Distributed System and Cloud Computing Lab

Subject Code: MCAL32

A Practical Journal Submitted in Fulfilment
of the Degree of

MASTER

In

COMPUTER APPLICATION

Year 2024-2025

By

Mr. Jadhav Harshal Sanjay

(Application Id-53942)
Seat No.: 1030199

Semester-III

Under the Guidance of

Prof. Ashwini Padwal

Centre for Distance and Online Education
Vidya Nagari, Kalina, Santacruz East – 400098.
University of Mumbai

PCP Center

[Satish Pradhan Dyanasadhana College, Thane]

Institute of Distance and Open Learning
Vidya Nagari, Kalina, Santacruz East – 400098.

CERTIFICATE

This to certify that, **"Mr. Jadhav Harshal Sanjay"** appearing **Master's in computer application (Semester III) Application Id: 53942** has satisfactorily completed the prescribed practical of **MCAL32 - Distributed System and Cloud Computing Lab** as laid down by the University of Mumbai for the academic year 2024-25.

Teacher In Charge	External Examiner	Coordinator – M.C.A

Date:

Place: -

INDEX

Practical 1

Aim: Multithreaded Server Socket Program using java.

MultithreadedSocketServer.java

```
import java.net.ServerSocket;
import java.net.Socket;
public class MultithreadedSocketServer
{
public static void main(String[] args) throws Exception
{
try
{
// Create server socket on port 8888
ServerSocket server = new ServerSocket(8888);
int counter = 0;
System.out.println("Server Started.....");
while (true)
{
counter++;
// Accept client connection
Socket serverClient = server.accept();
System.out.println(" >> Client No:" + counter + " started!");
// Create and start a new thread for each client connection
ServerClientThread sct = new ServerClientThread(serverClient,
counter);
sct.start();
}
}
catch (Exception e)
{
System.out.println("Error: " + e);
}
}
}
```

ServerClientThread.java

```
import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.Socket;

class ServerClientThread extends Thread

{

Socket serverClient;

int clientNo;

int squre;

ServerClientThread(Socket inSocket,int counter)

{

serverClient = inSocket;

clientNo=counter;

}

public void run()

{

try

{

DataInputStream inStream = new

DataInputStream(serverClient.getInputStream());

DataOutputStream outStream = new

DataOutputStream(serverClient.getOutputStream());

String clientMessage="", serverMessage="";

while(!clientMessage.equals("bye"))

{

clientMessage=inStream.readUTF();

System.out.println("From Client-" +clientNo+ ": Number is :"+clientMessage);
```

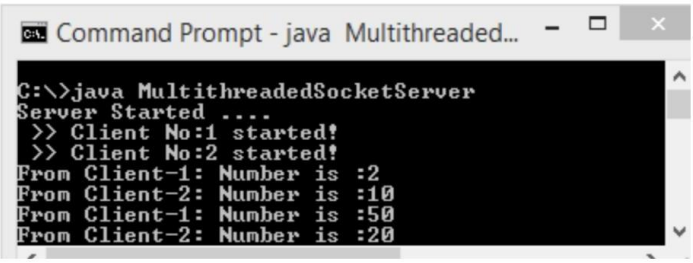
```
squre = Integer.parseInt(clientMessage) *
Integer.parseInt(clientMessage);
serverMessage="From Server to Client-" + clientNo + " Square of " +
clientMessage + " is " +squre;
outStream.writeUTF(serverMessage);
outStream.flush();
}
inStream.close();
outStream.close();
serverClient.close();
}
catch(Exception ex)
{
System.out.println(ex);
}
finally
{
System.out.println("Client -" + clientNo + " exit!! ");
}
}
}
```

