Deccan Education Society’s

Navinchandra Mehta Institute of Technology and Development

CERTIFICATE

This is to certify that Mr. **Siddhesh Dhumak** of M.C.A. Semester III with Roll No. **C23031** has completed **All** practicals of **Distributed System and Cloud Computing Lab** undermy supervision in this college during the year 2024- 2025.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CO | R1  (Attendance) | R2  (Performance during lab session) | R3  (Innovation in problem solving technique) | R4  (Mock Viva) | R5  (Variation in implementation of learnt topics on projects) |
| CO1 |  |  |  |  |  |
| CO2 |  |  |  |  |  |
| CO3 |  |  |  |  |  |
| CO4 |  |  |  |  |  |

Practical-in-charge Head of Department MCA Department

(NMITD)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MCAL32 Distributed System and Cloud Computing Lab Index** | | | | |
| **Sr.**  **No.** | **Topic Name** | **Date** | **CO** | **Sign** |
| **1.** | **Remote Process Communication:**  To develop a program for multi-client chat server using Socket |  | **CO1** |  |
| **2.** | **Remote Procedure Call:** |  | **CO1** |  |
| a) Implement a Server calculator containing ADD (), MUL (), SUB () | **CO2** |
| etc. |  |
| b) Implement a Date Time Server containing date() and time() |  |
| 1. Implement a Server to do the following    1. User will enter two values in hours and minutes from the client (Eg: 3 hour 40 minutes and 2 hour 50 minutes)    2. Server will add hour and minute components separately The client side receives the result and display the value       1. only in hours(Eg: 06:30 Hours)       2. only in minutes(Eg:390 Minutes) |  |
| 1. Implement a Server to do the following:    1. Get two numbers from the client.    2. Server processing the multiplication of the above two numbers    3. Server sends the processed data to the client and client checks whether the multiplication is greater than 250 or not |  |
| 1. Implement a Server to do the following    1. Get a string from the client.    2. Find the reverse of the string at the server side    3. The client side receives the reversed string and verifying whether it is palindrome |  |
| 1. Implement a Server to do the following    1. Get one number from the client.    2. Server will find whether the number from client is odd or even    3. After receiving the result from server client displays the message    4. If the number is even client will calculate the table of the number |  |
| **3.** | **Remote Method Invocation** |  | **CO1** |  |
| 1. Calculate addition of two numbers and send it to the client | **CO2** |
| using **RMI**. |  |
| 2. Retrieve time and date function from server to client. This |  |
| program should display server date and time by implementing **RMI** |  |
| **Equation Solver:**  3. The client should provide the values of a and b. The server will |  |
| solve the equation (a+b)2=a2+2ab+b2 and will give back the value of equation using **RMI**. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1. The client should provide the values of a and b. The server will solve the equation (a+b)2=a2+2ab+b2 and (a-b)2=a2+2ab+b2 and will give back the value of equation (Use **RMI**) 2. The client should provide the values of a, b & c. The server will solve the equation (ax2 + bx + c = 0) and will give back value of x.   If a = 1, b = 5 and c = 6 then return value will be x = -2 or x =-3.   1. Find X, where ‘Y’ and ‘n’ values provided. The client should provide equation and values to the server through an interface. The server will solve the expression given by the client. Use **RMI** X=Yn |  |  |  |
| **4.** | **Remote Method Invocation Graphical User Interface**   1. Design a Graphical User Interface for addition of two   numbers. Implement using RMI   1. Design a Graphical User Interface (GUI) to find factorial of a given numbers. Implement using RMI 2. Design a Graphical User Interface to convert Celsius to Fahrenheit. Implement using RMI 3. Design a Graphical User Interface (GUI) based BMI calculator by implementing RMI 4. Design a Graphical User Interface (GUI) based Basic calculator by implementing RMI 5. Design a Graphical User Interface (GUI) to find greatest of two numbers. Implement using RMI 6. Design a Graphical User Interface (GUI) which accepts a numerical value from the client. Convert the number in to words   .Implement using RMI   1. Design a Graphical User (GUI) Interface for reversing a given number. Implement using RMI 2. Design a Graphical User Interface (GUI) two find GCD of two numbers. Implement using RMI |  | **CO1 CO2** |  |
| **5** | Using MySQL create a Library database. Create table Book (Book\_id, Book\_name, Book\_author) and retrieve the Book information from  the Library database using the Remote Object Communication concept. |  | **CO2** |  |
| **6** | Using MySQL create a Student database. Create table student\_data(ID ,NAME , BRANCH ,PERCENTAGE ,EMAIL ) and retrieve the student\_data information from Student database using Remote Object Communication concept. |  | **CO2** |  |
| **7** | Using MySQL create ElecrticBill database. Create table Bill(consumer\_name, bill\_due\_date, bill\_amount) and retrieve the  Bill information from the ElecrticBill database using Remote Object Communication concept. |  | **CO2** |  |
| **8** | Implementation of mutual exclusion using Token ring algorithm |  | **CO3** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **9** | Implementation of Identity Management. |  | **CO4**  **CO5** |  |

Practical 1: Remote Process Communication: To develop a program for multi-client chat server using Socket

Server:

ServerThread.java package server;

//Client Main

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

import java.util.ArrayList;

public class ServerThread extends Thread{ private Socket socket;

private ArrayList<ServerThread> threadList; private PrintWriter output;

public ServerThread(Socket socket, ArrayList<ServerThread> threads) { this.socket = socket;

this.threadList = threads;

}

@Override public void run() {

try {

// TODO Auto-generated method stub

BufferedReader input = new BufferedReader(new

InputStreamReader(socket.getInputStream()));

output = new PrintWriter(socket.getOutputStream(),true);

while(true) {

String outputString = input.readLine(); if(outputString.equals("exit")) {

break;

}

printToAllClients(outputString); System.out.println("Server recieved"+outputString);

}

} catch (Exception e) {

// TODO: handle exception System.out.println("Error occured in main of the

server"+e.getStackTrace());

}

}

private void printToAllClients(String outputString) {

// TODO Auto-generated method stub for(ServerThread sT:threadList) {

sT.output.println(outputString);

}

}

}

Server:

Main.java package server;

import java.net.ServerSocket; import java.net.Socket; import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

ArrayList<ServerThread> threadList = new ArrayList<>(); try(ServerSocket serverSocket = new ServerSocket(5000)){

Socket socket = serverSocket.accept();

ServerThread serverThread = new ServerThread(socket,threadList); threadList.add(serverThread);

serverThread.start();

}catch(Exception e) {

System.out.println("Error occured in the main of server:"+e.getStackTrace());

}

}

}

Client:

ClientThread.java package client;

import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; import java.net.Socket;

public class ClientThread extends Thread {

private Socket socket; private BufferedReader input;

public ClientThread(Socket s) throws IOException{ this.socket = s; this.input = new BufferedReader(new

InputStreamReader(socket.getInputStream()));

}

@Override public void run() {

// TODO Auto-generated method stub try {

while(true) {

String response = input.readLine();

System.out.println(response);

}

}catch(IOException e) {

e.printStackTrace();

}finally {

try {

input.close();

}catch(IOException e) {

e.printStackTrace();

}

}

}

}

Client:

Main.java package client;

import java.io.BufferedReader; import java.io.InputStreamReader; import java.io.PrintWriter; import

java.net.Socket; import java.util.Scanner;

public class Main {

public static void main(String[] args) {

try(Socket socket = new Socket("localhost",5000)) {

@SuppressWarnings("unused")

BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter output = new PrintWriter(socket.getOutputStream(), true); Scanner sc = new Scanner(System.in);

String userInput; @SuppressWarnings("unuse d") String response;

String clientName = "empty";

ClientThread clientThread = new ClientThread(socket); clientThread.start();

do {

if(clientName.equals("empty")) { System.out.println("Enter your name:"); userInput = sc.nextLine(); output.println(userInput);

if(userInput.equals("exit")

) { break;

}

}else {

String message = ("("+clientName+")"+"message:"); System.out.println(message);

userInput = sc.nextLine(); output.println(message+" "+userInput); if(userInput.equals("exit")) {

break;

}

}

}while(!userInput.equals("exit")); sc.close();

} catch (Exception e) {

// TODO Auto-generated catch block System.out.println("Exception in client main"+e.getStackTrace());

}

}

}

Practical 2: Remote Procedure Call

1. calculator

Client:

package calc;

import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

public class CalculatorClient { public static void main(String[]

args) { try {

Registry registry = LocateRegistry.getRegistry("localhost", 1099); CalculatorInterface stub = (CalculatorInterface) registry.lookup("CalculatorService");

System.out.println("2 + 3 = " + stub.add(2, 3)); System.out.println("10 - 4 = " + stub.subtract(10, 4)); System.out.println("4 \* 5 = " + stub.multiply(4, 5)); System.out.println("20 / 4 = " + stub.divide(20, 4));

} catch (Exception e) {

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

}

}

}

Interface:

package calc;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface CalculatorInterface extends Remote { int add(int a, int b) throws RemoteException;

int subtract(int a, int b) throws RemoteException; int multiply(int a, int b) throws RemoteException; int divide(int a, int b) throws RemoteException;

}

Server:

package calc;

import java.rmi.RemoteException; import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class CalculatorServer extends UnicastRemoteObject implements CalculatorInterface

{

protected CalculatorServer() throws RemoteException { super();

}

public int add(int a, int b)

{ return a + b;

}

public int subtract(int a, int b) { return a - b;

}

public int multiply(int a, int b) { return a \* b;

}

public int divide(int a, int b) { return a / b;

}

public static void main(String[] args) { try {

CalculatorInterface stub = new CalculatorServer(); Registry registry = LocateRegistry.createRegistry(1099); registry.rebind("CalculatorService", stub); System.out.println("Calculator Server is ready.");

} catch (Exception e) {

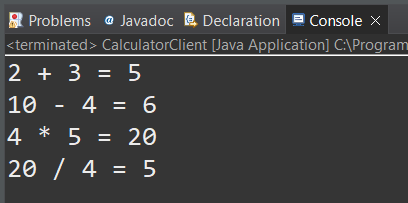
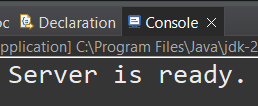
System.err.println("Server exception: " + e.toString()); e.printStackTrace();

}

}

}

o/p:



1. datetime Client:

package datetime;

import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

public class DateTimeClient { public static void main(String[]

args) { try {

Registry registry = LocateRegistry.getRegistry("localhost", 1020);

DateTimeInterface stub = (DateTimeInterface) registry.lookup("DateTimeService");

System.out.println("Current Date: " + stub.date()); System.out.println("Current Time: " + stub.time());

} catch (Exception e) {

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

}

}

}

Interface:

package datetime;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface DateTimeInterface extends Remote { String date() throws RemoteException;

String time() throws RemoteException;

}

Server:

package datetime;

import java.rmi.RemoteException; import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.rmi.server.UnicastRemoteObject; import java.text.SimpleDateFormat; import java.util.Date;

public class DateTimeServer extends UnicastRemoteObject implements DateTimeInterface

{

protected DateTimeServer() throws RemoteException { super();

}

public String date() {

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM- dd"); return dateFormat.format(new Date());

}

public String time() {

SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss"); return timeFormat.format(new Date());

}

public static void main(String[] args) { try {

DateTimeInterface stub = new DateTimeServer(); Registry registry = LocateRegistry.createRegistry(1020);

registry.rebind("DateTimeService", stub); System.out.println("DateTime Server is ready.");

} catch (Exception e) {

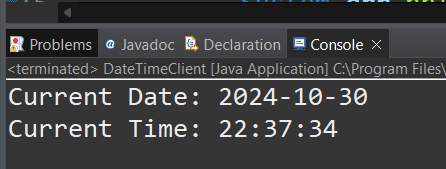
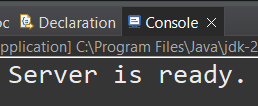
System.err.println("Server exception: " + e.toString()); e.printStackTrace();

}

}

}

o/p:



1. TimeAddition Client:

package timeAddition; import

java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry; import java.util.Scanner;

public class TimeAdditionClient {

public static void main(String[] args) {

try {

Registry registry = LocateRegistry.getRegistry("localhost", 1029); TimeAdditionInterface stub = (TimeAdditionInterface)

registry.lookup("TimeAdditionService");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter first time (hours and minutes) separated by space (e.g., 3 40): ");

int hours1 = scanner.nextInt(); int minutes1 = scanner.nextInt();

System.out.print("Enter second time (hours and minutes) separated by space (e.g., 2 50): ");

int hours2 = scanner.nextInt(); int minutes2 = scanner.nextInt();

int[] result = stub.addTime(hours1, minutes1, hours2, minutes2);

System.out.printf("Total Time: %02d:%02d Hours\n", result[0], result[1]); System.out.printf("Total Time in Minutes: %d Minutes\n", (result[0] \* 60 +

result[1]));

scanner.close();

} catch (Exception e) {

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

}

}

}

Interface:

package timeAddition;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface TimeAdditionInterface extends Remote {

int[] addTime(int hours1, int minutes1, int hours2, int minutes2) throws RemoteException;

}

Server:

package timeAddition;

import java.rmi.RemoteException; import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class TimeAdditionServer extends UnicastRemoteObject implements TimeAdditionInterface {

protected TimeAdditionServer() throws RemoteException { super();

}

public int[] addTime(int hours1, int minutes1, int hours2, int minutes2) { int totalMinutes = (hours1 \* 60 + minutes1) + (hours2

\* 60 + minutes2); int totalHours = totalMinutes / 60; totalMinutes = totalMinutes % 60;

return new int[] { totalHours, totalMinutes };

}

public static void main(String[] args) { try {

TimeAdditionInterface stub = new TimeAdditionServer(); Registry registry = LocateRegistry.createRegistry(1029); registry.rebind("TimeAdditionService", stub); System.out.println("Time Addition Server is ready.");

} catch (Exception e) {

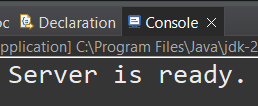
System.err.println("Server exception: " + e.toString()); e.printStackTrace();

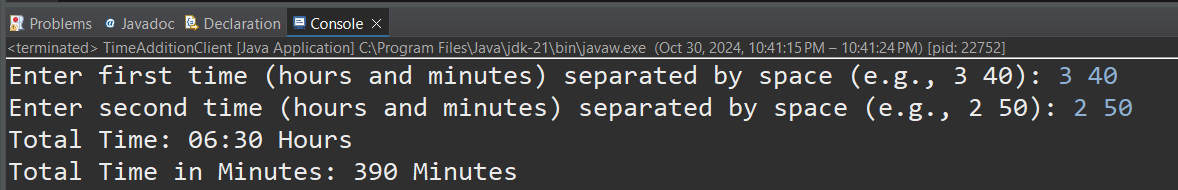
}

}

}

o/p:





1. Multiplication Client:

package multiplication; import

java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry; import java.util.Scanner;

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

try {

Registry registry = LocateRegistry.getRegistry("localhost", 8090); MultiplicationInterface stub = (MultiplicationInterface)

registry.lookup("MultiplicationService");

Scanner scanner = new Scanner(System.in); System.out.print("Enter first number:

"); int num1 = scanner.nextInt();

System.out.print("Enter second number: "); int num2 = scanner.nextInt();

int result = stub.multiply(num1, num2); System.out.println("Multiplication Result: " + result);

if (result > 250)

{

System.out.println("The result is greater than 250.");

} else {

System.out.println("The result is not greater than 250.");

}

scanner.close();

} catch (Exception e) {

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

}

}

}

Interface:

package multiplication;

import java.rmi.Remote;

import java.rmi.RemoteException;

}

Server:



public interface MultiplicationInterface extends Remote

{ int multiply(int a, int b) throws RemoteException;

package multiplication;

import java.rmi.RemoteException; import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class MultiplicationServer extends UnicastRemoteObject implements MultiplicationInterface {

protected MultiplicationServer() throws RemoteException { super();

}

public int multiply(int a, int b) { return a \* b;

}

public static void main(String[] args) { try {

MultiplicationInterface stub = new MultiplicationServer(); Registry registry = LocateRegistry.createRegistry(8090); registry.rebind("MultiplicationService", stub); System.out.println("Multiplication Server is ready.");

} catch (Exception e) {

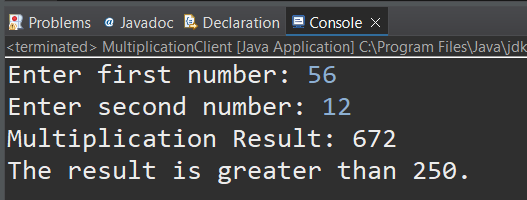
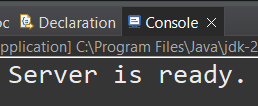
System.err.println("Server exception: " + e.toString()); e.printStackTrace();

}

}

}

o/p:



1. String

Client:

package stringreverse; import

java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.util.Scanner;

public class StringReverseClient { public static void main(String[]

args) { try {

Registry registry = LocateRegistry.getRegistry("localhost", 7009); StringReverseInterface stub = (StringReverseInterface)

registry.lookup("StringReverseService");

Scanner scanner = new Scanner(System.in); System.out.print("Enter a string: ");

String input = scanner.nextLine();

String reversed = stub.reverseString(input); System.out.println("Reversed String: " + reversed);

// Check if palindrome

if (input.equalsIgnoreCase(reversed)) { System.out.println("The string is a palindrome.");

} else {

System.out.println("The string is not a palindrome.");

}

scanner.close();

} catch (Exception e) {

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

}

}

}

Interface:



package stringreverse;

import java.rmi.Remote;

import java.rmi.RemoteException;

}

Server:

public interface StringReverseInterface extends Remote

{ String reverseString(String input) throws RemoteException;

package stringreverse;

import java.rmi.RemoteException; import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class StringReverseServer extends UnicastRemoteObject implements StringReverseInterface {

protected StringReverseServer() throws RemoteException { super();

}

public String reverseString(String input) { String reversed = new

StringBuilder(input).reverse().toString(); return reversed;

}

public static void main(String[] args) { try {

StringReverseInterface stub = new StringReverseServer(); Registry registry = LocateRegistry.createRegistry(7009);

registry.rebind("StringReverseService", stub); System.out.println("String Reverse Server is ready.");

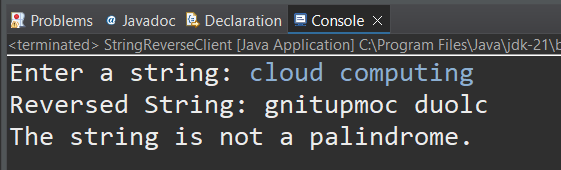
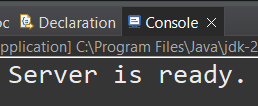
} catch (Exception e) {

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

}

}

}

o/p:

1. Oddeven

Client:

package oddeven; import

java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.util.Scanner;

public class OddEvenClient { public static void main(String[]

args) { try {

Registry registry = LocateRegistry.getRegistry("localhost", 2098); OddEvenInterface stub = (OddEvenInterface)

registry.lookup("OddEvenService");

Scanner scanner = new Scanner(System.in);



System.out.print("Enter a number:

"); int number = scanner.nextInt();

String result = stub.checkOddEven(number); System.out.println("The number is: " + result);

// If the number is even, calculate the multiplication table if ("Even".equals(result)) {

System.out.println("Multiplication Table for " + number + ":"); for (int i = 1; i <= 10; i++) {

System.out.println(number + " x " + i + " = " + (number \* i));

}

}

scanner.close();

} catch (Exception e) {

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

}

}

}

Interface:

package oddeven;

import java.rmi.Remote;

import java.rmi.RemoteException;

}

Server:

public interface OddEvenInterface extends Remote {

String checkOddEven(int number) throws RemoteException;

package oddeven;

import java.rmi.RemoteException; import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class OddEvenServer extends UnicastRemoteObject implements OddEvenInterface {

protected OddEvenServer() throws RemoteException {

super();

}

public String checkOddEven(int number) { String result;

if (number % 2 == 0) { result = "Even";

} else {

result = "Odd";

}

return result;

}

public static void main(String[] args) { try {

OddEvenInterface stub = new OddEvenServer(); Registry registry = LocateRegistry.createRegistry(2098); registry.rebind("OddEvenService", stub); System.out.println("OddEven Server is ready.");

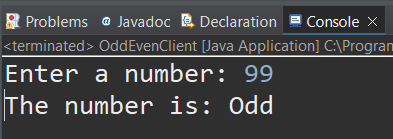
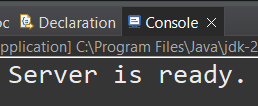
} catch (Exception e) {

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

}

}

}

o/p:

Practical 3: Remote Method Invocation

1. Addition

Client



package addition; import java.rmi.Naming;

public class AdditionClient {

public static void main(String[] args)

{ try {

AdditionInterface addition = (AdditionInterface) Naming.lookup("rmi://localhost/AdditionService");

int result = addition.add(5, 10); System.out.println("Result of addition: " + result);

} catch (Exception e) { System.out.println("Client exception: " + e);

}

}

}

Interface:

package addition;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface AdditionInterface extends Remote { int add(int a, int b) throws RemoteException;

}

Implementation:

package addition;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

@SuppressWarnings("serial")

public class Addition extends UnicastRemoteObject implements AdditionInterface {

public Addition() throws RemoteException { } @Override

public int add(int a, int b) throws RemoteException { return a + b;

}

}



Server:

package addition;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class AdditionServer {

public static void main(String[] args)

{ try {

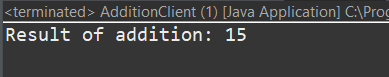
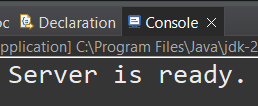
LocateRegistry.createRegistry(1099); // Start RMI registry Addition addition = new Addition(); Naming.rebind("rmi://localhost/AdditionService", addition); System.out.println("Addition Server is ready.");

} catch (Exception e) { System.out.println("Addition Server failed: " + e);

}

}

}

o/p:

1. Timeserver Client:

package timeservice;

import java.rmi.Naming;

public class TimeClient {

public static void main(String[] args)

{ try {

TimeService timeService = (TimeService) Naming.lookup("rmi://localhost/TimeService");

String currentTime = timeService.getCurrentTime(); System.out.println("Current Date and Time from Server: " +

currentTime);

} catch (Exception e) { System.out.println("Client exception: " + e);

}

}

}

Interface:



package timeservice;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface TimeService extends Remote { String getCurrentTime() throws RemoteException;

}

Implementation:

package timeservice;

import java.rmi.RemoteException; import java.rmi.server.UnicastRemoteObject; import java.text.SimpleDateFormat; import java.util.Date;

@SuppressWarnings("serial")

public class TimeServiceImpl extends UnicastRemoteObject implements TimeService {

protected TimeServiceImpl() throws RemoteException { }

@Override

public String getCurrentTime() throws RemoteException { SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd

HH:mm:ss");

return sdf.format(new Date());

}

}



Server:

package timeservice; import

java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class TimeServer {

public static void main(String[] args)

{ try {

LocateRegistry.createRegistry(1020); // Start RMI registry TimeService timeService = new TimeServiceImpl(); Naming.rebind("rmi://localhost/TimeService", timeService); System.out.println("Time Server is ready.");

} catch (Exception e) {

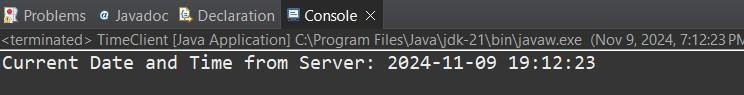
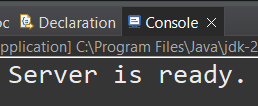
System.out.println("Time Server failed: " + e);

}

}

}

o/p:



1. Equation
   1. Eqn1

Client:



package eqn1;

import java.rmi.Naming; import java.util.Scanner;

public class EquationClient {

public static void main(String[] args)

{ try {

EquationService service = (EquationService) Naming.lookup("rmi://localhost/EquationService");

@SuppressWarnings("resource")

Scanner scanner = new Scanner(System.in); System.out.print("Enter value for a: "); int a = scanner.nextInt();

System.out.print("Enter value for b: "); int b = scanner.nextInt();

double result = service.calculateEquation(a, b); System.out.println("Result of (a + b)^2: " + result);

} catch (Exception e) { System.out.println("Client exception: " + e);

}

}

}

Interface:

package eqn1;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface EquationService extends Remote {

double calculateEquation(int a, int b) throws RemoteException;

}

Implementation:

package eqn1;



import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

@SuppressWarnings("serial")

public class EquationServiceImpl extends UnicastRemoteObject implements EquationService {

protected EquationServiceImpl() throws RemoteException { }

@Override

public double calculateEquation(int a, int b) throws RemoteException {

return Math.pow((a + b), 2); // (a + b)^2

}

}

Server:

package eqn1;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class EquationServer {

public static void main(String[] args)

{ try {

LocateRegistry.createRegistry(1021); // Start RMI registry EquationService service = new EquationServiceImpl(); Naming.rebind("rmi://localhost/EquationService", service); System.out.println("Equation Server is ready.");

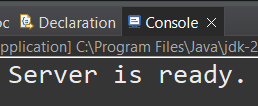
} catch (Exception e) { System.out.println("Equation Server failed: " + e);

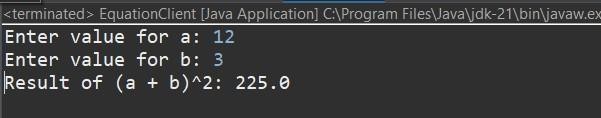
}

}

}

o/p:





* 1. Eqn2

Client:

package eqn2;

import java.rmi.Naming; import java.util.Scanner;

public class EquationClient {

public static void main(String[] args)

{ try {

EquationService service = (EquationService) Naming.lookup("rmi://localhost/EquationService");

@SuppressWarnings("resource")

Scanner scanner = new Scanner(System.in); System.out.print("Enter value for a: "); int a = scanner.nextInt();

System.out.print("Enter value for b: "); int b = scanner.nextInt();

double sumSquare = service.calculateSumSquare(a, b); double differenceSquare = service.calculateDifferenceSquare(a,

b);

System.out.println("Result of (a + b)^2: " + sumSquare); System.out.println("Result of (a - b)^2: " + differenceSquare);

} catch (Exception e) { System.out.println("Client exception: " + e);

}

}

}

Interface:

package eqn2;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface EquationService extends Remote { double calculateSumSquare(int a, int b) throws

RemoteException; double calculateDifferenceSquare(int a, int b) throws

RemoteException;

}



Implementation:

package eqn2;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

@SuppressWarnings("serial")

public class EquationServiceImpl extends UnicastRemoteObject implements EquationService {

protected EquationServiceImpl() throws RemoteException { }

@Override

public double calculateSumSquare(int a, int b) throws RemoteException {

return Math.pow((a + b), 2); // (a + b)^2

}

@Override

public double calculateDifferenceSquare(int a, int b) throws RemoteException {

return Math.pow((a - b), 2); // (a - b)^2

}

}

Server:

package eqn2;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class EquationServer {

public static void main(String[] args)

{ try {

LocateRegistry.createRegistry(9080); // Start RMI registry EquationService service = new EquationServiceImpl(); Naming.rebind("rmi://localhost/EquationService", service); System.out.println("Equation Server is ready.");

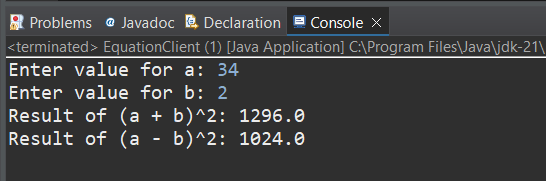
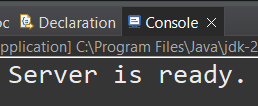
} catch (Exception e) { System.out.println("Equation Server failed: " + e);

}

}

}

o/p:



* 1. Eqn3 Client:

package eqn3;

import java.rmi.Naming; import java.util.Scanner;

public class QuadraticEquationClient {



public static void main(String[] args)

{ try {

QuadraticEquationService service = (QuadraticEquationService) Naming.lookup("rmi://localhost/QuadraticEquationService");

@SuppressWarnings("resource")

Scanner scanner = new Scanner(System.in); System.out.print("Enter value for a: "); int a = scanner.nextInt();

System.out.print("Enter value for b:

"); int b = scanner.nextInt();

System.out.print("Enter value for c:

"); int c = scanner.nextInt();

double[] roots = service.solveQuadraticEquation(a, b, c);

System.out.println("Roots of the equation are: "); System.out.println("x1 = " + roots[0]); System.out.println("x2 = " + roots[1]);

} catch (Exception e) { System.out.println("Client exception: " + e);

}

}

}

Interface:

package eqn3;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface QuadraticEquationService extends Remote { double[] solveQuadraticEquation(int a, int b, int c) throws

RemoteException;

}

Implementation:

package eqn3;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class QuadraticEquationServiceImpl extends UnicastRemoteObject implements QuadraticEquationService {



protected QuadraticEquationServiceImpl() throws RemoteException { }

@Override

public double[] solveQuadraticEquation(int a, int b, int c) throws RemoteException {

double[] roots = new double[2]; double discriminant = b \* b - 4 \* a \* c;

if (discriminant < 0) {

throw new RemoteException("No real roots.");

}

roots[0] = (-b + Math.sqrt(discriminant)) / (2 \* a); // First root roots[1] = (-b - Math.sqrt(discriminant)) / (2 \* a); // Second root

}

Server:

return roots;

}

package eqn3;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class QuadraticEquationServer

{ public static void main(String[] args) {

try {

LocateRegistry.createRegistry(4567); // Start RMI registry QuadraticEquationService service = new

QuadraticEquationServiceImpl(); Naming.rebind("rmi://localhost/QuadraticEquationService",

service);

System.out.println("Quadratic Equation Server is ready.");

} catch (Exception e) {

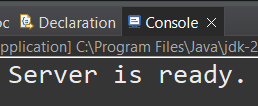
System.out.println("Quadratic Equation Server failed: " + e);

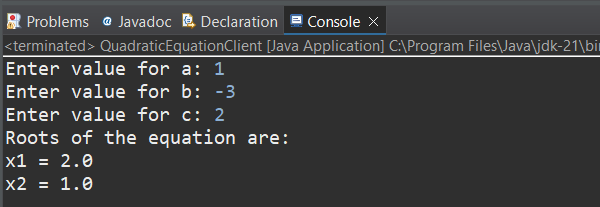
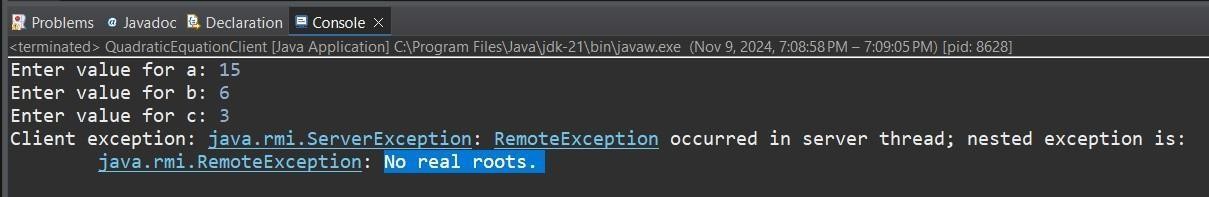
}

}

}

o/p:





* 1. Eqn4 Client:

package eqn4;

import java.rmi.Naming; import java.util.Scanner;

public class PowerClient {

public static void main(String[] args)

{ try {

PowerService service = (PowerService) Naming.lookup("rmi://localhost/PowerService");

@SuppressWarnings("resource")

Scanner scanner = new Scanner(System.in); System.out.print("Enter value for Y: "); double Y = scanner.nextDouble();

System.out.print("Enter value for n: "); int n = scanner.nextInt();

double result = service.calculatePower(Y, n); System.out.println("Result of Y^n: " + result);

} catch (Exception e) { System.out.println("Client exception: " + e);

}

}

}



Interface:

package eqn4;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface PowerService extends Remote { double calculatePower(double Y, int n) throws RemoteException;

}

Implementation:

package eqn4;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

@SuppressWarnings("serial")

public class PowerServiceImpl extends UnicastRemoteObject implements PowerService {

protected PowerServiceImpl() throws RemoteException { }

@Override

public double calculatePower(double Y, int n) throws RemoteException {

return Math.pow(Y, n); // Calculate Y^n

}

}

Server:

package eqn4;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class PowerServer {

public static void main(String[] args)

{ try {

LocateRegistry.createRegistry(3466); // Start RMI registry PowerService service = new PowerServiceImpl(); Naming.rebind("rmi://localhost/PowerService", service); System.out.println("Power Server is ready.");

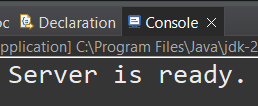
} catch (Exception e) { System.out.println("Power Server failed: " + e);

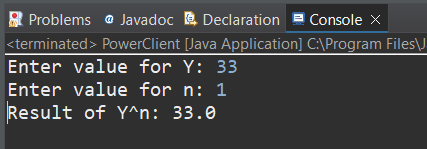
}

}

}

o/p:





Practical 4: Remote Method Invocation GUI

1. Gui1- addition Client:

package gui1;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

public class AdditionClient extends JFrame {

private JTextField num1Field, num2Field, resultField;

public AdditionClient() { setTitle("RMI Addition Client"); setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(null);

JLabel label1 = new JLabel("Number 1:"); label1.setBounds(10, 20, 80, 25); add(label1);

num1Field = new JTextField(); num1Field.setBounds(100, 20, 165, 25); add(num1Field);

JLabel label2 = new JLabel("Number 2:"); label2.setBounds(10, 50, 80, 25); add(label2);

num2Field = new JTextField(); num2Field.setBounds(100, 50, 165, 25); add(num2Field);

JButton addButton = new JButton("Add"); addButton.setBounds(10, 80, 80, 25);

add(addButton);



resultField = new JTextField(); resultField.setBounds(100, 110, 165,

* 1. ; resultField.setEditable(false); add(resultField);

addButton.addActionListener(new ActionListener() { @Override

public void actionPerformed(ActionEvent e) { try {

AdditionService service = (AdditionService) Naming.lookup("rmi://localhost/AdditionService");

int num1 = Integer.parseInt(num1Field.getText()); int num2 = Integer.parseInt(num2Field.getText()); int result = service.add(num1, num2); resultField.setText(String.valueOf(result));

} catch (Exception ex) { ex.printStackTrace();

}

}

});

}

public static void main(String[] args) { AdditionClient client = new AdditionClient(); client.setVisible(true);

}

}

Interface:

package gui1;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface AdditionService extends Remote { int add(int a, int b) throws RemoteException;

}

Implementation:

package gui1;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class AdditionServiceImpl extends UnicastRemoteObject implements AdditionService {

protected AdditionServiceImpl() throws RemoteException { super();

}

@Override

public int add(int a, int b) throws RemoteException

{ return a + b;

}

}



Server:

package gui1;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class AdditionServer { public static void main(String[]

args) { try {

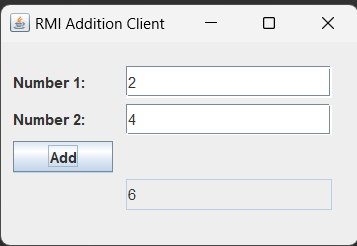
LocateRegistry.createRegistry(1099); // Start RMI registry AdditionService service = new AdditionServiceImpl(); Naming.rebind("AdditionService", service); System.out.println("Addition Service is running...");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui2- factorial Client:

package gui2;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.math.BigInteger; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class FactorialClient extends JFrame { private JTextField inputField, resultField;

public FactorialClient() { setTitle("Factorial Calculator"); setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(null);

JLabel inputLabel = new JLabel("Enter a number:"); inputLabel.setBounds(10, 20, 120,

* 1. ; add(inputLabel);

inputField = new JTextField(); inputField.setBounds(130, 20, 150, 25);

add(inputField);



JButton calculateButton = new JButton("Calculate"); calculateButton.setBounds(10, 60, 120, 25); add(calculateButton);

resultField = new JTextField(); resultField.setBounds(10, 100, 270, 25); resultField.setEditable(false); add(resultField);

calculateButton.addActionListener(new ActionListener() { @Override

public void actionPerformed(ActionEvent e) { try {

int number = Integer.parseInt(inputField.getText()); FactorialService service = (FactorialService)

Naming.lookup("rmi://localhost/FactorialService"); BigInteger result = service.factorial(number); resultField.setText("Factorial: " + result.toString());

} catch (Exception ex) { ex.printStackTrace();

resultField.setText("Error: " + ex.getMessage());

}

}

});

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

FactorialClient client = new FactorialClient(); client.setVisible(true);

});

}

}

Interface:

package gui2; import

java.math.BigInteger; import java.rmi.Remote;

import java.rmi.RemoteException;

public interface FactorialService extends Remote {

BigInteger factorial(int number) throws RemoteException;

}

Implementation:



package gui2;

import java.math.BigInteger; import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class FactorialServiceImpl extends UnicastRemoteObject implements FactorialService {

protected FactorialServiceImpl() throws RemoteException { super();

}

@Override

public BigInteger factorial(int number) throws RemoteException { BigInteger result = BigInteger.ONE;

for (int i = 1; i <= number; i++) {

result = result.multiply(BigInteger.valueOf(i));

}

Server:

}

return result;

}

package gui2;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class FactorialServer { public static void main(String[]

args) { try {

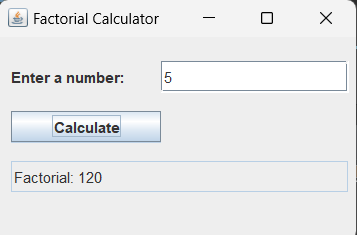
LocateRegistry.createRegistry(3490); // Start RMI registry FactorialService service = new FactorialServiceImpl(); Naming.rebind("rmi://localhost/FactorialService", service); System.out.println("Factorial Service is running...");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui3- temperature

Client:

package gui3;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class TemperatureClient extends JFrame { private JTextField celsiusField, fahrenheitField;

public TemperatureClient() {

setTitle("Celsius to Fahrenheit Converter"); setSize(300, 200); setDefaultCloseOperation(JFrame.EXIT\_ON\_CL OSE); setLayout(null);

JLabel celsiusLabel = new JLabel("Celsius:"); celsiusLabel.setBounds(10, 20, 80, 25);

add(celsiusLabel);



celsiusField = new JTextField(); celsiusField.setBounds(100, 20, 165,

* 1. ; add(celsiusField);

JButton convertButton = new JButton("Convert"); convertButton.setBounds(10, 60, 120, 25); add(convertButton);

fahrenheitField = new JTextField(); fahrenheitField.setBounds(10, 100, 255,

25); fahrenheitField.setEditable(false); add(fahrenheitField);

convertButton.addActionListener(new ActionListener() { @Override

public void actionPerformed(ActionEvent e) { try {

double celsius = Double.parseDouble(celsiusField.getText()); TemperatureConverter service = (TemperatureConverter)

Naming.lookup("rmi://localhost/TemperatureService");

double fahrenheit = service.celsiusToFahrenheit(celsius); fahrenheitField.setText("Fahrenheit: " + fahrenheit);

} catch (Exception ex) { ex.printStackTrace();

fahrenheitField.setText("Error: " + ex.getMessage());

}

}

});

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

TemperatureClient client = new TemperatureClient(); client.setVisible(true);

});

}

}

Interface:

package gui3;

import java.rmi.Remote;

import java.rmi.RemoteException;

}

Impl:



public interface TemperatureConverter extends Remote {

double celsiusToFahrenheit(double celsius) throws RemoteException;

package gui3;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class TemperatureConverterImpl extends UnicastRemoteObject implements TemperatureConverter {

protected TemperatureConverterImpl() throws RemoteException { super();

}

@Override

public double celsiusToFahrenheit(double celsius) throws RemoteException { return (celsius \* 1.8) + 32;

}

}

Server:

package gui3;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class TemperatureServer { public static void main(String[]

args) { try {

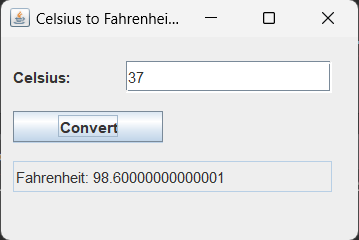
LocateRegistry.createRegistry(2342); // Start RMI registry TemperatureConverter service = new TemperatureConverterImpl(); Naming.rebind("rmi://localhost/TemperatureService", service); System.out.println("Temperature Conversion Service is running...");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui4- BMI

Client:

package gui4;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class BMICalculatorClient extends JFrame { private JTextField weightField, heightField, resultField;

public BMICalculatorClient() { setTitle("BMI Calculator"); setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(null);

JLabel weightLabel = new JLabel("Weight (kg):"); weightLabel.setBounds(10, 20, 100, 25); add(weightLabel);

weightField = new JTextField(); weightField.setBounds(120, 20, 150,

25);

add(weightField);

JLabel heightLabel = new JLabel("Height (m):"); heightLabel.setBounds(10, 60, 100,

* 1. ; add(heightLabel);

heightField = new JTextField(); heightField.setBounds(120, 60, 150, 25); add(heightField);

JButton calculateButton = new JButton("Calculate BMI"); calculateButton.setBounds(10, 100, 150, 25); add(calculateButton);

resultField = new JTextField(); resultField.setBounds(10, 140, 260, 25); resultField.setEditable(false); add(resultField);

calculateButton.addActionListener(new ActionListener() { @Override

public void actionPerformed(ActionEvent e) { try {

double weight = Double.parseDouble(weightField.getText()); double height = Double.parseDouble(heightField.getText()); BMICalculator service = (BMICalculator)

Naming.lookup("rmi://localhost/BMICalculatorService"); double bmi = service.calculateBMI(weight, height);

resultField.setText("Your BMI: " + String.format("%.2f", bmi));

} catch (Exception ex) { ex.printStackTrace();

resultField.setText("Error: " + ex.getMessage());

}

}

});

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

BMICalculatorClient client = new BMICalculatorClient(); client.setVisible(true);

});

}

}

Interface:

package gui4;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface BMICalculator extends Remote {

double calculateBMI(double weight, double height) throws RemoteException;

}



Impl:

package gui4;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class BMICalculatorImpl extends UnicastRemoteObject implements BMICalculator {

protected BMICalculatorImpl() throws RemoteException { super();

}

@Override

public double calculateBMI(double weight, double height) throws RemoteException {

return weight / (height \* height);

}

}

Server:

package gui4;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class BMIServer {

public static void main(String[] args) { try {

LocateRegistry.createRegistry(7658); // Start RMI registry BMICalculator service = new BMICalculatorImpl(); Naming.rebind("rmi://localhost/BMICalculatorService", service);

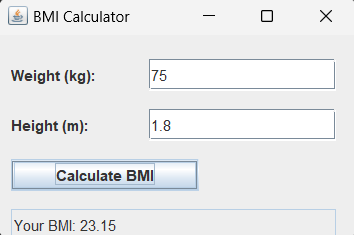
System.out.println("BMI Calculator Service is running...");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui5- calculator Client:

package gui5;

import java.awt.Container; import java.awt.GridBagConstraints; import java.awt.GridBagLayout;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JTextField; import javax.swing.SwingUtilities;

public class CalculatorClient extends JFrame implements ActionListener { String n1 = "", n2 = "", op = "";

double x, y, result; boolean f = true, dotf, rf; ICalculator intf;

GridBagConstraints gbc = new GridBagConstraints();

JTextField txt1 = new JTextField(20); JButton btn[] = new JButton[16];

String[] btnname = {"C", "1", "2", "3", "4", "5", "6", "-", "7", "8", "9", "\*", "0", ".", "=", "/"};

public CalculatorClient() {

// Initialize RMI and interface here try {

// Corrected URL for the RMI lookup

intf = (ICalculator) Naming.lookup("//localhost:5060/CalculatorServer");

} catch (Exception e) {

e.printStackTrace(

); System.exit(1);

}

Container con = this.getContentPane(); con.setLayout(new GridBagLayout()); gbc.weightx = 1.0;

gbc.weighty = 1.0;

for (int b = 0; b < btn.length; b++) { btn[b] = new JButton(btnname[b]); btn[b].addActionListener(this);

}

gbc.gridx = 0;

gbc.gridy = 0; gbc.gridwidth = 4; con.add(txt1, gbc); gbc.gridwidth = 1;

gbc.gridx = 0;

gbc.gridy = 1;

con.add(btn[0], gbc); // Clear button

int i = 1;

for (int k = 2; k <= 5; k++) { for (int j = 0; j <= 3; j++) {

gbc.gridx = j; gbc.gridy = k;

if (i < btn.length) { con.add(btn[i], gbc); i++;

}

}

}

setSize(300, 300);

setVisible(true); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public void actionPerformed(ActionEvent ae) { String cmd = ae.getActionCommand();

if (cmd.equals("C")) {

txt1.setText(""); n1 = n2 = op = ""; x = y = result

= 0; f = true; dotf = false; rf = false;

} else if (cmd.equals("+") || cmd.equals("-") || cmd.equals("\*") || cmd.equals("/")) {

if (f) {

n1 = txt1.getText(); x =

Double.parseDouble(n1); f = false;

dotf = false; rf = false;

}

op = cmd;

txt1.setText(""); // Clear for next input

} else if (cmd.equals("=")) { n2

= txt1.getText(); y =

Double.parseDouble(n2); try

{

switch (op) { case "+":

result = intf.add(x, y); break;

case "-":

result = intf.sub(x, y); break;

case "\*":

result = intf.mul(x, y); break;

case "/":

result = intf.div(x, y); break;

}



} catch (Exception e) { e.printStackTrace();

}

txt1.setText(Double.toString(result

)); f = true; // Reset for next operation

} else if (cmd.equals("."))

{ if (!dotf) { txt1.setText(txt1.getText() + cmd); dotf = true;

}

} else {

if (!rf) {

txt1.setText(txt1.getText() + cmd);

}

}

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

new CalculatorClient();

});

}

}

Interface:

package gui5;

import java.rmi.\*;

public interface ICalculator extends Remote

{

double add (double x, double y) throws RemoteException; double sub (double x,double y) throws RemoteException; double mul (double x, double y) throws RemoteException; double div

(double x,double y)throws RemoteException;

}

Impl:

package gui5;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

}}

Server:

public class CalculatorImpl extends UnicastRemoteObject implements ICalculator

{



public CalculatorImpl() throws RemoteException{

}

public double add (double x, double y) throws RemoteException{ return (x+y);

}

public double sub (double x, double y) throws RemoteException{ return (x-y);

}

public double mul (double x, double y) throws RemoteException{ return (x\*y);

}

public double div (double x, double y) throws RemoteException{ return (x/y);

package gui5;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class CalculatorServer { public static void main(String[]

args) { try {

// Create and export the registry on port 5060 LocateRegistry.createRegistry(5060); CalculatorImpl ci = new CalculatorImpl();

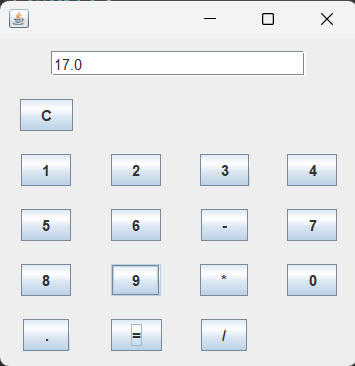
Naming.rebind("//localhost:5060/CalculatorServer", ci); System.out.println("Calculator Server is ready on port 5060");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui6- greatest number Client:

package gui6;

import java.awt.GridLayout;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class GreatestNumberClient extends JFrame {

private JTextField numberField1, numberField2, resultField;

public GreatestNumberClient() { setTitle("Greatest of Two Numbers"); setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(4, 2));

JLabel label1 = new JLabel("Enter first number:"); JLabel label2 = new JLabel("Enter second number:"); JLabel resultLabel = new JLabel("Greatest number:");

numberField1 = new JTextField(); numberField2 = new JTextField(); resultField = new JTextField(); resultField.setEditable(false);

JButton calculateButton = new JButton("Find Greatest"); calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) { try {

int num1 = Integer.parseInt(numberField1.getText()); int num2 = Integer.parseInt(numberField2.getText());

GreatestNumberService service = (GreatestNumberService) Naming.lookup("rmi://localhost/GreatestNumberService");

int greatest = service.findGreatest(num1, num2); resultField.setText(String.valueOf(greatest));

} catch (Exception ex) { ex.printStackTrace();

resultField.setText("Error: " + ex.getMessage());

}

}

});

add(label1); add(numberField1); add(label2); add(numberField2); add(calculateButton); add(resultLabel); add(resultField);

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

GreatestNumberClient client = new GreatestNumberClient(); client.setVisible(true);

});

}

}

Interface:

package gui6;

import java.rmi.Remote;

import java.rmi.RemoteException;

}

Impl:



public interface GreatestNumberService extends Remote { int findGreatest(int a, int b) throws RemoteException;

package gui6; import

java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class GreatestNumberServiceImpl extends UnicastRemoteObject implements GreatestNumberService {

protected GreatestNumberServiceImpl() throws RemoteException {

super();

}

@Override

public int findGreatest(int a, int b) throws RemoteException

{ return Math.max(a, b);

}

}

Server:

package gui6;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class GreatestNumberServer

{ public static void main(String[] args) {

try {

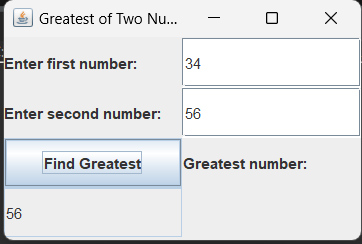
LocateRegistry.createRegistry(1245); // Start RMI registry GreatestNumberService service = new GreatestNumberServiceImpl(); Naming.rebind("rmi://localhost/GreatestNumberService", service); System.out.println("Greatest Number Service is running...");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui7- numbers to words Client:

package gui7;

import java.awt.GridLayout;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class NumberToWordsClient extends JFrame

{ private JTextField numberField, resultField;

public NumberToWordsClient() { setTitle("Number to Words Converter"); setSize(300, 150);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(3, 2));



JLabel numberLabel = new JLabel("Enter a number:"); JLabel resultLabel = new JLabel("In words:");

numberField = new JTextField(); resultField = new JTextField(); resultField.setEditable(false);

JButton convertButton = new JButton("Convert"); convertButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) { try {

int number = Integer.parseInt(numberField.getText()); NumberToWordsService service =

(NumberToWordsService) Naming.lookup("rmi://localhost/NumberToWordsService");

String words = service.convertNumberToWords(number); resultField.setText(words);

} catch (Exception ex) { ex.printStackTrace(); resultField.setText("Error: Invalid input");

}

}

});

add(numberLabel); add(numberField); add(convertButton); add(resultLabel); add(resultField);

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

NumberToWordsClient client = new NumberToWordsClient(); client.setVisible(true);

});

}

}

Interface:

package gui7;

import java.rmi.Remote;

import java.rmi.RemoteException;

}

Impl:



public interface NumberToWordsService extends Remote {

String convertNumberToWords(int number) throws RemoteException;

package gui7;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class NumberToWordsServiceImpl extends UnicastRemoteObject implements NumberToWordsService {

protected NumberToWordsServiceImpl() throws RemoteException { super();

}

@Override

public String convertNumberToWords(int number) throws RemoteException { if (number < 0) {

return "Negative " + convertNumberToWords(-number);

}

if (number == 0) { return "Zero";

}

String[] units = {

"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"

};

String[] tens = {

"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"

};

StringBuilder words = new StringBuilder();

if (number >= 1000) {



words.append(units[number / 1000]).append(" Thousand "); number %= 1000;

}

if (number >= 100) {

words.append(units[number / 100]).append(" Hundred "); number %= 100;

}

if (number >= 20) {

words.append(tens[number / 10]).append(" "); number %= 10;

}

if (number > 0) { words.append(units[number]).append(" ");

}

}

Server:

return words.toString().trim();

}

package gui7;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class NumberToWordsServer

{ public static void main(String[] args) {

try {

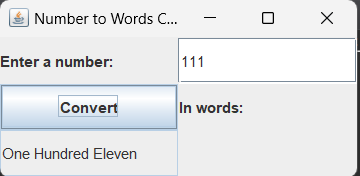
LocateRegistry.createRegistry(6751); // Start RMI registry NumberToWordsService service = new NumberToWordsServiceImpl(); Naming.rebind("rmi://localhost/NumberToWordsService", service); System.out.println("Number to Words Service is running...");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui8- reverse number

Client:

package gui8;

import java.awt.GridLayout;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class ReverseNumberClient extends JFrame { private JTextField numberField, resultField;

public ReverseNumberClient() { setTitle("Reverse Number"); setSize(300, 150);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(3, 2));

JLabel numberLabel = new JLabel("Enter a number:"); JLabel resultLabel = new JLabel("Reversed number:");

numberField = new JTextField(); resultField = new JTextField(); resultField.setEditable(false);

JButton reverseButton = new JButton("Reverse");



reverseButton.addActionListener(new ActionListener() { @Override

public void actionPerformed(ActionEvent e) { try {

int number = Integer.parseInt(numberField.getText()); ReverseNumberService service =

(ReverseNumberService) Naming.lookup("rmi://localhost/ReverseNumberService");

int reversed = service.reverseNumber(number); resultField.setText(String.valueOf(reversed));

} catch (Exception ex) { ex.printStackTrace(); resultField.setText("Error: Invalid input");

}

}

});

add(numberLabel); add(numberField); add(reverseButton); add(resultLabel); add(resultField);

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

ReverseNumberClient client = new ReverseNumberClient(); client.setVisible(true);

});

}

}

Interface:

package gui8;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface ReverseNumberService extends Remote { int reverseNumber(int number) throws RemoteException;

}



Impl:

package gui8;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class ReverseNumberServiceImpl extends UnicastRemoteObject implements ReverseNumberService {

protected ReverseNumberServiceImpl() throws RemoteException { super();

}

@Override

public int reverseNumber(int number) throws RemoteException { int reversed = 0;

while (number != 0) {

int digit = number % 10; reversed = reversed \* 10 + digit; number /= 10;

}

return reversed;

}

}

Server:

package gui8;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class ReverseNumberServer

{ public static void main(String[] args) {

try {

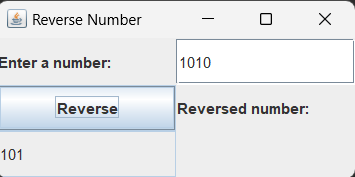
LocateRegistry.createRegistry(8712); // Start RMI registry ReverseNumberService service = new ReverseNumberServiceImpl(); Naming.rebind("rmi://localhost/ReverseNumberService", service); System.out.println("Reverse Number Service is running...");

} catch (Exception e) { e.printStackTrace();

}

}

}



1. Gui9- gcd

Client:

package gui9;

import java.awt.GridLayout;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class GCDClient extends JFrame {

private JTextField numberField1, numberField2, resultField;

public GCDClient() { setTitle("GCD Calculator"); setSize(300, 150);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(4, 2));

JLabel label1 = new JLabel("Enter first number:"); JLabel label2 = new JLabel("Enter second number:"); JLabel resultLabel = new JLabel("GCD:");

numberField1 = new JTextField(); numberField2 = new JTextField(); resultField = new JTextField();

resultField.setEditable(false);



JButton calculateButton = new JButton("Calculate GCD"); calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) { try {

int num1 = Integer.parseInt(numberField1.getText()); int num2 = Integer.parseInt(numberField2.getText()); GCDService service =

(GCDService) Naming.lookup("rmi://localhost/GCDService"); int gcd = service.calculateGCD(num1, num2); resultField.setText(String.valueOf(gcd));

} catch (Exception ex) { ex.printStackTrace(); resultField.setText("Error: Invalid input");

}

}

});

add(label1); add(numberField1); add(label2); add(numberField2); add(calculateButton); add(resultLabel); add(resultField);

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> {

GCDClient client = new GCDClient(); client.setVisible(true);

});

}

}

Interface:

package gui9;

import java.rmi.Remote;

import java.rmi.RemoteException;

}

Impl:



public interface GCDService extends Remote {

int calculateGCD(int a, int b) throws RemoteException;

package gui9;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class GCDServiceImpl extends UnicastRemoteObject implements GCDService {

protected GCDServiceImpl() throws RemoteException

{ super();

}

@Override

public int calculateGCD(int a, int b) throws RemoteException { while (b != 0) {

int temp = b; b = a % b;

a = temp;

}

Server:

}

return Math.abs(a);

}

package gui9;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class GCDServer {

public static void main(String[] args) { try {

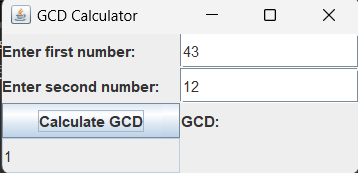
LocateRegistry.createRegistry(7819); // Start RMI registry GCDService service = new GCDServiceImpl(); Naming.rebind("rmi://localhost/GCDService", service); System.out.println("GCD Service is running...");

} catch (Exception e) { e.printStackTrace();

}

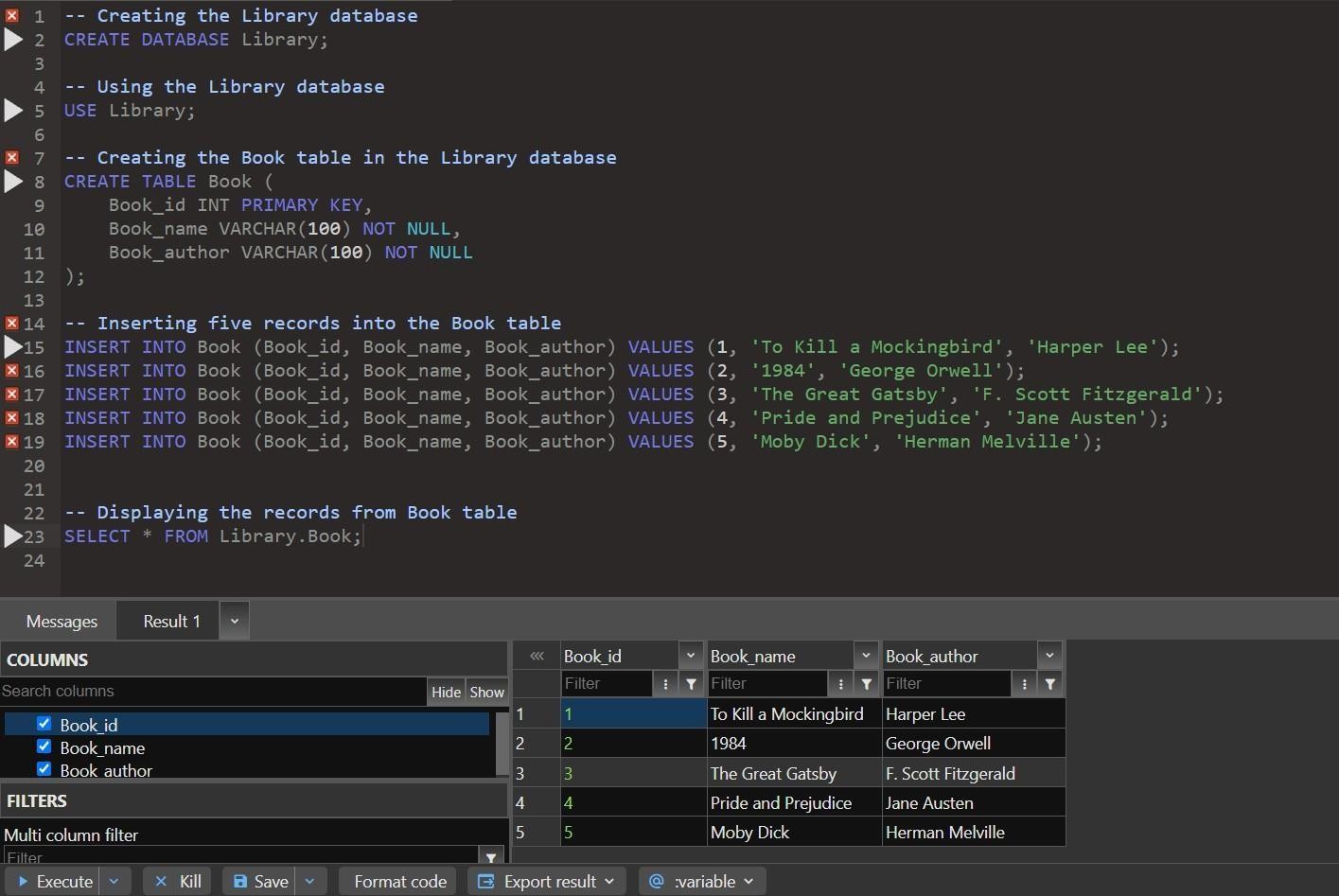
}

}



Practical 5:

Database setup:



Hello.java:

package library;

import java.rmi.Remote; import java.rmi.RemoteException; import java.util.List;

public interface Hello extends Remote {

List<String> getBookInfo() throws RemoteException;

}

Client:

package library;

import java.rmi.Naming; import java.util.List;



public class Client {

public static void main(String[] args) { try {

Hello hello = (Hello) Naming.lookup("rmi://localhost/Hello"); List<String> books = hello.getBookInfo();

for (String book : books) { System.out.println(book)

;

}

} catch (Exception e) { e.printStackTrace();

}

}

}

Library:

package library;

public class Library {

public static void main(String[] args) {

// Here you can choose to run the server or client based on input arguments or conditions. if (args.length > 0 && args[0].equals("server")) {

try {

Server.main(args); // Start the RMI server

} catch (Exception e) {

e.printStackTrace();



}

} else { try {

Client.main(args); // Start the RMI client

} catch (Exception e) { e.printStackTrace();

}

}

}

}

Impl:

package library;

import java.rmi.RemoteException; import java.rmi.server.UnicastRemoteObject; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.ResultSet; import java.sql.Statement; import java.util.ArrayList; import java.util.List;

public class ImplExample extends UnicastRemoteObject implements Hello

{ private static final long serialVersionUID = 1L;

protected ImplExample() throws RemoteException {

super();



}

public List<String> getBookInfo() throws RemoteException { List<String> bookList = new ArrayList<>();

String url = "jdbc:mysql://localhost:3306/library"; // Adjust the URL as needed String user = "root"; // Replace with your DB username String password = "root"; // Replace with your DB password

try {

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

String query = "SELECT BookName, BookAuthor FROM book"; ResultSet rs = stmt.executeQuery(query);

while (rs.next()) {

String bookInfo = "Name: " + rs.getString("BookName") + ", Author: " + rs.getString("BookAuthor");

bookList.add(bookInfo);

}

rs.close();

stmt.close();

conn.close();

} catch (Exception e) { e.printStackTrace();

}

return bookList;

}

}



Server:

package library;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class Server {

public static void main(String[] args) { try {

LocateRegistry.createRegistry(1099); // Start RMI registry on port 1099 ImplExample obj = new ImplExample(); Naming.rebind("rmi://localhost/Hello", obj); System.out.println("Server is ready.");

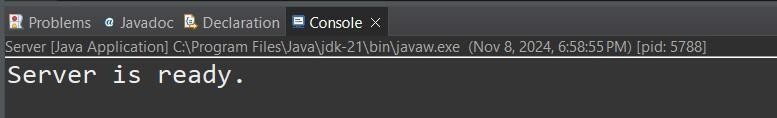
} catch (Exception e) { e.printStackTrace();

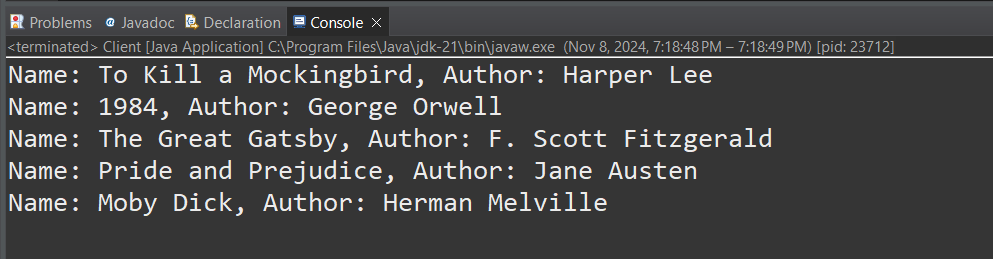
}

}

}

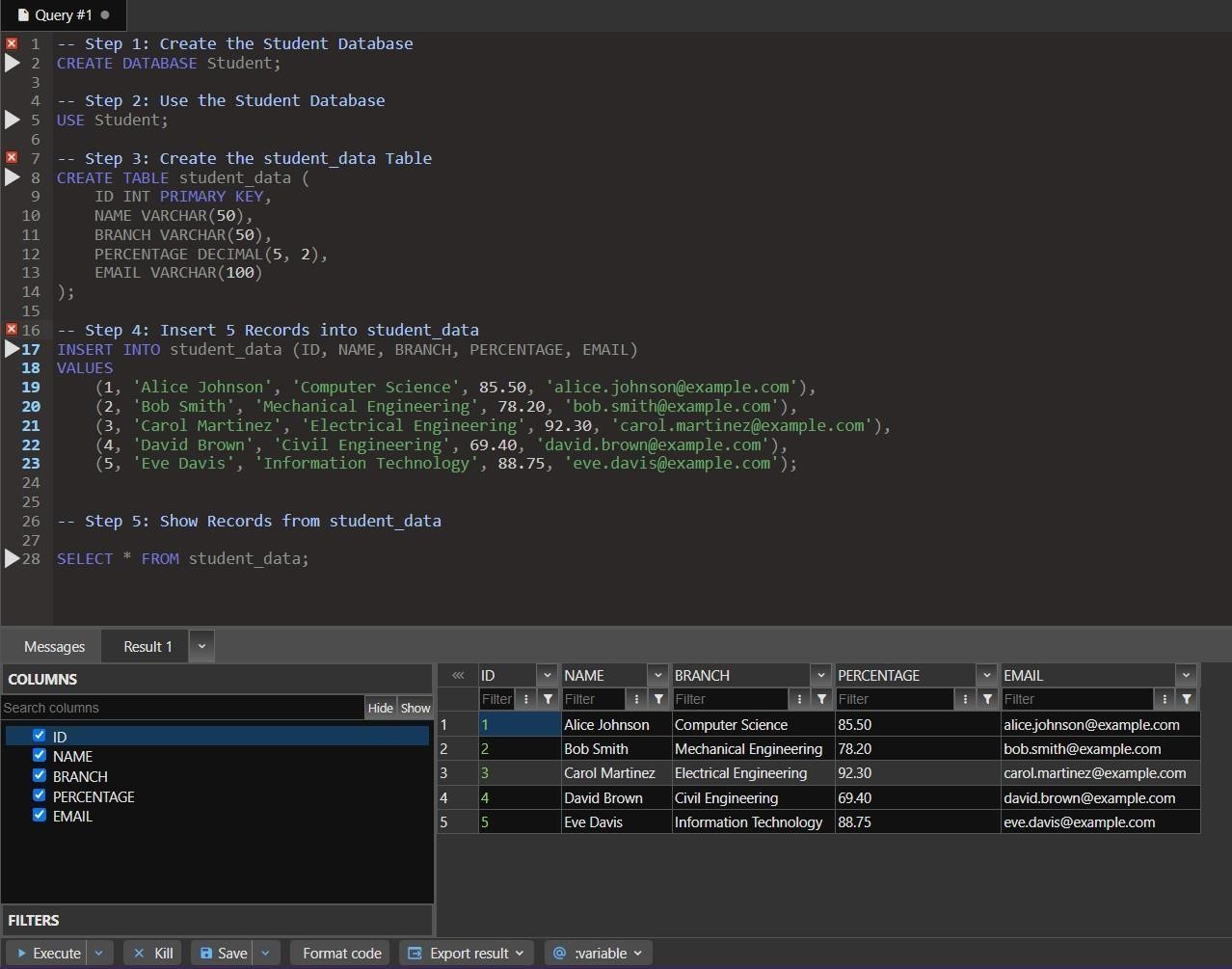
o/p:





Practical 6:

Database setup:



Student(interface):

package student;

import java.rmi.Remote; import java.rmi.RemoteException; import java.util.List;

public interface Student extends Remote { List<StudentData> getStudents() throws RemoteException;

}

Client:

package student;

import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.util.List;

public class Client {

public static void main(String[] args) { try {

Registry registry = LocateRegistry.getRegistry("localhost", 5666); Student stub = (Student) registry.lookup("StudentService"); List<StudentData> students = stub.getStudents();

for (StudentData student : students) { System.out.println("ID: " + student.getId() +

", Name: " + student.getName()

+ ", Branch: " + student.getBranch() + ", Percentage: " +

student.getPercentage() + ", Email: " + student.getEmail());

}

} catch (Exception e) {

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

}

}

}

Server:

package student;



import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

public class Server {

public static void main(String[] args) { try {

ImplExample obj = new ImplExample(); Registry registry = LocateRegistry.createRegistry(5666); registry.bind("StudentService", obj); System.out.println("Server is ready.");

} catch (Exception e) {

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

}

}

}

Impl:

package student;

import java.rmi.RemoteException; import java.rmi.server.UnicastRemoteObject; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.ResultSet; import java.sql.SQLException;

import java.sql.Statement; import java.util.ArrayList; import java.util.List;



@SuppressWarnings("serial")

public class ImplExample extends UnicastRemoteObject implements Student {

protected ImplExample() throws RemoteException { super();

}

@Override

public List<StudentData> getStudents() throws RemoteException { List<StudentData> students = new ArrayList<>();

String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver"; String DB\_URL = "jdbc:mysql://localhost:3306/student"; String USER = "root"; // replace with your username String PASS = "root"; // replace with your password

Connection conn = null; Statement stmt = null;

try {

Class.forName(JDBC\_DRIVER);

conn = DriverManager.getConnection(DB\_URL, USER, PASS); stmt = conn.createStatement();

String sql = "SELECT ID, NAME, BRANCH, PERCENTAGE, EMAIL FROM

student\_data";

ResultSet rs = stmt.executeQuery(sql);



while (rs.next()) {

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

String name = rs.getString("NAME"); String branch = rs.getString("BRANCH");

double percentage = rs.getDouble("PERCENTAGE"); String email = rs.getString("EMAIL");

students.add(new StudentData(id, name, branch, percentage, email));

}

rs.close();

} catch (SQLException | ClassNotFoundException e) { e.printStackTrace();

} finally { try {

if (stmt != null) stmt.close(); if (conn != null) conn.close();

} catch (SQLException se)

{ se.printStackTrace();

}

}

return students;

}

}

StudentData:

package student;



import java.io.Serializable;

public class StudentData implements Serializable {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L; private int id;

private String name; private String branch; private double percentage; private String email;

public StudentData(int id, String name, String branch, double percentage, String email) { this.id = id;

this.name = name; this.branch = branch; this.percentage = percentage; this.email = email;

}

// Getters

public int getId() { return id; }

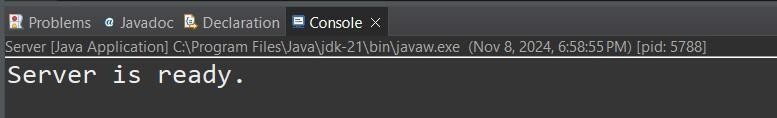
public String getName() { return name; } public String getBranch() { return branch; }

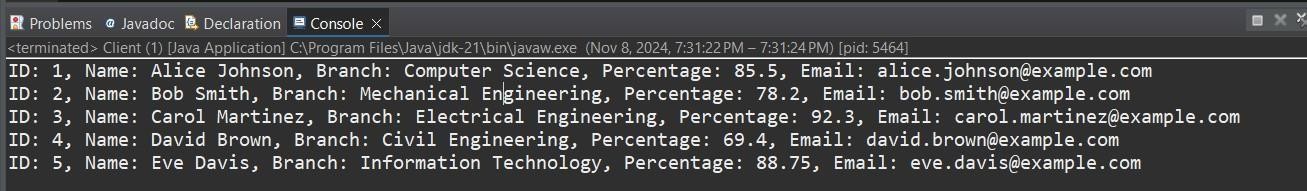
public double getPercentage() { return percentage; }

public String getEmail() { return email; }

}

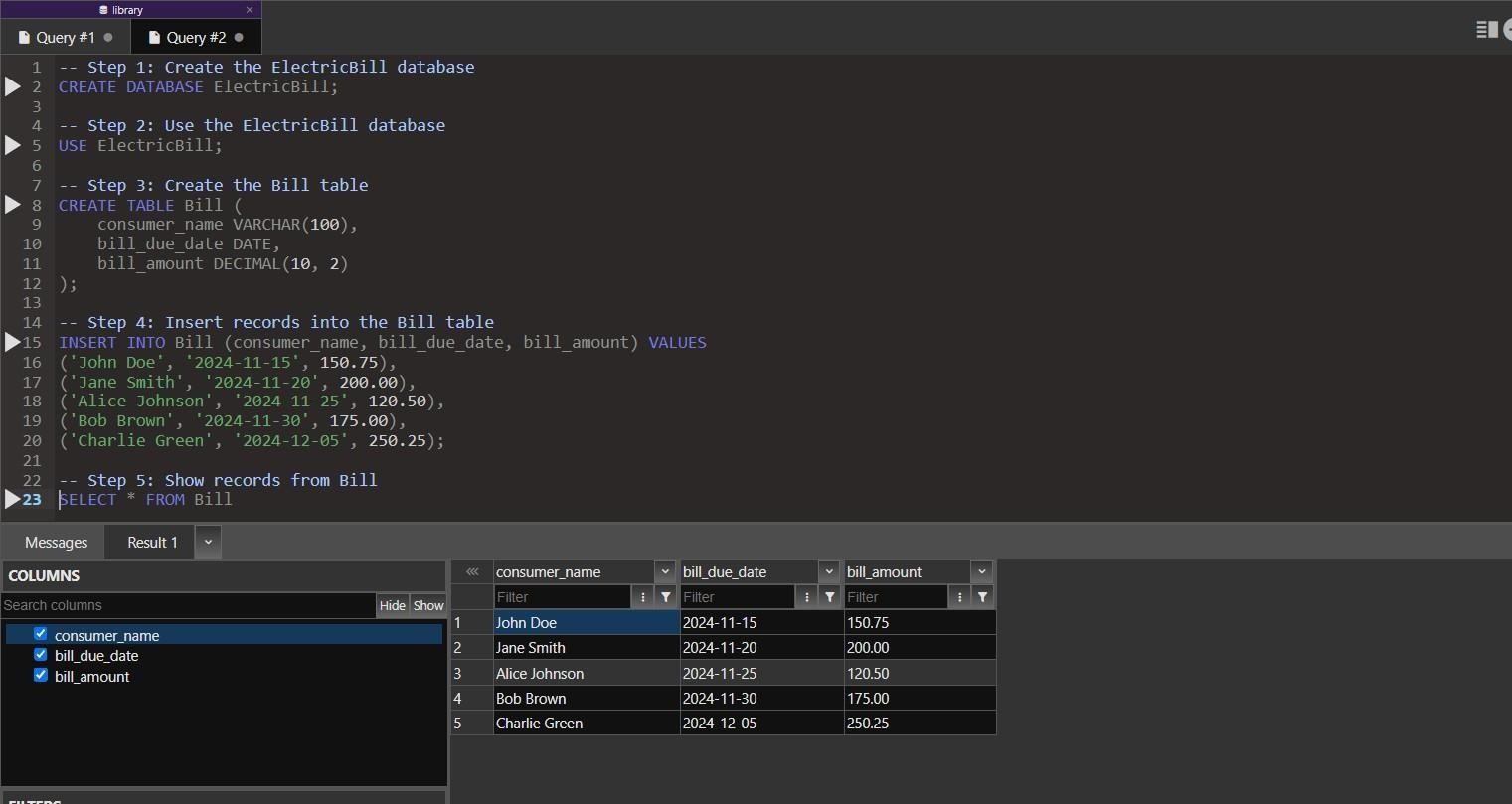
o/p:





Practical 7:

Database setup:



Bill:

package bill;

import java.io.Serializable;

@SuppressWarnings("serial")

public class Bill implements Serializable { private String consumerName;

private String billDueDate; private double billAmount;

public Bill(String consumerName, String billDueDate, double billAmount) { this.consumerName = consumerName;

this.billDueDate = billDueDate; this.billAmount



= billAmount;

}

// Getters

public String getConsumerName() { return consumerName; } public String getBillDueDate() { return billDueDate; }

public double getBillAmount() { return billAmount; }

}

Electricbillservice(interface):

package bill;

import java.rmi.Remote; import java.rmi.RemoteException; import java.util.List;

public interface ElectricBillService extends Remote { List<Bill> getBills() throws RemoteException;

}

Implexample:

package bill;

import java.rmi.RemoteException; import java.rmi.server.UnicastRemoteObject; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.ResultSet;

import java.sql.SQLException; import java.sql.Statement; import java.util.ArrayList; import java.util.List;



@SuppressWarnings("serial")

public class ImplExample extends UnicastRemoteObject implements ElectricBillService {

protected ImplExample() throws RemoteException { super();

}

@Override

public List<Bill> getBills() throws RemoteException

{ List<Bill> bills = new ArrayList<>();

String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver"; String DB\_URL = "jdbc:mysql://localhost:3306/electricbill"; String USER

= "root"; // replace with your username

String PASS = "root"; // replace with your password

Connection conn = null; Statement stmt = null;

try {

Class.forName(JDBC\_DRIVER);

conn = DriverManager.getConnection(DB\_URL, USER, PASS); stmt = conn.createStatement();

String sql = "SELECT consumer\_name, bill\_due\_date, bill\_amount FROM Bill"; ResultSet rs = stmt.executeQuery(sql);



while (rs.next()) {

String consumerName = rs.getString("consumer\_name"); String billDueDate

= rs.getString("bill\_due\_date"); double billAmount = rs.getDouble("bill\_amount");

bills.add(new Bill(consumerName, billDueDate, billAmount));

}

rs.close();

} catch (SQLException | ClassNotFoundException e) { e.printStackTrace();

} finally { try {

if (stmt != null) stmt.close(); if (conn != null) conn.close();

} catch (SQLException se)

{ se.printStackTrace();

}

}

return bills;

}

}

Client:

package bill;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry; import java.util.List;



public class Client {

public static void main(String[] args) { try {

Registry registry = LocateRegistry.getRegistry("localhost", 5266); ElectricBillService stub = (ElectricBillService) registry.lookup("ElectricBillService"); List<Bill> bills = stub.getBills();

for (Bill bill : bills) { System.out.println("Consumer Name: " +

bill.getConsumerName() + ", Due Date: " + bill.getBillDueDate() +

", Amount: $" + bill.getBillAmount());

}

} catch (Exception e) {

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

}

}

}

Server:

package bill;

import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

public class Server {

public static void main(String[] args) { try {

ImplExample obj = new ImplExample(); Registry registry = LocateRegistry.createRegistry(5266); registry.bind("ElectricBillService", obj); System.out.println("Server is ready.");

} catch (Exception e) {

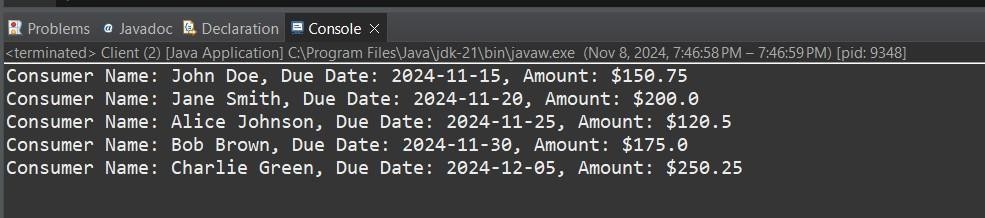
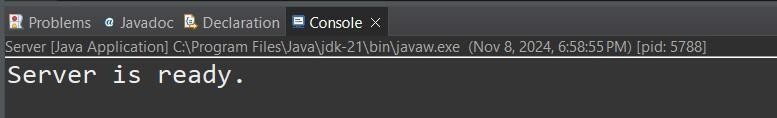
System.err.println("Server exception: " + e.toString()); e.printStackTrace();

}

}

}

o/p:



Practical 8:



Token:

package token;

public class Token {

// Token can have additional properties if needed

}

Process:

package token;

import java.util.Random;

public class Process extends Thread { private int id; private TokenRingManager manager; private boolean hasToken = false;

public Process(int id, TokenRingManager manager)

{ this.id = id; this.manager = manager;

}

public void run() { while (true) {

if (hasToken) { enterCriticalSection();

hasToken = false; // Pass token after critical section manager.passToken(id);



} else {

// Wait for the token try {

Thread.sleep(new Random().nextInt(1000)); // Simulate waiting time

} catch (InterruptedException e) { e.printStackTrace();

}

}

}

}

public void receiveToken() { hasToken = true;

System.out.println("Process " + id + " received the token.");

}

private void enterCriticalSection() { System.out.println("Process " + id + " is entering critical section."); try {

Thread.sleep(500); // Simulate work in critical section

} catch (InterruptedException e) { e.printStackTrace();

}

System.out.println("Process " + id + " is leaving critical section.");

}

public long getId() { return id;

}

}

TokenRingManager:



package token;

public class TokenRingManager

{ private Process[] processes; private int currentIndex = 0;

public TokenRingManager(int numberOfProcesses) { processes = new Process[numberOfProcesses]; for (int i = 0; i

< numberOfProcesses; i++) { processes[i] = new Process(i, this);

}

processes[0].receiveToken(); // Start with first process having the token

}

public void startProcesses() {

for (Process process : processes) { process.start();

}

}

public synchronized void passToken(int fromId) {

currentIndex = (fromId + 1) % processes.length; // Pass to next process processes[currentIndex].receiveToken();

}

}



Main:

package token; public class Main {

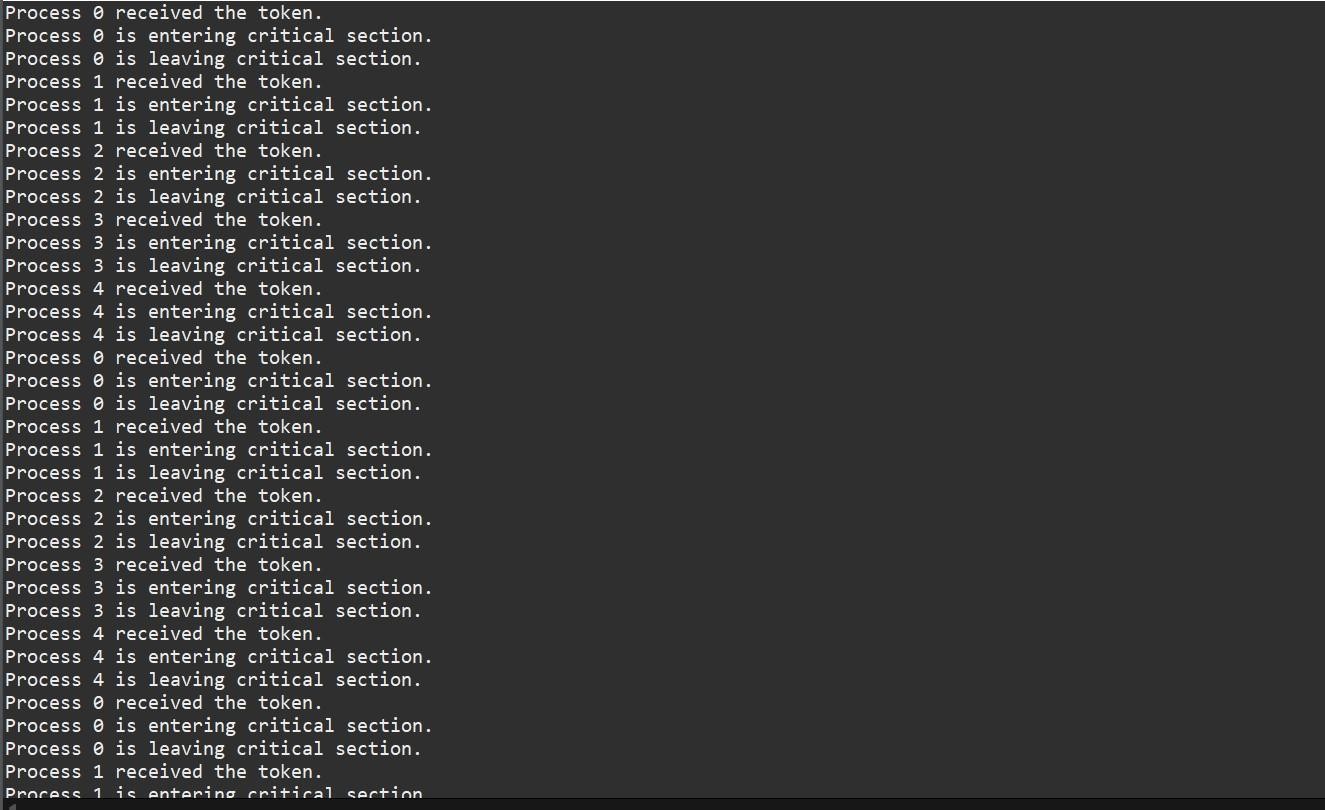
public static void main(String[] args) {

int numberOfProcesses = 5; // You can change this to add more processes TokenRingManager manager = new TokenRingManager(numberOfProcesses); manager.startProcesses();

}

}

o/p:



Practical 9:



Loginformdemo:

package login;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import javax.swing.JButton;

import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JPanel; import javax.swing.JTextField; import javax.swing.JOptionPane;

@SuppressWarnings("serial")

class CreateLoginForm extends JFrame implements ActionListener { JButton loginButton;

JPanel newPanel;

JLabel userLabel, passLabel;

final JTextField textField1, textField2;

// Constructor CreateLoginForm(

) {

// Set up the frame this.setTitle("Login Form"); this.setSize(300, 150);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setLocationRelativeTo(null);



// Initialize components userLabel = new

JLabel("Username:"); textField1 = new JTextField(15); passLabel = new JLabel("Password:"); textField2 = new JTextField(15);

loginButton = new JButton("Login"); loginButton.addActionListener(this);

// Set up the panel and layout newPanel = new JPanel(); newPanel.add(userLabel); newPanel.add(textField1); newPanel.add(passLabel); newPanel.add(textField2); newPanel.add(loginButton);

this.add(newPanel);

}

// Action listener for button @Override

public void actionPerformed(ActionEvent e) { String username = textField1.getText(); String password = textField2.getText();

// Simple validation



if (username.equals("admin") && password.equals("password")) {

// Create and display the welcome page new WelcomePage(username); this.dispose(); // Close the login form

} else {

JOptionPane.showMessageDialog(this, "Invalid Username or Password", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

@SuppressWarnings("serial")

class WelcomePage extends JFrame { WelcomePage(String username) {

// Set up the frame this.setTitle("Welcome"); this.setSize(300, 100);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); this.setLocationRelativeTo(null);

// Create a welcome message

JLabel welcomeLabel = new JLabel("Welcome, " + username); JPanel welcomePanel = new JPanel(); welcomePanel.add(welcomeLabel);

this.add(welcomePanel); this.setVisible(true);

}

}

public class LoginFormDemo {

public static void main(String[] args) {

CreateLoginForm loginForm = new CreateLoginForm(); loginForm.setVisible(true);

}

}

o/p:

