Assignment No - 5

Token Ring Algorithm -

Step -

1. Enter the no. of nodes that you want

2 . Take Sender, Receiver, and Data from user.

3 . Inialize token to 0 at starting.

4 . pass the token to the sender .

5 . sender sending the data .

6 . send data to the receiver .

7 . receiver receives the data .

Assignment no - 1

Multithreaded client / server communication.

Files –

1 server interface

Import java .rmi package.

Define interface and extend with Remote interface.

* Define the method that you want and throw RemoteException.

2 server implementation

Import java .rmi package

Import java.rmi.server package

Define class and extend unicastRemoteObject and implements interface (your interface name).

(unicastRemoteObject - until your server is start you call the remote methods)

-implement all the methos

3 server

4 client

import java.util.Scanner;

import javax.xml.crypto.Data;

public class TokenRing {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        // Enter how many nodes you want .

        System.out.print("Enter no of nodes =");

        int n = sc.nextInt();

        for (int i = 0; i < n; i++) {

            System.out.print(i + " ");

        }

        System.out.println("0");

        // for execute code atleast one time

        int ch = 0;

        do {

            System.out.print("Enter Sender - ");

            int sender = sc.nextInt();

            System.out.print("Enter Receiver - ");

            int Receiver = sc.nextInt();

            System.out.print("Enter Data - ");

            int Data = sc.nextInt();

            int token = 0;

            System.out.println("Token Passing ");

            for (int i = token; i < sender; i++) {

                System.out.print(i + "->");

            }

            System.out.println(sender);

            System.out.println("sender" + sender + "sending data" + Data);

            for (int i = sender; i < Receiver; i = (i + 1) % n) {

                System.out.println(Data + "data forwarding ...." + i);

            }

            System.out.println("reicevr recive the data ..");

            token = sender;

            System.out.print("you want to continue Y = 1 , N = 0");

            ch = sc.nextInt();

        } while (ch == 1);

    }

}

import java.rmi.\*;

import java.rmi.registry.Registry;

public class Server {

    public static void main(String[] args) {

        try {

            ServerImplemenation si = new ServerImplemenation();

            // register object in rmi Registry

            Naming.rebind("Server", si);

        } catch (Exception e) {

            System.out.println(e.getMessage());

        }

    }

}

import java.rmi.\*;

import java.util.Scanner;

public class Client {

       pu

    ic static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        // where our rmi registry run

        try {

            String serverUrl = "rmi://localhost/Server";

            // we check particular object is present or not

            ServerInterface serinf = ( ServerInterface ) Naming.lookup(serverUrl);

            System.out.println("Emter num1- ");

            double num1 = sc.nextDouble();

            System.out.println("Emter num2- ");

            double num2 = sc.nextDouble();

           System.out.println(serinf.add(num1, num2)) ;

           System.out.println (serinf.sub(num1, num2));

           System.out.println(serinf.mul(num1, num2));

           System.out.println (serinf.div(num1, num2));

        } catch (Exception e) {

            System.out.println(e);

        }

    }

}

import java.rmi.\*;

import java.util.Scanner;

public class Client {

       pu

    ic static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        // where our rmi registry run

        try {

            String serverUrl = "rmi://localhost/Server";

            // we check particular object is present or not

            ServerInterface serinf = ( ServerInterface ) Naming.lookup(serverUrl);

            System.out.println("Emter num1- ");

            double num1 = sc.nextDouble();

            System.out.println("Emter num2- ");

            double num2 = sc.nextDouble();

           System.out.println(serinf.add(num1, num2)) ;

           System.out.println (serinf.sub(num1, num2));

           System.out.println(serinf.mul(num1, num2));

           System.out.println (serinf.div(num1, num2));

        } catch (Exception e) {

            System.out.println(e);

        }

    }

}

import java.rmi.\*;

public interface ServerInterface extends Remote // it is interface in java

{

    public double add(double num1, double num2) throws RemoteException;

    public double sub(double num1, double num2) throws RemoteException;

    public double mul(double num1, double num2) throws RemoteException;

    public double div(double num1, double num2) throws RemoteException;

}

import java.util.\*;

public class BerkeleyAlgorithm {

    private int numProcesses;

    private int[] processClocks;

    public BerkeleyAlgorithm(int numProcesses) {

        this.numProcesses = numProcesses;

        this.processClocks = new int[numProcesses];

        Arrays.fill(processClocks, 0);

    }

    public void synchronizeClocks() {

        int[] timeOffsets = new int[numProcesses];

        // Calculate the average clock time

        int sum = 0;

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

            sum += processClocks[i];

        }

        int averageTime = sum / numProcesses;

        // Calculate the time offsets for each process

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

            timeOffsets[i] = averageTime - processClocks[i];

        }

        // Adjust the clocks of each process based on the time offsets

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

            processClocks[i] += timeOffsets[i];

        }

    }

    public void displayClocks() {

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

            System.out.println("Process " + (i + 1) + " clock: " + processClocks[i]);

        }

    }

    public void setClock(int processId, int time) {

        if (processId >= 1 && processId <= numProcesses) {

            processClocks[processId - 1] = time;

        } else {

            System.out.println("Invalid process ID.");

        }

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("How many processes you want -");

        int numProcesses = scanner.nextInt();

        BerkeleyAlgorithm berkeley = new BerkeleyAlgorithm(numProcesses);

        // Set initial clock times for each process

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

            System.out.print("Enter the clock time for process " + (i + 1) + ": ");

            int time = scanner.nextInt();

            berkeley.setClock(i + 1, time);

        }

        // Synchronize the clocks

        berkeley.synchronizeClocks();

        // Display the synchronized clocks

        berkeley.displayClocks();

    }

}

//============================================================================

//

//  Name        : Bully.java

//  Author      : Sachin Vijaykumar Kunte

//  Copyright   : Copyright ©2019 by Sachin Kunte All rights reserved.

//  Description : A Java program for a Bully algorithem

//  contact     : kuntesv@gmail.com

//

//============================================================================

import java.io.InputStream;

import java.io.PrintStream;

import java.util.Scanner;

public class Bully {

    static boolean[] state = new boolean[5];

    int coordinator;

    public static void up(int up) {

        if (state[up - 1]) {

            System.out.println("process" + up + "is already up");

        } else {

            int i;

            Bully.state[up - 1] = true;

            System.out.println("process " + up + "held election");

            for (i = up; i < 5; ++i) {

                System.out.println("election message sent from process" + up + "to process" + (i + 1));

            }

            // for (i = up + 1; i <= 5; ++i) {

            // if (!state[i - 1])

            // continue;

            // System.out.println("alive message send from process" + i + "to process" +

            // up);

            // break;

            // }

        }

    }

    public static void down(int down) {

        if (!state[down - 1]) {

            System.out.println("process " + down + "is already dowm.");

        } else {

            Bully.state[down - 1] = false;

        }

    }

    public static void mess(int mess) {

        if (state[mess - 1]) {

            if (state[4]) {

                System.out.println("0K");

            } else if (!state[4]) {

                int i;

                System.out.println("process" + mess + "election");

                for (i = mess; i < 5; ++i) {

                    System.out.println("election send from process" + mess + "to process " + (i + 1));

                }

                for (i = 5; i >= mess; --i) {

                    if (!state[i - 1])

                        continue;

                    System.out.println("Coordinator message send from process" + i + "to all");

                    break;

                }

            }

        } else {

            System.out.println("Prccess" + mess + "is down");

        }

    }

    public static void main(String[] args) {

        int choice;

        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 5; ++i) {

            Bully.state[i] = true;

        }

        System.out.println("5 active process are:");

        System.out.println("Process up  = p1 p2 p3 p4 p5");

        System.out.println("Process 5 is coordinator");

        do {

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

            System.out.println("1 up a process.");

            System.out.println("2.down a process");

            System.out.println("3 send a message");

            System.out.println("4.Exit");

            choice = sc.nextInt();

            switch (choice) {

                case 1: {

                    System.out.println("bring proces up");

                    int up = sc.nextInt();

                    if (up == 5) {

                        System.out.println("process 5 is co-ordinator");

                        Bully.state[4] = true;

                        break;

                    }

                    Bully.up(up);

                    break;

                }

                case 2: {

                    System.out.println("bring down any process.");

                    int down = sc.nextInt();

                    Bully.down(down);

                    break;

                }

                case 3: {

                    System.out.println("which process will send message");

                    int mess = sc.nextInt();

                    Bully.mess(mess);

                }

            }

        } while (choice != 4);

    }

}