstraints.REMAINDER;
    for (i = 0; i < 10; i++) {
        leftPanel.add(new JCheckBox("Checkbox " + i), gbc);
    }
}

public static void main(String[] args) {
    CheckboxPane cb = new CheckboxPane();
    cb.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    cb.pack();
    cb.setLocationRelativeTo(null);
    cb.setVisible(true);
}
}

```

## Output:-



## 5. Write a program to implement Swing Components.

### a) TabbedPane:-

```
import javax.swing.*;

public class TabbedPaneExample {
    public static void main(String[] args) {
        // Create a JFrame
        JFrame frame = new JFrame("Tabbed Pane Example");

        // Create a JTabbedPane
        JTabbedPane tabbedPane = new JTabbedPane();

        // Create tabs
        JPanel tab1 = new JPanel();
        JLabel label1 = new JLabel("Content of Tab 1");
        tab1.add(label1);

        JPanel tab2 = new JPanel();
        JLabel label2 = new JLabel("Content of Tab 2");
        tab2.add(label2);

        JPanel tab3 = new JPanel();
        JLabel label3 = new JLabel("Content of Tab 3");
        tab3.add(label3);

        // Add tabs to the tabbed pane
        tabbedPane.addTab("Tab 1", tab1);
        tabbedPane.addTab("Tab 2", tab2);
        tabbedPane.addTab("Tab 3", tab3);

        // Add tabbed pane to the frame
        frame.add(tabbedPane);
    }
}
```

```

        // Set frame properties
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

    public static boolean matchSubstring(String mainString, String
                                        substring) {

        int mainLength = mainString.length();
        int subLength = substring.length();

        // Iterate through the main string
        for (int i = 0; i <= mainLength - subLength; i++) {
            int j;
            /*Check if the substring matches the current portion of the main
            string*/

            for (j = 0; j < subLength; j++) {
                if (mainString.charAt(i + j) != substring.charAt(j)) {
                    break;
                }
            }
            /* If the inner loop completed, it means the substring matches*/
            if (j == subLength) {
                return true;
            }
        }

        // If no match is found, return false
        return false;
    }

    public static void main(String[] args) {
        String mainString = "Hello World";
        String substring = "World";
    }
}

```

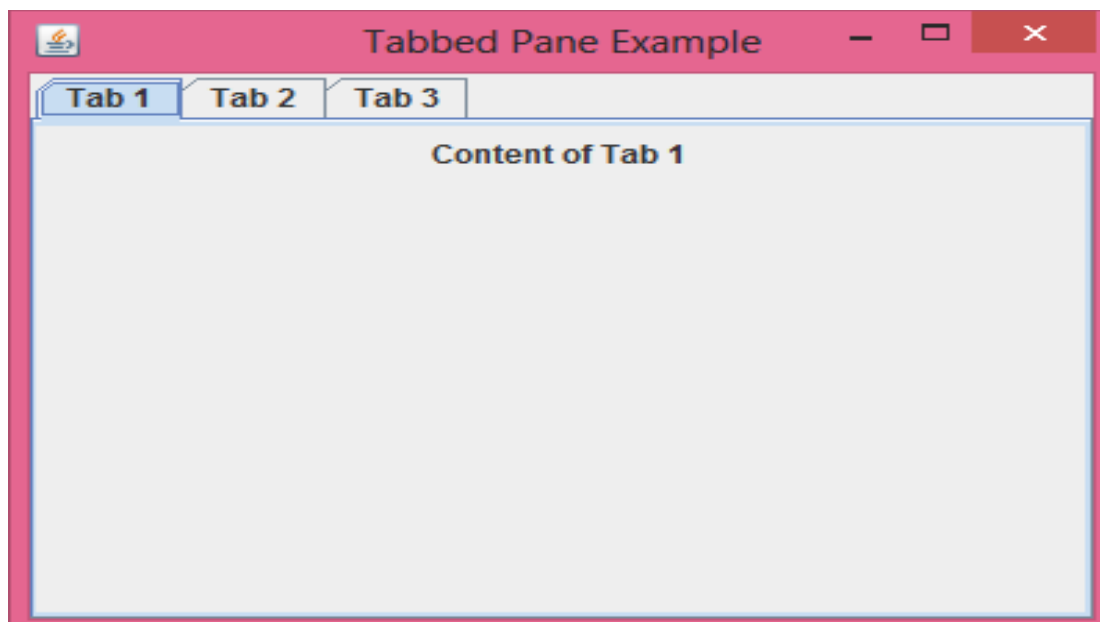


```

// Check if the substring matches the main string
if (matchSubstring(mainString, substring))
{
    System.out.println("Main String is: "+mainString);
    System.out.println("Sustring is : "+substring);
    System.out.println("Substring found in Main String!!!");
}
else
{
    System.out.println("Main String is "+ mainString);
    System.out.println("Sustring is :"+ substring);
    System.out.println("Substring not found in Main String!!!");
}
}
}
}

```

## **Output:-**



## **b) ScrollPane:-**

```
import javax.swing.*.*;
```

```

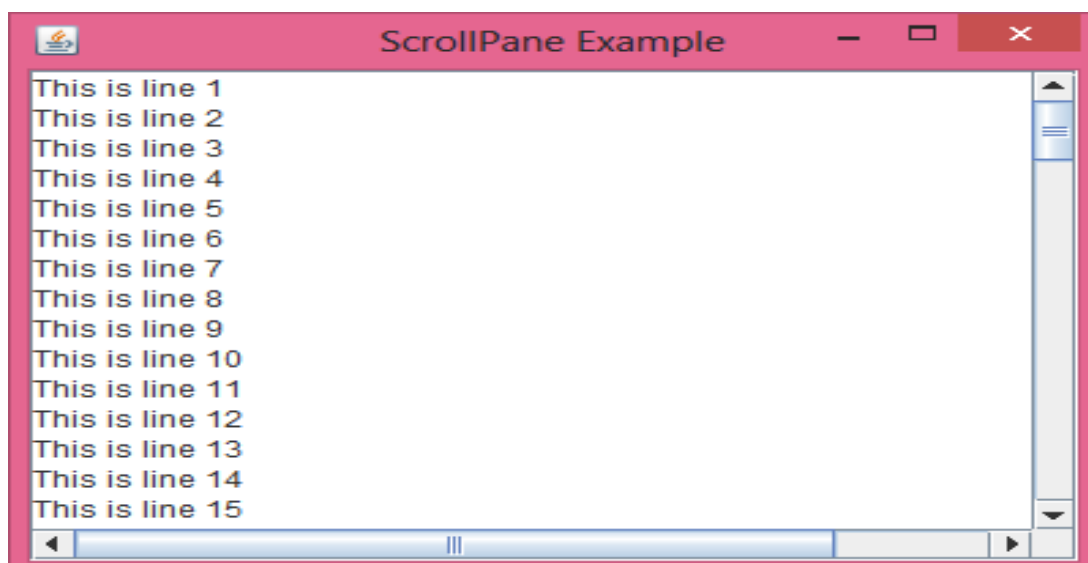
public class ScrollPaneExample {
    public static void main(String[] args) {
        // Create a JFrame
        JFrame frame = new JFrame("ScrollPane Example");
        // Create a JTextArea with a long text
        JTextArea textArea = new JTextArea(20, 40);
        for (int i = 0; i < 100; i++) {
            textArea.append("This is line " + (i + 1) + "\n");
        }

