Poornima College of Engineering, Jaipur

Department of Computer Engineering

Advance Java Lab (5CS4-24)

B.Tech III Year, V Semester

Experiment – 05

Objective: Create a simple calculator application that demonstrates the use of RMI. You are not required to create GUI.

We have to run these four classes to make calculator using RMI.

- Calculator.java
- CalculatorImpl.java
- CalculatorClient.java
- CalculatorServer.java

Calculator.java

```
public interface Calculator
        extends   java.rmi.Remote   {
   public long add(long a, long b)
      throws java.rmi.RemoteException;

   public long sub(long a, long b) throws
      java.rmi.RemoteException;

   public long mul(long a, long b) throws
      java.rmi.RemoteException;

   public long div(long a, long b)
      throws java.rmi.RemoteException;
```

}

CalculatorImpl.java

```
public class CalculatorImpl extends
                java.rmi.server.UnicastRemoteObject
            implements Calculator {
            // Implementations must have an
            //explicit constructor
            // in order to declare the
            //RemoteException exception

            public CalculatorImpl()
                throws  java.rmi.RemoteException  {
                super();
            }
            public long add(long a, long b)
                throws  java.rmi.RemoteException  {
                return a + b;
            }
            public long sub(long a, long b)
                throws  java.rmi.RemoteException  {
                return a - b;
            }
            public long mul(long a, long b)
                throws java.rmi.RemoteException {
```

```
    return a * b;
  }

  public long div(long a, long b)
    throws java.rmi.RemoteException {
    return a / b;
  }}
```

CalculatorClient.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import      java.net.MalformedURLException;
import java.rmi.NotBoundException;

public class CalculatorClient {

  public static void main(String[] args) {
    try {
      Calculator c = (Calculator)
                Naming.lookup(
        "rmi://localhost/CalculatorService");
      System.out.println( c.sub(4, 3) );
      System.out.println( c.add(4, 5) );
      System.out.println( c.mul(3, 6) );
      System.out.println( c.div(9, 3) );
```

```
}
catch (MalformedURLException murle) {
   System.out.println();
   System.out.println(
   "MalformedURLException");
   System.out.println(murle);
}
catch (RemoteException re) {
   System.out.println();
   System.out.println(
            "RemoteException");
   System.out.println(re);
}
catch (NotBoundException nbe) {
   System.out.println();
   System.out.println(
           "NotBoundException");
   System.out.println(nbe);
}
catch (
   java.lang.ArithmeticException
                        ae)                {
   System.out.println();
   System.out.println(
   "java.lang.ArithmeticException");
   System.out.println(ae);}}
```

CalculatorServer.java

```
import java.rmi.Naming;

public class CalculatorServer {

  public CalculatorServer() {
    try {
      Calculator c = new CalculatorImpl();
      Naming.rebind("rmi://localhost:1099/CalculatorService", c);
    } catch (Exception e) {
      System.out.println("Trouble: " + e);
    }
  }

  public static void main(String args[]) {
    new CalculatorServer();
  }
}
```

//Now use rmic to create the stub and skeleton class files.

```
 > rmic CalculatorImpl
```

Running the RMI System

You are now ready to run the system! You need to start three consoles, one for the server, one for the client, and one for the RMIRegistry.

- *Start with the Registry. You must be in the directory that contains the classes you have written. From there, enter the following:*

  > rmiregistry

- *If all goes well, the registry will start running and you can switch to the next console.*

  In the second console start the server hosting the CalculatorService, and enter the following:

  > java CalculatorServer

- *It will start, load the implementation into memory and wait for a client connection.*

  In the last console, start the client program.

  > java CalculatorClient//

OUTPUT –

1

9

18

Advance JAVA Practical

6.1 Create Servlet that prints "Hello World".

```java
package com.example;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloWorldServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {

        // Set the content type of the response
        response.setContentType("text/html");

        // Get the PrintWriter to write the response
        PrintWriter out = res.getWriter();

        // Write "Hello World" response
        out.println("<html>");
        out.println("<head><title>Hello World</title></head>");
        out.println("<body>");
        out.println("<h1>Hello World</h1>");
        out.println("</body>");
        out.println("</html>");

        // Close the writer
        out.close();
    }
}
```

Web. xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app>

  <!-- Servlet Declaration -->
  <servlet>
    <servlet-name>HelloWorldServlet</servlet-name>
    <servlet-class>HelloWorldServlet</servlet-class>
  </servlet>

  <!-- Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>HelloWorldServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>

</web-app>
```

6.2 Create a Servlet that prints Today's Date.

```
import java.io.IOException;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DateServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");

        // Get today's date
        Date today = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat("EEEE, dd MMMM yyyy");
        String formattedDate = dateFormat.format(today);

        // Write the response
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Today's Date: " + formattedDate + "</h1>");
        out.println("</body></html>");
    }
}
```

```
<web-app >

  <servlet>
    <servlet-name>DateServlet</servlet-name>
    <servlet-class>DateServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>DateServlet</servlet-name>
    <url-pattern>/date</url-pattern>
  </servlet-mapping>

</web-app>
```

Q-6.3) Create Servlet for login page, if the username and password is correct then print message "Hello Username" else a message "Login Failed".

Step 1: Create the HTML Login Form

First, create an HTML form (e.g., login.html) to submit the username and password to the servlet.

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
</head>
<body>
  <h2>Login Page</h2>
  <form action="LoginServlet" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
    <br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <br><br>
    <input type="submit" value="Login">
  </form>
</body>
```

```
</html>
```

Step 2: Create the Servlet (LoginServlet.java)

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    // Dummy credentials for demonstration
    private static final String VALID_USERNAME = "user";
    private static final String VALID_PASSWORD = "password";

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Set response content type
        response.setContentType("text/html");

        // Get PrintWriter to write the response
        PrintWriter out = response.getWriter();

        // Get username and password from the request
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Validate credentials
        if (VALID_USERNAME.equals(username) && VALID_PASSWORD.equals(password)) {
            out.println("<h1>Hello " + username + "</h1>");
        } else {
            out.println("<h1>Login Failed</h1>");
        }

        out.close();
    }
}
```

Step 3: Configure the Servlet in web.xml (Optional)

```
<servlet>
  <servlet-name>LoginServlet</servlet-name>
  <servlet-class>LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>LoginServlet</servlet-name>
  <url-pattern>/LoginServlet</url-pattern>
</servlet-mapping>
```

Q-7

Create a servlet that implements ServletContextAttributeListener interface such that a message dialog is displayed whenever an attribute is added or removed or replaced.

```
package com.example;

import jakarta.servlet.ServletContext;
import jakarta.servlet.ServletContextAttributeEvent;
import jakarta.servlet.ServletContextAttributeListener;
import jakarta.servlet.annotation.WebListener;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

// Listener to handle attribute changes
@WebListener
public class ContextAttributeListener implements
ServletContextAttributeListener {

    @Override
    public void attributeAdded(ServletContextAttributeEvent event) {
        String name = event.getName();
        Object value = event.getValue();
        System.out.println("Attribute Added: Name=" + name + ", Value="
+ value);
    }

    @Override
    public void attributeRemoved(ServletContextAttributeEvent event) {
        String name = event.getName();
        System.out.println("Attribute Removed: Name=" + name);
    }

    @Override
    public void attributeReplaced(ServletContextAttributeEvent event) {
        String name = event.getName();
        Object newValue = event.getServletContext().getAttribute(name);
        System.out.println("Attribute Replaced: Name=" + name + ", New
Value=" + newValue);
    }
}



// Servlet to trigger attribute events
@WebServlet("/attribute")
```

```
public class AttributeServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        ServletContext context = getServletContext();

        // Add an attribute
        context.setAttribute("exampleAttribute", "InitialValue");

        // Replace the attribute
        context.setAttribute("exampleAttribute", "UpdatedValue");

        // Remove the attribute
        context.removeAttribute("exampleAttribute");

        response.getWriter().println("Attribute events triggered. Check
server logs for details.");
    }
}
```

8.1 Create a JSP that prints “Hello World”.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Hello World JSP</title>
</head>
<body>
    <h1>Hello World</h1>
</body>
</html>
```

8.2 Create a JSP that implements Decision –Making Statement.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Decision-Making in JSP</title>
</head>
<body>
    <h1>Decision-Making Example</h1>

    <%
        // Get the current hour of the day
        java.util.Calendar calendar = java.util.Calendar.getInstance();
        int hour = calendar.get(java.util.Calendar.HOUR_OF_DAY);

        // Decision-making logic
        if (hour >= 6 && hour < 12) {
    %>
        <p>Good Morning!</p>
    <%
        } else if (hour >= 12 && hour < 18) {
    %>
        <p>Good Afternoon!</p>
    <%
        } else if (hour >= 18 && hour < 21) {
    %>
        <p>Good Evening!</p>
    <%
        } else {
    %>
        <p>Good Night!</p>
    <%
        }
    %>
</body>
</html>
```

8.3 Create a JSP that add and subtract two numbers.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Add and Subtract Numbers</title>
</head>
<body>
    <h1>Add and Subtract Two Numbers</h1>
    <form method="post">
        <label for="num1">Enter First Number:</label>
        <input type="number" id="num1" name="num1" required>
        <br><br>

        <label for="num2">Enter Second Number:</label>
        <input type="number" id="num2" name="num2" required>
        <br><br>

        <input type="submit" value="Calculate">
    </form>

    <%
        // Retrieve numbers from user input
        String num1Str = request.getParameter("num1");
        String num2Str = request.getParameter("num2");

        if (num1Str != null && num2Str != null) {
            try {
                // Parse input as integers
                int num1 = Integer.parseInt(num1Str);
                int num2 = Integer.parseInt(num2Str);

                // Perform calculations
                int sum = num1 + num2;
                int difference = num1 - num2;

                // Display results
    %>
                <h2>Results:</h2>
                <p>Addition: <%= num1 %> + <%= num2 %> = <%= sum %></p>
                <p>Subtraction: <%= num1 %> - <%= num2 %> = <%= difference
%></p>
    <%
}

catch (NumberFormatException e) {
                // Handle invalid input
    %>
```

```
                <p style="color: red;">Please enter valid numbers.</p>
    <%
            }
        }
    %>
</body>
</html>
```

9.1 Create a JSP for login module

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login Module</title>
</head>
<body>
    <h1>Login</h1>
    <form method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>
        <br><br>

        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <br><br>

        <input type="submit" value="Login">
    </form>

    <%
        // Hardcoded credentials (for demonstration purposes only)
        String correctUsername = "admin";
        String correctPassword = "1234";

        // Retrieve user input
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        if (username != null && password != null) {
            // Validate credentials
            if (username.equals(correctUsername) &&
password.equals(correctPassword)) {
    %>
                <p style="color: green;">Login Successful! Welcome, <%=
username %>.</p>
    <%
            } else {
    %>
                <p style="color: red;">Invalid username or password. Please try
again.</p>
    <%
            }
        }
    %>
</body> </html>
```

9.2 Create a web page that prints 1 to 10 using JSTL.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <title>Print Numbers 1 to 10</title>
</head>
<body>
    <h1>Numbers from 1 to 10</h1>
    <ul>
        <!-- Use JSTL forEach loop to iterate over numbers -->
        <c:forEach var="i" begin="1" end="10">
            <li>${i}</li>
        </c:forEach>
    </ul>
</body>
</html>
```

Q-10

Create a custom JSP tag that prints current date and time. Use this tag into JSP page.

```
package customtags;

import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import jakarta.servlet.jsp.JspException;
import jakarta.servlet.jsp.tagext.TagSupport;

public class CurrentDateTimeTag extends TagSupport {
    @Override
    public int doStartTag() throws JspException {
        try {
            // Get the current date and time
            LocalDateTime now = LocalDateTime.now();
            DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
            String currentDateTime = now.format(formatter);

            // Print the date and time to the JSP
```

```
            pageContext.getOut().write("Current Date and Time: " +
currentDateTime);
        } catch (IOException e) {
            throw new JspException("Error in CurrentDateTimeTag", e);
        }
        return SKIP_BODY; // Skip the body content of the tag
    }
}
```