**Group Member 1: Aakanksha Dhawale**

**Group Member 2: Sakshi Gaikwad**

**Group Member 3: Chirag Girdhar**

**Group Member 4: Shubham Golwal**

**Aim** : To implement a Client Server based program using RMI.

**Objective** :

* To learn about remote process communication using RMI.
* To implement a client server based program for a Banking System using RMI in Java.

**Problem Statement:** Implement a Banking System using client server communication with the help of RMI in Java.

**Code:**

**Server.java**

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

*// unicast remote object is a remote object that can only be accessed by one client at a time*

public class Server extends UnicastRemoteObject implements checkBal {

    public Server() throws RemoteException {

*super*();

    }

    static ArrayList<Account> accounts = *new* ArrayList<Account>();

    public double checkBalance(String acctNo, String password) throws RemoteException {

        System.out.println("Request received for account number " + acctNo);

*for* (Account i *:* accounts) {

*if* (i.acc\_no.equals(acctNo) && i.password.equals(password)) {

*return* i.balance;

            }

        }

*// for (int i = 0; i < a.size(); i++) {*

*// double bal = a.get(i).checkBalance(acc\_no, password);*

*// if (bal != -1)*

*// return bal;*

*// }*

*// return -1.0;*

*return* -1.0;

    }

    public static void main(String[] args) {

        String serverName = "bankServer";

*try* {

            Registry reg = LocateRegistry.createRegistry(8000);

            reg.rebind(serverName, *new* Server()); *// rebind is used to bind the object to the registry*

            System.out.println("Server is running...");

            accounts.add(*new* Account("123456", "shubham", 2000.0));

            accounts.add(*new* Account("456789", "password2", 3700.50));

        } *catch* (Exception e) {

            e.printStackTrace();

        }

    }

}

class Account {

    String acc\_no;

    String password;

    double balance;

    Account(String acc\_no, String password, double balance) {

*this*.acc\_no = acc\_no;

*this*.password = password;

*this*.balance = balance;

    }

    public double checkBalance(String acc\_no, String password) {

*if* (*this*.acc\_no.equals(acc\_no) && *this*.password.equals(password))

*return* balance;

*return* -1.0;

    }

    public String toString() {

*return* "Account number: " + acc\_no + "\nPassword: " + password + "\nBalance: " + balance;

    }

}

**//Interface**

**checkBal**

import java.rmi.*\**;

*// Remote interface for our application*

public interface checkBal extends Remote {

    public double checkBalance(String acc\_no, String password) throws RemoteException;

}

**Client.java**

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.Scanner;

public class Client {

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

        String serverName1 = "bankServer";

*try* {

            Scanner sc = *new* Scanner(System.in);

            Registry reg = LocateRegistry.getRegistry("localhost", 8000); *// get the registry*

            checkBal ob = (checkBal) reg.lookup(serverName1); *// lookup is used to find the object*

            System.out.print("\nEnter account number:");

            String acc\_no = sc.nextLine();

            System.out.print("Enter password: ");

            String password = sc.nextLine();

            double bal = ob.checkBalance(acc\_no, password);

*if* (bal == -1) {

                System.out.println("\nInvalid credentials");

*return*;

            } *else* {

                System.out.println("\nBalance: Rs." + bal + "\n");

            }

        } *catch* (Exception e) {

            e.printStackTrace();

        }

    }

}

**Conclusion**: Implemented a banking server, where user information and methods to retrieve that information are stored. The client can remotely call the methods to check his/her balance. Client server communication is achieved using RMI.