        // Create a JScrollPane with the text area
        JScrollPane scrollPane = new JScrollPane(textArea);

        // Add scroll pane to the frame
        frame.add(scrollPane);
        // Set frame properties
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

```

## Output:-



## 6. Write a program in Java to implement all the phases of the life cycle of servlet.

### Servlet Life Cycle Methods:-

There are three life cycle methods of a Servlet :-

- **init() method:** The **Servlet.init()** method is called by the Servlet container to indicate that this Servlet instance is instantiated successfully and is about to put into service.
- **service() method:** The **service()** method of the Servlet is invoked to inform the Servlet about the client requests.
  - This method uses **ServletRequest** object to collect the data requested by the client.
  - This method uses **ServletResponse** object to generate the output content.
- **destroy() method:** The **destroy()** method runs only once during the lifetime of a Servlet and signals the end of the Servlet instance.

```
import javax.servlet.*;  
import javax.servlet.http.*;  
import java.io.*;
```

```
public class ServletLifeCycleExample extends HttpServlet {  
  
    // Initialization phase  
    public void init() throws ServletException {  
        System.out.println("Initialization Phase: Initializing Servlet...");  
        // Perform initialization tasks here  
    }  
}
```

```

// Service phase
protected void service(HttpServletRequest request,
                        HttpServletResponse response)
    throws ServletException, IOException {
    System.out.println("Service Phase: Servicing Request...");
    // Perform request handling tasks here
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html><body>");
    out.println("<h1>Hello,Servlet!</h1>");
    out.println("</body></html>");
}

// Destruction phase
public void destroy() {
    System.out.println("Destruction Phase: Destroying Servlet...");
    // Perform destruction tasks here
}
}

```

## Output:-

```

Initialization Phase: Initializing Servlet...
Service Phase: Servicing Request...
Destruction Phase: Destroying Servlet...

```



## **7. Write a program to show implement DHTML and CSS with JAVASCRIPT.**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
                                scale=1.0">
  <title>Responsive Webpage with Dynamic Element</title>
  <style>
    /* CSS for styling */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f0f0f0;
    }

    .header {
      background-color: #333;
      color: white;
      padding: 10px;
      text-align: center;
    }

    .container {
      margin: 20px auto;
      padding: 20px;
      background-color: white;
      border-radius: 5px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      text-align: center;
    }

    .dynamic-element {
      width: 200px;
```

```

        height: 200px;
        background-color: red;
        margin: 20px auto;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }
</style>
</head>
<body>
    <div class="header">
        <h1>Responsive Webpage with Dynamic Element</h1>
    </div>
    <div class="container">
        <h2>Welcome to our Responsive Webpage!</h2>
        <p>This webpage adjusts its layout and content based on the size of
            the screen or viewport, providing a better user experience on
            different devices such as desktops, tablets, and smartphones
        .</p>
        <div class="dynamic-element" onclick="changeColor()"></div>
    </div>

    <script>
        // JavaScript for responsiveness and dynamic behavior
        function changeColor() {
            var element = document.querySelector('.dynamic-element');
            var randomColor = '#' + Math.floor(Math.random()*16777215).
                toString(16);
            // Generate a random color
            element.style.backgroundColor = randomColor;
        }
    </script>
</body>
</html>

```

## Output:-

### Responsive Webpage with Dynamic Element

#### Welcome to our Responsive Webpage!

This webpage adjusts its layout and content based on the size of the screen or viewport, providing a better user experience on different devices such as desktops, tablets, and smartphones.



### Responsive Webpage with Dynamic Element

#### Welcome to our Responsive Webpage!

This webpage adjusts its layout and content based on the size of the screen or viewport, providing a better user experience on different devices such as desktops, tablets, and smartphones.



## 8. How is role of server side is different from client-side in a typical website. Clear using an example.

### Server-side:-

Server-side code runs on the web server and is responsible for handling requests, processing data, interacting with databases, and generating dynamic content to be sent to the client's browser.

### Example using JSP (JavaServer Pages):

We have a simple web application where users can enter their name, and the server will generate a personalized greeting message.

```
<!-- index.jsp -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
  <title>Greeting Form (Server-side)</title>
</head>
<body>
  <h1>Welcome to our Website!</h1>
  <form action="greet.jsp" method="post">
    Enter your name: <input type="text" name="name">
    <input type="submit" value="Submit">
  </form>
</body>
</html>

<!-- greet.jsp -->
```

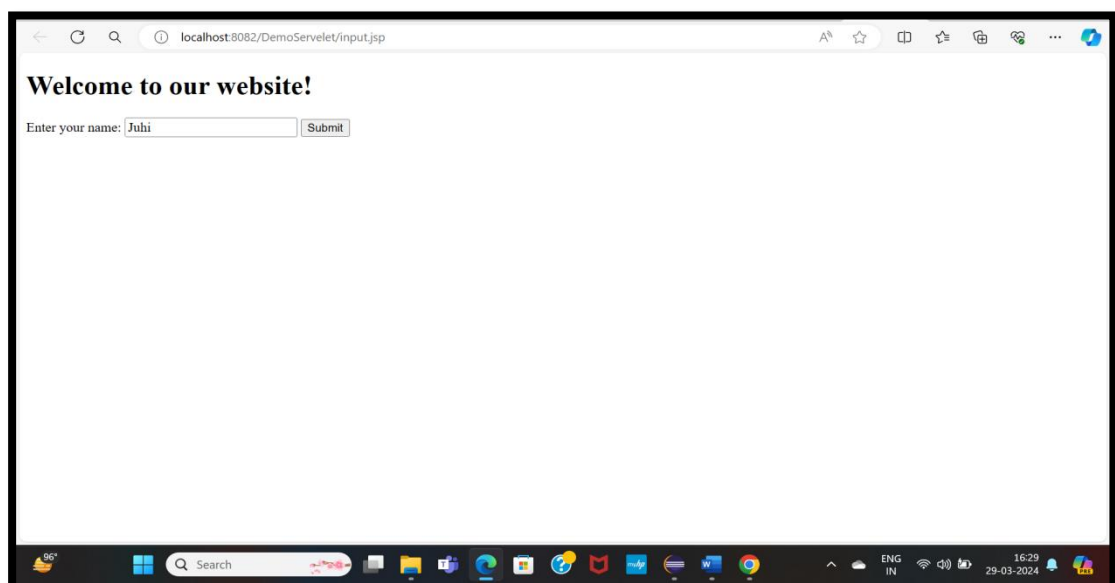


```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">

<title>Greeting Message</title>
</head>
<body>
<%
String name = request.getParameter("name");
String greeting = name != null ? "Hello, " + name + "! Welcome to
our website." : "Welcome!";
%>
<h1><%= greeting %></h1>
</body>
</html>
}
}
```

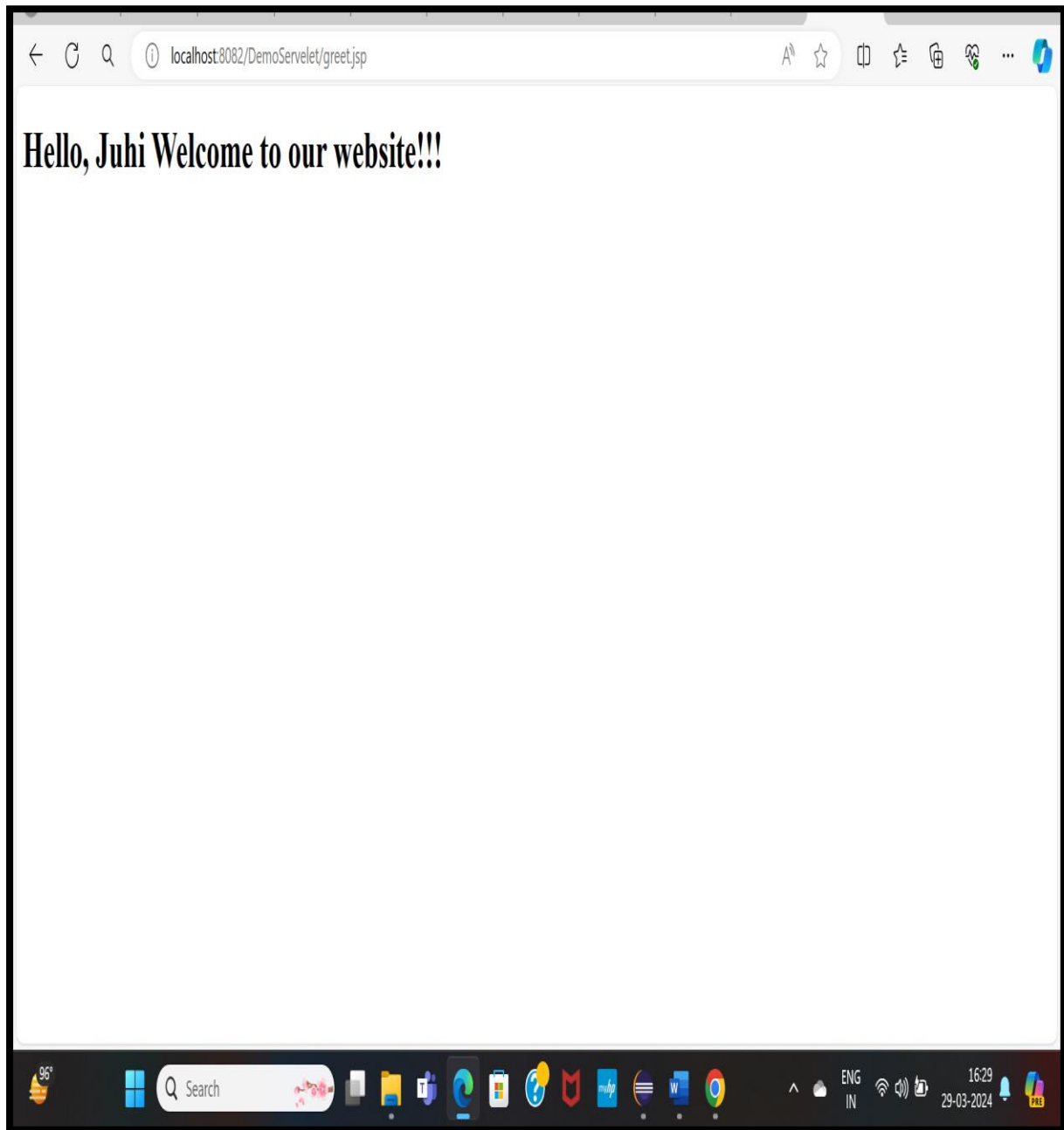
## Output:-

(1)



## Output:-

(2)



## **Client-side:-**

Client-side code runs in the user's web browser and is responsible for rendering the user interface, handling user interactions, and enhancing the user experience without needing to communicate with the server.

## **Example using JavaScript:**

We have the same functionality as above, but we want to implement it using client-side JavaScript for interactivity.

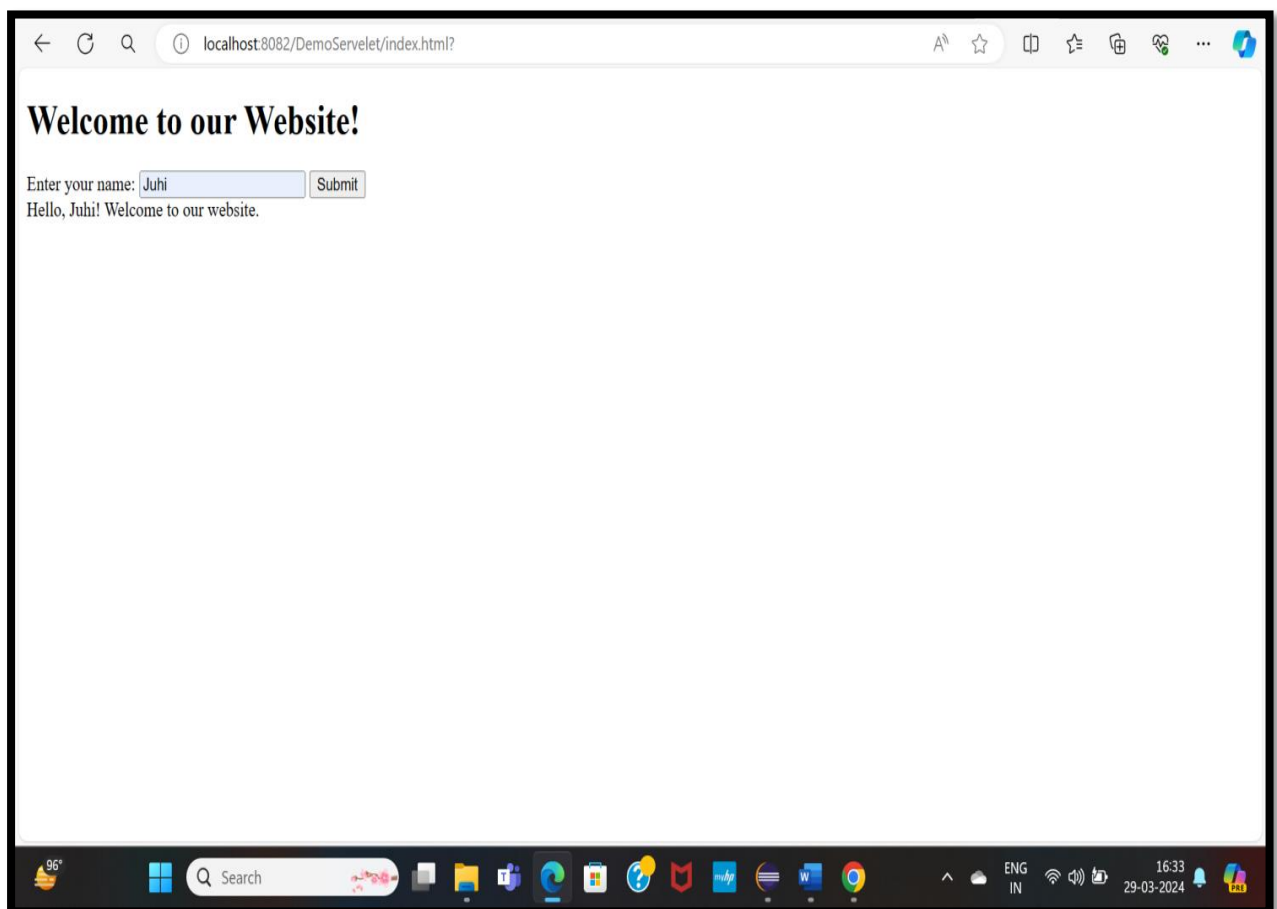
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
                                scale=1.0">
  <title>Greeting Form (Client-side)</title>
</head>
<body>
  <h1>Welcome to our Website!</h1>
  <form id="greetingForm">
    Enter your name: <input type="text" id="name">
    <input type="submit" value="Submit">
  </form>
  <div id="greetingMessage"></div>
</body>
</html>
```

## **Javascript Code:-**

```
<script>
document.getElementById('greetingForm').addEventListener('submit',
                                                    function(event)
{
    event.preventDefault();
    var name = document.getElementById('name').value;
    var greetingMessage = name ? 'Hello, ' + name + '! Welcome to
                                our website.' : 'Welcome!';
    document.getElementById('greetingMessage').innerText =
                                                greetingMessage;

});
</script>
```

## Output:-



## **9. Write a program in Java using JSP which accept two integer numbers from user and display the result.**

### **Index.html:-**

```
<html>
  <head>
    <title>Entertwonumberstoaddup</title>
  </head>
  <body>
    <formaction="./add.jsp">
      First number:
      <input type="text" name="t1"/>
      Secondnumber:
      <input type="text" name="t2"/>
      <input type="submit" value="SUBMIT"/>
    </form>
  </body>
</html>
```

### **Index.jsp:-**

```
<html>
  <head>
    <title>Entertwonumberstoaddup</title>
  </head>
  <body>
    <%= "<h1> The sum is" + (Integer.parseInt(request.get
      Parameter("t1")) + Integer.parseInt(request.get
      Parameter("t2"))) + "</h1>" %>
  </body>
</html>
```

## Output:-



## 10. Write a program using POST and GET Method in swing.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;

public class Main extends JFrame {
    private JTextField textField;
    private JTextArea textArea;

    public Main() {
        setTitle("POST/GET Example");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Text field for user input
        textField = new JTextField();
        add(textField, BorderLayout.NORTH);

        // Button for POST method
        JButton postButton = new JButton("Send POST");
        postButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                sendData("POST");
            }
        });

        // Button for GET method
        JButton getButton = new JButton("Send GET");
        getButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                sendData("GET");
            }
        });
    }

    private void sendData(String method) {
        // Implementation of sendData method
    }
}
```

```

    }
    });

    // Panel for buttons
    JPanel buttonPanel = new JPanel();
    buttonPanel.add(postButton);
    buttonPanel.add(getButton);
    add(buttonPanel, BorderLayout.CENTER);

    // Text area to display server response
    textArea = new JTextArea();
    add(new JScrollPane(textArea), BorderLayout.SOUTH);

    setSize(400, 300);
    setVisible(true);
}

private void sendData(String method) {
    String urlString = "http://jsonplaceholder.typicode.com/posts";
    // Example URL
    String data = textField.getText();

    try {
        URL url = new URL(urlString);
        HttpURLConnection connection = (HttpURLConnection)
            url.openConnection();
        connection.setRequestMethod(method);

        if (method.equals("POST")) {
            connection.setDoOutput(true);
            OutputStreamWriter writer = new
                OutputStreamWriter(connection.getOutputStream());
            writer.write(data);
            writer.flush();
            writer.close();
        }
    }
}

```



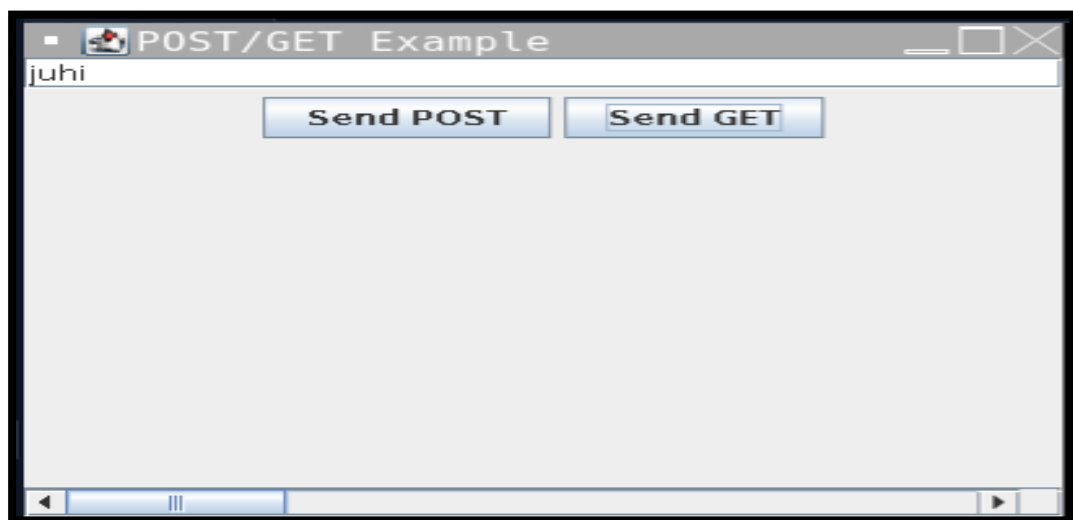
```

BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
    StringBuilder response = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
        response.append(line);
        response.append("\n");
    }
    reader.close();
    textArea.setText(response.toString());
} catch (IOException ex) {
    ex.printStackTrace();
    textArea.setText("Error: " + ex.getMessage());
}
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new Main();
        }
    });
}
}

```

## Output:-



## POST Method:

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "Post 1 Title",
    "body": "juhi is great."
  },
  {
    "userId": 1,
    "id": 2,
    "title": "Post 2 Title",
    "body": "juhi is awesome."
  },
  // More posts...
]
```

## GET Method:

```
{
  "userId": 1,
  "id": 101, // This will be automatically generated by the server
  "title": "New Post Title",
  "body": "juhi"
}
```

## **11. Write a Javascript program to check number entered is an Armstrong number or not.**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
        scale=1.0">
<title>Armstrong Number Checker</title>

<style>
    body {
        font-family: Arial, sans-serif;
        text-align: center;
    }
    h1 {
        margin-top: 50px;
    }
    #result {
        margin-top: 20px;
    }
</style>

</head>
<body>
<h1>Armstrong Number Checker</h1>
<div>
    <label for="numberInput">Enter a number:</label>
    <input type="number" id="numberInput" min="0">
```

```
<button onclick="checkArmstrongNumber()">Check</button>
</div>
<div id="result"></div>
```

```
<script>
function checkArmstrongNumber() {
const numberInput =
document.getElementById("numberInput").value;
const number = parseInt(numberInput);

    if (isNaN(number)) {
        document.getElementById("result").textContent =
        "Please enter a valid number.";
        return;
    }

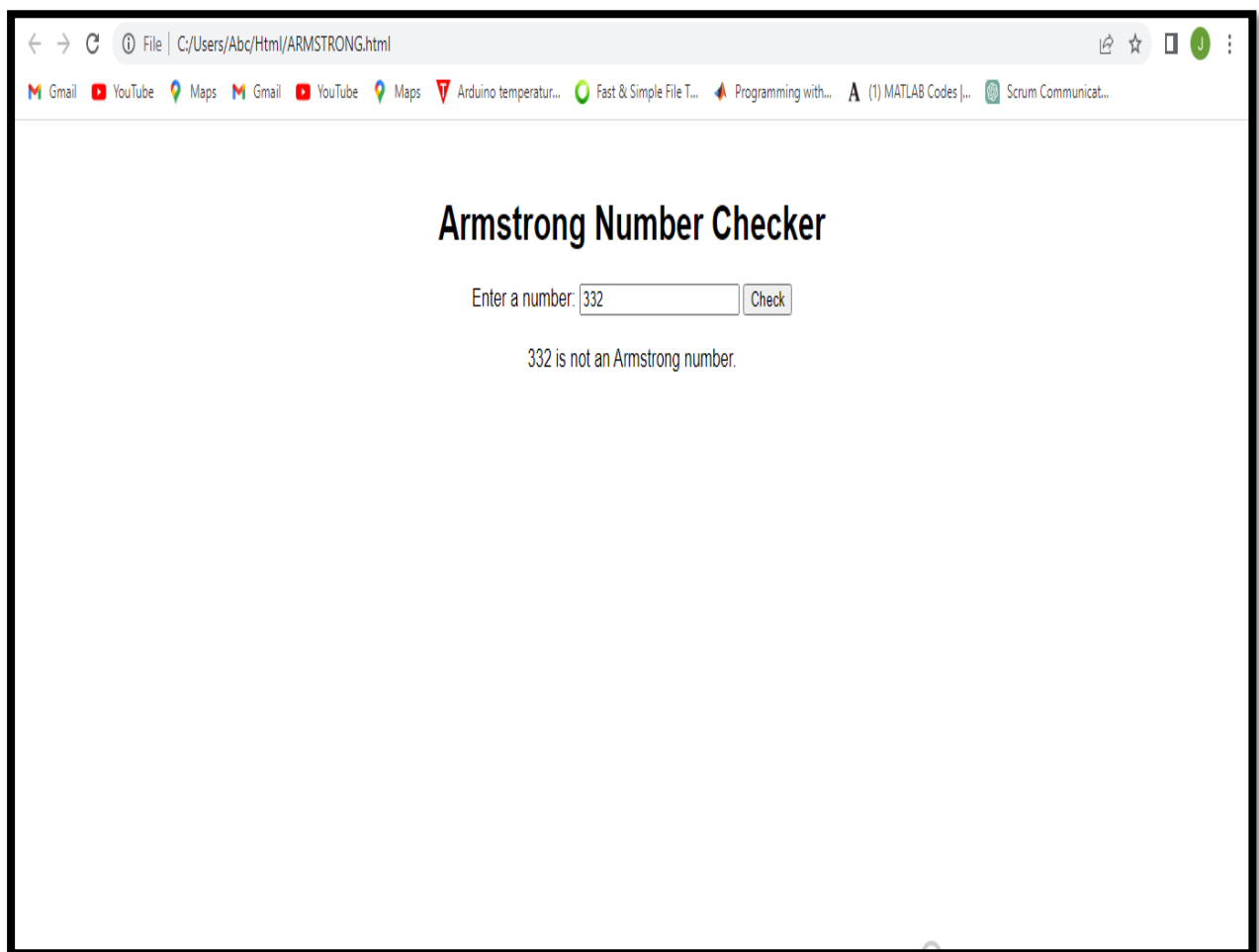
    const numString = number.toString();
    const numDigits = numString.length;
    let sum = 0;

    for (let digit of numString) {
        sum += Math.pow(parseInt(digit), numDigits);
    }

    if (sum === number)
    {
        document.getElementById("result").textContent =
        number + " is an Armstrong number.";
    }
}
```

```
else {  
    document.getElementById("result").textContent =  
        number + " is not an Armstrong number.";  
}  
}  
</script>  
</body>  
</html>
```

## Output:-



## **12. Write a Javascript program to create a Login Form and validate it.**

### **Index.html:-**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
      scale=1.0">
<title>Login Form</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>

<div class="login-container">
  <h2>Login</h2>
  <form id="loginForm">
    <input type="email" id="email" placeholder="Email"
          required><br>
    <input type="password" id="password"
          placeholder="Password" required><br>
    <input type="submit" value="Login">
    <p id="errorMessage" class="error"></p>
  </form>
  <div class="links">
    <a href="#">Forgot password?</a>
    <span>|</span>
    <a href="#">Create account</a>
  </div>
</div>
```

### **styles.css:-**

```
body {
  font-family: Arial, sans-serif;
  background-color: #f1f1f1;
```

```

    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
.login-container {
    background-color: #fff;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
    text-align: center;
    max-width: 400px;
    width: 100%;
}
h2 {
    margin-bottom: 20px;
    color: #333;
}
input[type="email"], input[type="password"] {
    width: calc(100% - 22px);
    padding: 10px;
    margin-bottom: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box;
}
input[type="submit"] {
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    width: 100%;
}

```

```

        font-size: 16px;
    }
    input[type="submit"]:hover {
        background-color: #45a049;
    }
    .links {
        margin-top: 20px;
    }
    .links a {
        color: #333;
        text-decoration: none;
        margin: 0 10px;
    }
    .links a:hover {
        text-decoration: underline;
    }
    .error {
        color: red;
    }
}

```

### **script.js:-**

```

document.getElementById("loginForm").addEventListener
    ("submit", function(event) {
        event.preventDefault();
        var email = document.getElementById("email").value;
        var password = document.getElementById("password").value;
        var errorMessage = document.getElementById("errorMessage");

        // Simple validation for email and password
        if (email.trim() === "" || password.trim() === "") {
            errorMessage.textContent =
                "Please enter both email and password.";
            return false;
        }

        // Simulate authentication - replace with your own

```

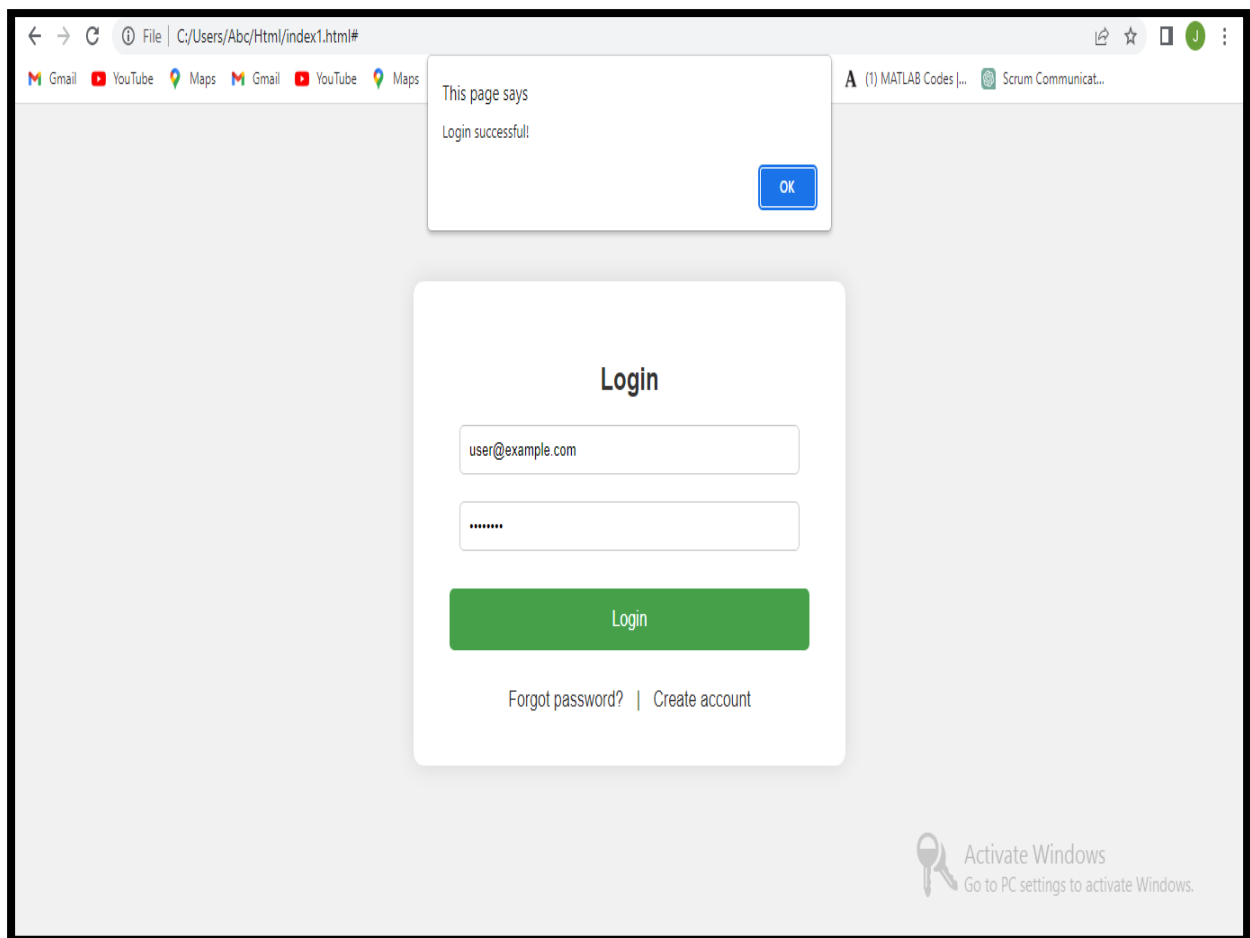


```
if (email !== "user@example.com" || password !== "password")
{
    errorMessage.textContent = "Incorrect email or password.";
    return false;
}

// Clear error message if no validation issues
errorMessage.textContent = "";

// Successful login, you can redirect user to another page here
alert("Login successful!");
});
```

## Output:-



### 13. Write a program to implement Event and AWT compnents.

a) CANVAS:-

b) SCROLLBAR:-

```
import java.awt.*;
//class to construct a frame and containing main method
public class CanvasExample
{
    // class constructor

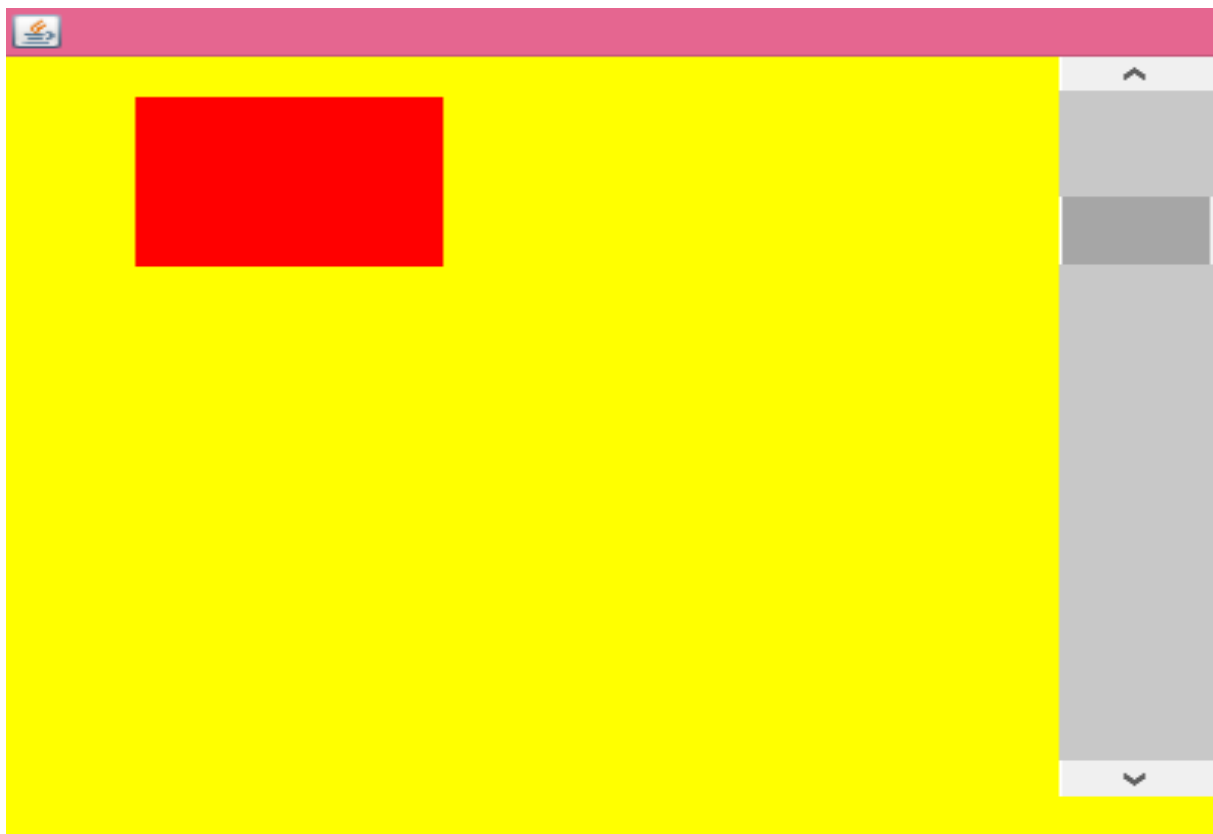
    public CanvasExample()
    {
        Frame f =new Frame("CanvasExample");

        //scrollbar
        Scrollbar s = new Scrollbar();
        s.setBounds(350,30,50,350);
        f.add(s);
        f.add(new MyCanvas());
        f.setLayout(null);
        f.setSize(400,400);
        f.setVisible(true);
    }

    public static void main(String args[])
    {
        new CanvasExample();
    }
}
//canvas
```

```
class MyCanvas extends Canvas
{
//classconstructor
public MyCanvas(){
setBackground(Color.YELLOW);
setSize(400,400);
}
public void paint(Graphics g)
{
g.setColor(Color.red);
g.fillRect(50,50,100,80);
}
}
}
```

## **Output:-**



## **14. Write a program using JSP to implement the scripting elements.**

```
%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Scripting Elements Example</title>
</head>
<body>
    <h1>Scripting Elements Example:-</h1>

    <%-- Declaration: Declares a variable --%>
    <%!
        int number = 10;
    %>

    <%-- Scriptlet: Executes Java code --%>
    <%
        String message = "Hello, World!";
        int result = number * 2;
    %>

    <%-- Expression: Evaluates and prints the result --%>
    <p>Message: <%= message %></p>
    <p>Result: <%= result %></p>

    <%-- Expression with Scriptlet: Mixes Java code and HTML --%>
    <% if (number > 5) { %>
        <p>Number is greater than 5</p>
    <% } else { %>
        <p>Number is not greater than 5</p>
    <% } %>
```

```

<%-- Directives: Provide instructions to the JSP container --%>
<%@ page import="java.util.Date" %>
<%
    Date currentDate = new Date();
%>
<p>Current Date: <%= currentDate %></p>
</body>
</html>
}
}

```

### **Explanation:**

**Declaration:** The '`<%! ... %>`' tags are used for declaring variables or methods that will be accessible throughout the JSP page. In this example, we declare an integer variable 'number'.

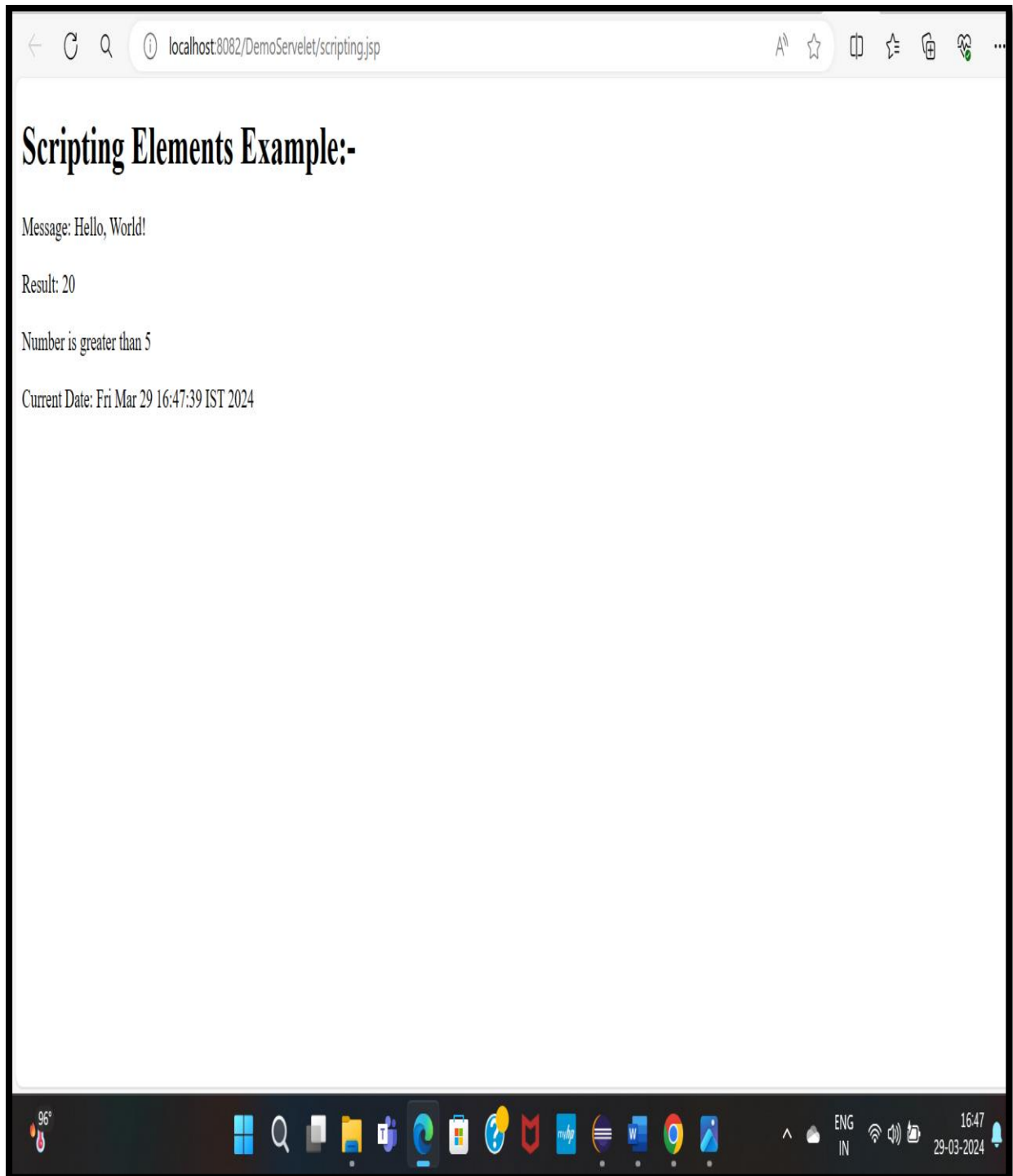
**Scriptlet:** The '`<% ... %>`' tags are used for inserting Java code directly into the JSP page. Here, we initialize a String variable 'message' and perform a calculation to assign a value to the 'result' variable.

**Expression:** The '`<%= ... %>`' tags are used to evaluate an expression and print its result directly into the HTML output. Here, we print the values of 'message' and 'result'.

**Expression with Scriptlet:** We mix Java code and HTML by using scriptlets within expressions. Here, we use an 'if-else' condition to determine whether the 'number' variable is greater than 5 and display a message accordingly.

**Directives:** The '`<%@ ... %>`' directive is used to provide instructions to the JSP container. In this example, we import the 'Date' class from 'java.util' package to display the current date.

# Output:-



## 15. Write a program using JSP to implement any five

### Implicit objects.

```
<%@page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
// 1. request object
    String requestURI = request.getRequestURI();
%>
<p>Request URI: <%=requestURI%></p>

<%
// 2. response object
    response.setContentType("text/html");
%>
<p>Response content type set to HTML.</p>

<%
// 3. out object
    out.println("<p>Hello from out implicit object!</p>");
%>

<%
// 4. session object
    session.setAttribute("username", "John");
    String username = (String)session.getAttribute("username");
%>
<p>Welcome, <%= username %>!</p>

<%
// 5. application object
    application.setAttribute("appName", "MyApp");
    String appName = (String)application.getAttribute("appName");
%>
<p>Application Name: <%=appName%></p>
```

```

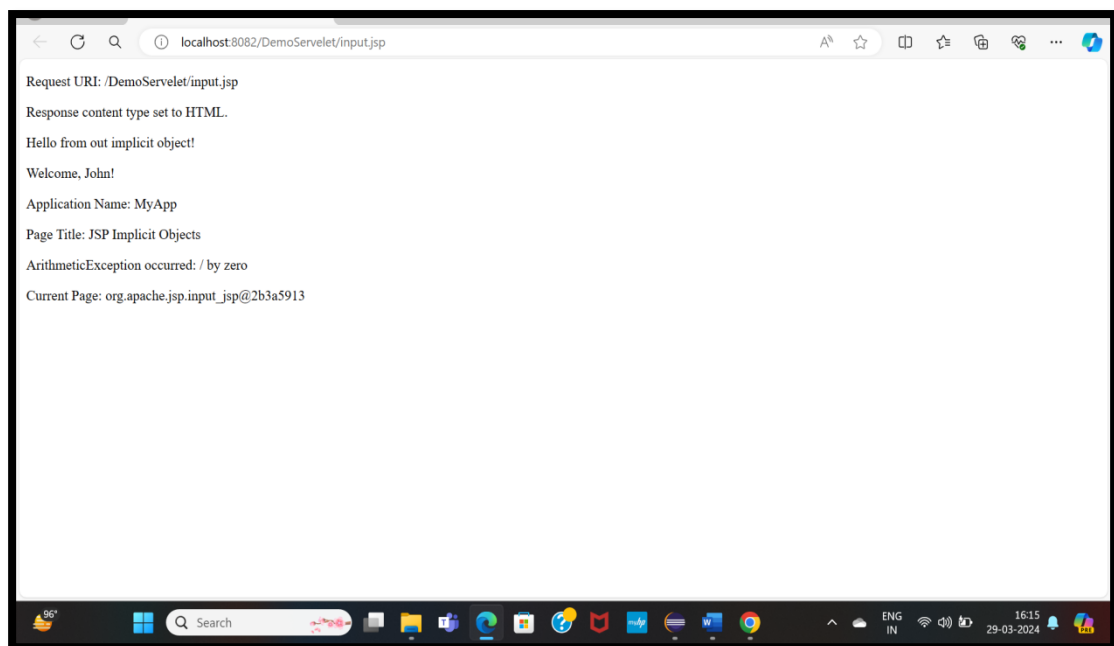
<%
// 6. pageContext object
pageContext.setAttribute("pageTitle", "JSP Implicit Objects",
PageContext.PAGE_SCOPE);
    String pageTitle = (String)pageContext.getAttribute("pageTitle");
%>
<p>Page Title: <%=pageTitle%></p>

<%
// 7. exception object
try {
int result = 10 / 0; // This will cause an ArithmeticException
    } catch (ArithmeticException e) {
out.println("<p>ArithmeticException occurred: " + e.getMessage() + "</p>");
    }
%>

<%
// 8. page object
    String currentPage = page.toString();
%>
<p>Current Page: <%=currentPage%></p>
</body>
</html>

```

## Output:-





## **In this example:**

1. **'request'** object is used to get the request URI.
2. **'response'** object is used to set content type.
3. **'out'** object is used to print HTML content.
4. **'session'** object is used to set and get a session attribute.
5. **'application'** object is used to set and get an application attribute
6. **'pageContext'** object is used to set and get a page scope attribute.
7. **'exception'** object is used to catch and handle exceptions.
8. **'page'** object is used to get the current page information.