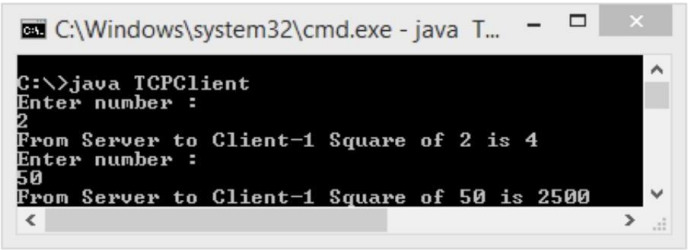
TCPClient.java

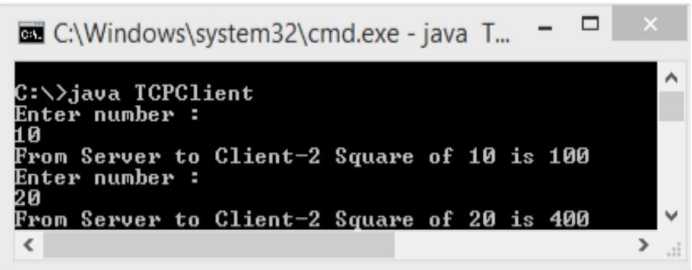
```
import java.io.*;
import java.net.*;
public class TCPClient
{
public static void main(String[] args) throws Exception
{
try
{
Socket socket=new Socket("127.0.0.1",8888);
DataInputStream inStream=new
DataInputStream(socket.getInputStream());
DataOutputStream outStream=new
DataOutputStream(socket.getOutputStream());
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
String clientMessage="",serverMessage="";
while(!clientMessage.equals("bye"))
{
System.out.println("Enter number :");
clientMessage=br.readLine();
outStream.writeUTF(clientMessage);
outStream.flush();
serverMessage=inStream.readUTF();
System.out.println(serverMessage);
```

```
}
outStream.close();
outStream.close();
socket.close();
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Output:







Practical 2

Aim: Write a java program print addition multiplication division subtraction using remote method call in client server program

RPCServer.java

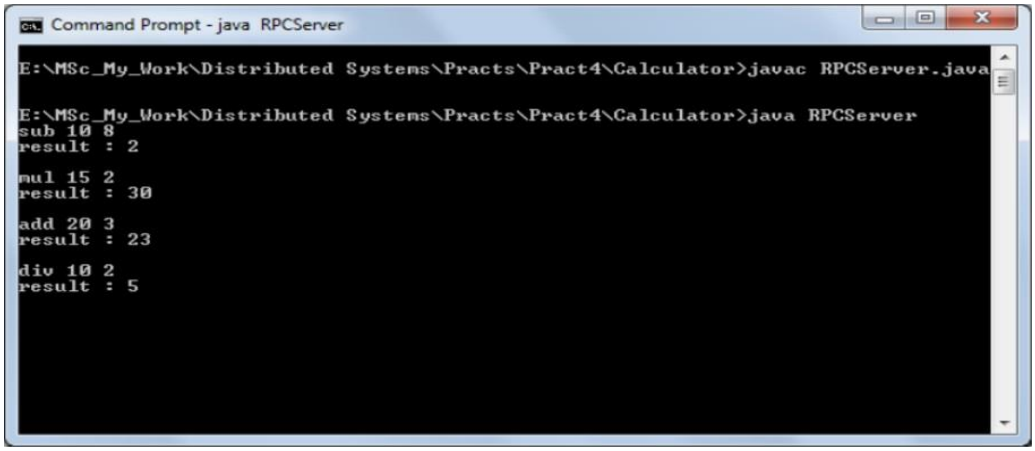
```
import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.StringTokenizer;

class RPCServer

{

DatagramSocket ds;

DatagramPacket dp;

String str,methodName,result;

int val1,val2;

RPCServer()

{

try

{

ds=new DatagramSocket(1200);

byte b[]=new byte[4096];

while(true)

{

dp=new DatagramPacket(b,b.length);

ds.receive(dp);

str=new String(dp.getData(),0,dp.getLength());
```

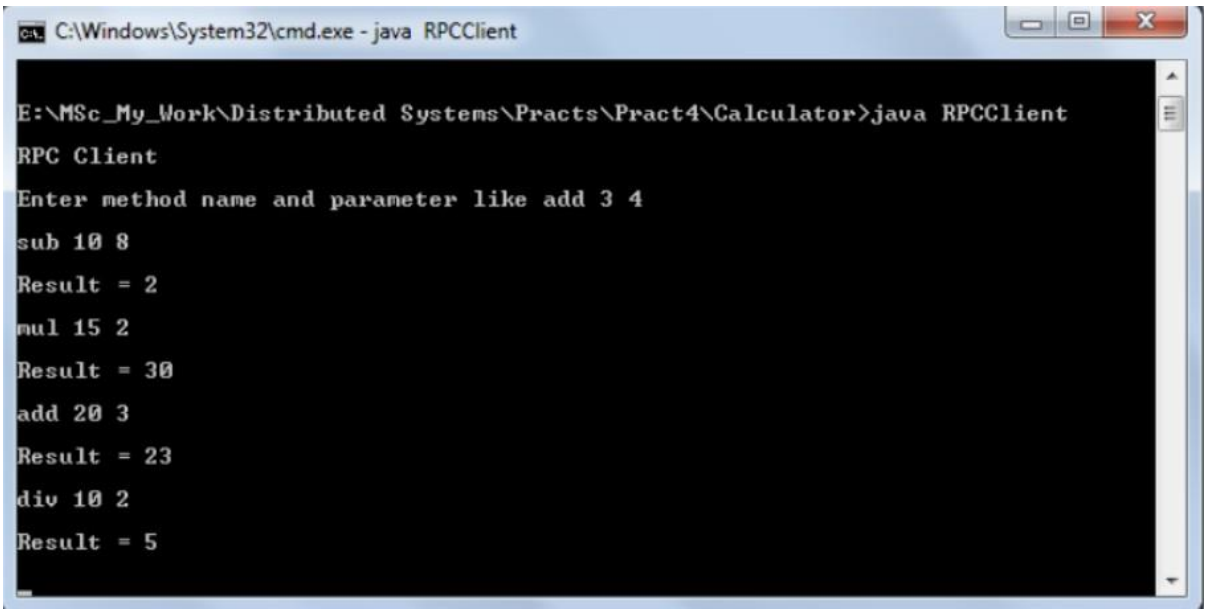
```
if(str.equalsIgnoreCase("q"))
{
System.exit(1);
}
else
{
StringTokenizer st = new StringTokenizer(str," ");
int i=0;
while(st.hasMoreTokens())
{
String token=st.nextToken();
methodName=token;
val1 = Integer.parseInt(st.nextToken());
val2 = Integer.parseInt(st.nextToken());
}
}
System.out.println(str);
InetAddress ia = InetAddress.getLocalHost();
if(methodName.equalsIgnoreCase("add"))
{
result= "" + add(val1,val2);
}
else if(methodName.equalsIgnoreCase("sub"))
{
result= "" + sub(val1,val2);
}
```

```
else if(methodName.equalsIgnoreCase("mul"))
{
result= "" + mul(val1,val2);
}
else if(methodName.equalsIgnoreCase("div"))
{
result= "" + div(val1,val2);
}
byte b1[]=result.getBytes();
DatagramSocket ds1 = new DatagramSocket();
DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
System.out.println("result : "+result+"\n");
ds1.send(dp1);
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
public int add(int val1, int val2)
{
return val1+val2;
}
public int sub(int val3, int val4)
```

```
{
return val3-val4;
}
public int mul(int val3, int val4)
{
return val3*val4;
}
public int div(int val3, int val4)
{
return val3/val4;
}
public static void main(String[] args)
{
new RPCServer();
}
}
```

Output:




C:\Windows\System32\cmd.exe - java RPCClient
E:\MSc_My_Work\Distributed Systems\Practs\Pract4\Calculator>java RPCClient
RPC Client
Enter method name and parameter like add 3 4
sub 10 8
Result = 2
mul 15 2
Result = 30
add 20 3
Result = 23
div 10 2
Result = 5

Practical 3

Aim: Write a client server program to print datetime while establishing connection between client and server

Server-side program:

```
import java.io.DataOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Date;

class DateServer

{

public static void main(String args[]) throws Exception

{

ServerSocket s=new ServerSocket(5217);

while(true)

{

System.out.println("Waiting For Connection ...");

Socket soc=s.accept();

DataOutputStream out=new

DataOutputStream(soc.getOutputStream());

out.writeBytes("Server Date: " + (new Date()).toString() + "\n");

out.close();

soc.close();

}

}

}
```

Client side:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.Socket;
class DateClient
{
public static void main(String args[]) throws Exception
{
Socket soc=new Socket(InetAddress.getLocalHost(),5217);
BufferedReader in=new BufferedReader(new
InputStreamReader(soc.getInputStream() ));
System.out.println(in.readLine());
}
}
```

Output:

Server Date: Fri Nov 16 17:05:42 IST 2016

Practical 4

Aim: Demonstrate a sample RMI Java application.

Server.java

```
import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class Server extends ImplExample {

public Server() {}

public static void main(String args[]) {

try {

// Instantiating the implementation class

ImplExample obj = new ImplExample();

// Exporting the object of implementation class

// (here we are exporting the remote object to the stub)

Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

// Binding the remote object (stub) in the registry

Registry registry = LocateRegistry.getRegistry();

registry.bind("Hello", stub);

System.err.println("Server ready");

} catch (Exception e) {

System.err.println("Server exception: " + e.toString());

e.printStackTrace();

}

}

}
```

Client.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class Client {
private Client() {}
public static void main(String[] args) {
try {
// Getting the registry
Registry registry = LocateRegistry.getRegistry(null);
// Looking up the registry for the remote object
Hello stub = (Hello) registry.lookup("Hello");
// Calling the remote method using the obtained object
stub.printMsg();
// System.out.println("Remote method invoked");
} catch (Exception e) {
System.err.println("Client exception: " + e.toString());
e.printStackTrace();
}
}
}
```

Output:

C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>javac *.java

C:\EXAMPLES\rmi>start rmiregistry

C:\EXAMPLES\rmi>
```

C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>javac *.java

C:\EXAMPLES\rmi>
```

C:\WINDOWS\system32\cmd.exe - java Server

```
C:\EXAMPLES\rmi>java Server
Server ready
```

C:\Program Files\Java\jdk1.8.0_101\bin\rmiregistry.exe

```
C:\WINDOWS\system32\cmd.exe - java Client
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>java Client
```

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>javac *.java

C:\EXAMPLES\rmi>start rmiregistry

C:\EXAMPLES\rmi>java Server
Server ready
This is an example RMI program
```

Practical 5

Aim: The program illustrates the concept of thread synchronization and mutual exclusion using ReentrantLock in Java.

```
import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

public class Main {

private static int count = 0; // Shared variable

private static final Lock lock = new ReentrantLock(); // Lock for mutual exclusion

// Runnable task for threads

public static void runThread() {

lock.lock(); // Acquire lock before entering critical section

try {

long threadId = Thread.currentThread().getId();

System.out.println("Thread " + threadId + ": Current value of count = " + count);

System.out.println("Thread " + threadId + " incrementing count...");

count++; // Modify shared variable

Thread.sleep(1000); // Simulate work

System.out.println("Value of count after incremented by thread " + threadId + " = " + count);

} catch (InterruptedException e) {

System.out.println("Thread interrupted");

} finally {

lock.unlock(); // Release lock after leaving critical section

}

}

public static void main(String[] args) {

Thread[] threads = new Thread[4];
```

```
// Create and start threads

for (int i = 0; i < 4; i++) {

threads[i] = new Thread(Main::runThread);

threads[i].start();

}

// Wait for all threads to complete

for (int i = 0; i < 4; i++) {

try {

threads[i].join();

} catch (InterruptedException e) {

System.out.println("Thread interrupted");

}

}

System.out.println("Final value of count: " + count);

}

}
```

Output:

```
E:\P5>java Main
Thread 21: Current value of count = 0
Thread 21 incrementing count...
Value of count after incremented by thread 21 = 1
Thread 22: Current value of count = 1
Thread 22 incrementing count...
Value of count after incremented by thread 22 = 2
Thread 23: Current value of count = 2
Thread 23 incrementing count...
Value of count after incremented by thread 23 = 3
Thread 24: Current value of count = 3
Thread 24 incrementing count...
Value of count after incremented by thread 24 = 4
Final value of count: 4
```

Practical 6

Aim: To set up a virtual machine in Azure using IaaS for complete infrastructure control.

1. IaaS (Infrastructure as a Service): Creating a Storage Account in Azure

1. **Log into the Azure Portal**:
   - Open Azure Portal.
2. **Navigate to Storage Accounts**:
   - From the left-hand menu, select **Storage accounts** to view a list of storage accounts. If the portal menu is hidden, click the menu icon to display it.
3. **Create a New Storage Account**:
   - Click on **Create** at the top of the Storage accounts page.
   - Select **Resource group**: Create a new resource group or use an existing one.
   - Enter a **Name** for your storage account and choose your **Region**.
4. **Set Configuration Options**:
   - Set additional configurations in the **Basics** and **Advanced** tabs if needed (e.g., replication type, access tier).
5. **Review and Create**:
   - Click **Review + Create**, then **Create** after the validation completes.

---

2. PaaS (Platform as a Service): Deploying a Node.js App Using Azure DevOps and GitHub Actions

1. **Access DevOps Starter**:
   - In the Azure Portal, type **DevOps Starter** in the search bar, then click on **Add**.
2. **Set GitHub as CI/CD Provider**:
   - Select **GitHub Actions** as your CI/CD provider to automate deployment.
3. **Choose Application Framework**:
   - Select **Node.js** as the application language and then click **Next**.
   - Under **Application Framework**, choose **Express.js**.
4. **Set Up Azure Web App**:
   - For deployment, select **Windows Web App** as the environment and click **Next**.
5. **Configure and Deploy**:
   - Choose your subscription, resource group, and region, then complete the setup.

- DevOps Starter will create the resources in Azure and set up a GitHub Actions workflow to automate deployment.

---

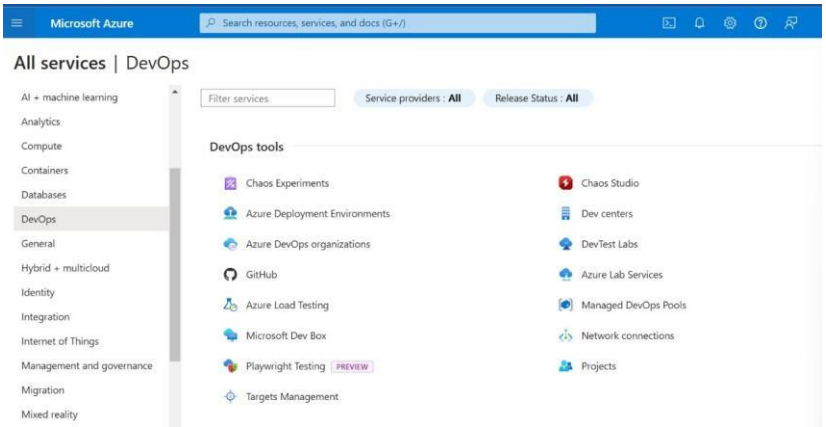
3. SaaS (Software as a Service): Managing a Web Application in Azure

1. **Open the App Services Section**:
   - In the Azure Portal, go to **App Services**.
2. **Manage Your Web Application**:
   - Select the web application you previously deployed.
   - On the **Overview** page, you can perform basic management tasks like starting, stopping, or restarting the app.
3. **Configure App Settings**:
   - Use the left-hand menu for more advanced settings, such as authentication, app configurations, and scaling options.

Understanding Cloud Service Categories in Use

- **IaaS**: You created an Azure storage account, which provides raw infrastructure (storage) without any specific software stack, allowing you full control.
- **PaaS**: Using Azure DevOps and GitHub Actions to deploy an app simplifies the development pipeline with an environment pre-configured for application deployment.
- **SaaS**: Managing the app through Azure's interface (App Services) abstracts away hardware and OS management, offering a full software experience with the platform handling infrastructure.

OUTPUT:



Practical 7

Aim: To demonstrate login form using swing

NewPage.java

```
// Import required classes
import javax.swing.*;

// Create NewPage class to show a welcome message
class NewPage extends JFrame {
    // Constructor
    NewPage(String username) {
        // Set up the frame
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setTitle("Welcome");
        setSize(400, 200);

        // Add welcome label
        JLabel welLabel = new JLabel("Welcome, " + username);
        welLabel.setHorizontalAlignment(SwingConstants.CENTER);
        getContentPane().add(welLabel);
    }
}
```

CreateLoginForm.java

```
// Import required classes and packages
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

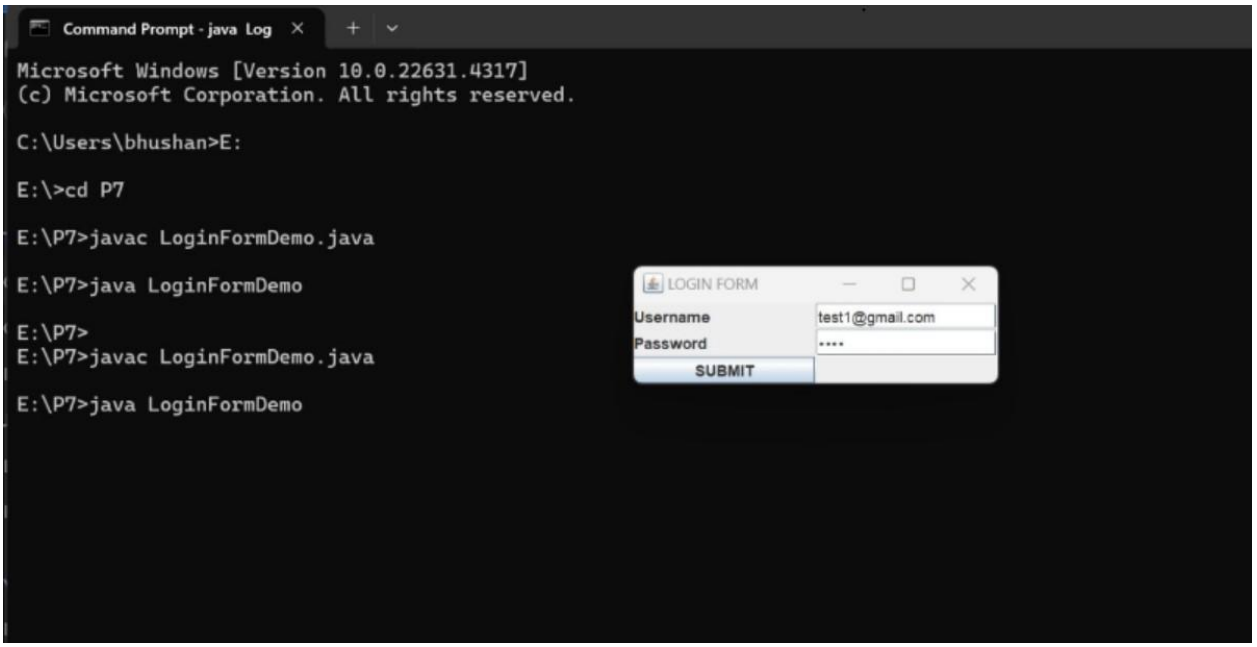
```
// CreateLoginForm class to create login form
// Class extends JFrame to create a window and implements ActionListener to handle button click
class CreateLoginForm extends JFrame implements ActionListener {
    // Initialize button, panel, labels, and text fields
    JButton b1;
    JPanel newPanel;
    JLabel userLabel, passLabel;
    final JTextField textField1;
    final JPasswordField textField2;

    // Constructor
    CreateLoginForm() {
        // Set up the labels for username and password
        userLabel = new JLabel("Username");
        textField1 = new JTextField(15); // Username field

        passLabel = new JLabel("Password");
        textField2 = new JPasswordField(15); // Password field

        // Create submit button
        b1 = new JButton("SUBMIT");

        // Create panel and add components to it
        newPanel = new JPanel(new GridLayout(3, 2));
        newPanel.add(userLabel);
        newPanel.add(textField1);
        newPanel.add(passLabel);
        newPanel.add(textField2);
        newPanel.add(b1);
```

```
        // Add the panel to the frame
        add(newPanel, BorderLayout.CENTER);

        // Set frame properties
        setTitle("LOGIN FORM");
        setSize(300, 100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Register button click with ActionListener
        b1.addActionListener(this);
    }

    // Define actionPerformed() to handle button click
    public void actionPerformed(ActionEvent ae) {
        String userValue = textField1.getText(); // Get entered username
        String passValue = new String(textField2.getPassword()); // Get entered password

        // Check if credentials are valid
        if (userValue.equals("test1@gmail.com") && passValue.equals("test")) {
            // Navigate to new page if valid
            NewPage page = new NewPage(userValue);
            page.setVisible(true);
            dispose(); // Close the login form
        } else {
            // Show error message if credentials are invalid
            JOptionPane.showMessageDialog(this, "Invalid Username or Password");
        }
    }
}
```

LoginFormDemo.java

```
// Main class
public class LoginFormDemo {
    public static void main(String[] args) {
        try {
            // Create an instance of the login form
            CreateLoginForm form = new CreateLoginForm();
            form.setVisible(true); // Show login form
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT



Practical 8

Aim: Dynamically changing the background color of a webpage on each click

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Change Background Color</title>
<!-- Include jQuery from Google CDN -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body style="text-align:center;" id="body">
<!-- Heading to display instructions -->
<h1>Enter Your Color Choice</h1>
<!-- Button that will trigger color change on click -->
<button type="button" onclick="changecolor()">Change Color</button>
<script>
// JavaScript function to generate a random color and change the background color
function changecolor() {
// Generate a random hexadecimal color code
var color = "#" + (Math.random() * 16777215 | 0).toString(16);
// Use jQuery to change the background color of the body
$("body").css("background-color", color);
}
```

```
</script>

</body>

</html>
```

OUTPUT: