EXP 4\_1

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class exp4\_1 {

private static final int PORT = 12345;

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

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

ServerSocket serverSocket = new ServerSocket(PORT);

try {

while (true) {

new ClientHandler(serverSocket.accept()).start();

}

} finally {

serverSocket.close();

}

}

private static class ClientHandler extends Thread {

private static Set<ClientHandler> clientHandlers = new HashSet<>();

private Socket socket;

private PrintWriter out;

private BufferedReader in;

private String clientName;

public ClientHandler(Socket socket) {

this.socket = socket;

}

public void run() {

try {

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

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

out.println("Enter your name: ");

clientName = in.readLine();

synchronized (clientHandlers) {

clientHandlers.add(this);

}

out.println("Welcome, " + clientName + "! You can start chatting now.");

String message;

while ((message = in.readLine()) != null) {

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

synchronized (clientHandlers) {

for (ClientHandler handler : clientHandlers) {

if (handler != this) {

handler.out.println(clientName + ": " + message);

}

}

}

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (clientName != null) {

synchronized (clientHandlers) {

clientHandlers.remove(this);

}

}

if (out != null) {

out.close();

}

if (socket != null) {

socket.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

EXP 4\_2

import java.io.\*;

import java.net.\*;

public class exp4\_2 {

private static final String SERVER\_ADDRESS = "localhost";

private static final int SERVER\_PORT = 12345;

private static PrintWriter out;

private static BufferedReader in;

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

Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);

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

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

BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Enter your name: ");

String name = userInput.readLine();

out.println(name);

Thread readThread = new Thread(new ReadMessages(in));

readThread.start();

String message;

while ((message = userInput.readLine()) != null) {

out.println(message);

}

socket.close();

}

private static class ReadMessages implements Runnable {

private BufferedReader in;

public ReadMessages(BufferedReader in) {

this.in = in;

}

@Override

public void run() {

try {

String message;

while ((message = in.readLine()) != null) {

System.out.println(message);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

EXP 5

import java.util.\*;

public class exp5 {

static int max1(int a, int b) {

return Math.max(a, b);

}

static void display(int e1, int e2, int[] p1, int[] p2) {

System.out.println("\nThe time stamps of events in P1:");

for (int i = 0; i < e1; i++) {

System.out.print(p1[i] + " ");

}

System.out.println("\nThe time stamps of events in P2:");

for (int i = 0; i < e2; i++) {

System.out.print(p2[i] + " ");

}

}

static void lamportLogicalClock(int e1, int e2, int[][] m) {

int[] p1 = new int[e1];

int[] p2 = new int[e2];

// Initialize timestamps

for (int i = 0; i < e1; i++) {

p1[i] = i + 1;

}

for (int i = 0; i < e2; i++) {

p2[i] = i + 1;

}

// Display matrix

System.out.print("\n\t");

for (int i = 0; i < e2; i++) {

System.out.print("e2" + (i + 1) + "\t");

}

for (int i = 0; i < e1; i++) {

System.out.print("\ne1" + (i + 1) + "\t");

for (int j = 0; j < e2; j++) {

System.out.print(m[i][j] + "\t");

}

}

// Update timestamps based on message matrix

for (int i = 0; i < e1; i++) {

for (int j = 0; j < e2; j++) {

if (m[i][j] == 1) {

p2[j] = max1(p2[j], p1[i] + 1);

for (int k = j + 1; k < e2; k++) {

p2[k] = p2[k - 1] + 1;

}

}

if (m[i][j] == -1) {

p1[i] = max1(p1[i], p2[j] + 1);

for (int k = i + 1; k < e1; k++) {

p1[k] = p1[k - 1] + 1;

}

}

}

}

// Display final timestamps

display(e1, e2, p1, p2);

}

public static void main(String[] args) {

int e1 = 5, e2 = 3;

int[][] m = new int[5][3];

System.out.println("DC Experiment No. 05\nName: Saumya Poojari");

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

// Initialize message matrix

m[0][0] = 0; m[0][1] = 0; m[0][2] = 0;

m[1][0] = 0; m[1][1] = 0; m[1][2] = 1;

m[2][0] = 0; m[2][1] = 0; m[2][2] = 0;

m[3][0] = 0; m[3][1] = 0; m[3][2] = 0;

m[4][0] = 0; m[4][1] = -1; m[4][2] = 0;

// Execute Lamport clock algorithm

lamportLogicalClock(e1, e2, m);

}

}

EXP 6

import java.util.ArrayList;

import java.util.Random;

import java.util.Scanner;

public class exp6 {

public static void main(String[] args){

System.out.println("\n\*\*\* ELECTION ALGORITHM: Bully Algorithm \*\*\*\n");

System.out.println("DC Experiment No. 06\nName: Saumya Poojari");

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

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of Processes: ");

int n = sc.nextInt();

ArrayList<Boolean> process = new ArrayList<>();

int prev\_req = -1;

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

process.add(new Random().nextBoolean());

}

process.set(n-1, false);

System.out.println("\nCoordinator Process: "+n+"\nCoordinator Process Now Dead: " + n);

int req = new Random().nextInt(n-1);

System.out.println(process);

System.out.println(req+1);

while (req < n-1 && prev\_req != req){

System.out.println("\n\nRequesting Process: "+(req+1));

for(int j = req+1; j<n; j++){

System.out.println("\n\tProcess["+(req+1)+"]====Election====>Process["+(j+1)+"]");

}

System.out.println();

prev\_req = req;

req = BullyAlgo(req, n, process);

}

System.out.println("\n\nElected Coordinator Process: "+(req+1)+"\nSending Message to All other Process...");

for(int k=0; k<req; k++){

System.out.println("\n\tProcess["+(req+1)+"]====Coordinator====>Process["+(k+1)+"]");

}

System.out.println("\nAll Messages Sent!!!");

}

public static int BullyAlgo(int req, int n, ArrayList<Boolean> processes){

for(int m=req+1; m<n; m++){

if(processes.get(m)){

System.out.println("\n\tReply from Process["+(m+1)+"]: OK");

req=m;

break;

}

}

return req;

}

}

EXP 7

import java.util.Scanner;

public class exp7 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] process\_time = new int[10];

System.out.println("\*\*\* Ricart Agrawala Algorithm \*\*\*");

System.out.println("DC Experiment No. 07\nName: Saumya Poojari");

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

System.out.print("Enter the number of processes: ");

int n = Integer.parseInt(scanner.nextLine());

System.out.println("Now enter their timestamps...");

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

System.out.print("Enter the timestamp for Process [" + i + "]: ");

process\_time[i] = Integer.parseInt(scanner.nextLine());

}

System.out.print("Enter 2 process who wants a shared resource: ");

String[] processIds = scanner.nextLine().split(" ");

int p1 = Integer.parseInt(processIds[0]);

int p2 = Integer.parseInt(processIds[1]);

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

System.out.println("Process [" + p1 + "] sends timestamp " + process\_time[p1] + " to Process [" + i +"]" );

}

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

System.out.println("Process [" + p2 + "] sends timestamp " + process\_time[p2] + " to Process [" + i +"]" );

}

int p = (process\_time[p1] < process\_time[p2]) ? p1 : p2;

int t = Math.min(process\_time[p1], process\_time[p2]);

System.out.println("Process [" + p + "] has the lowest timestamp = " + t);

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

if (i==p) continue;

else System.out.println("Process [" + i + "] sent OK! message to Process [" + p +"]" );

}

System.out.println("Hence Process [" + p + "] is accessing the shared resource, once it is done using it,");

System.out.println("Process [" + (p1 == p? p2:p1) + "] can use it");

scanner.close();

}

}

EXP 8

import java.util.Scanner;

public class exp8 {

public static void main(String[] args) {

System.out.println("\n\*\*\* LOAD BALANCING \*\*\*");

System.out.println("DC Experiment No. 08\nName: Saumya Poojari");

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

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of Servers: ");

int numServers = sc.nextInt();

System.out.print("Enter the number of Processes: ");

int numProcesses = sc.nextInt();

while (true) {

printServerLoad(numServers, numProcesses);

displayMenu();

int choice = sc.nextInt();

int temp;

switch (choice) {

case 1:

System.out.print("Enter number of servers to be added: ");

temp = sc.nextInt();

numServers += temp;

System.out.println(temp + " servers added successfully!");

break;

case 2:

System.out.print("Enter number of servers to be removed: ");

temp = sc.nextInt();

numServers -= temp;

System.out.println(temp + " servers removed successfully!");

break;

case 3:

System.out.print("Enter number of processes to be added: ");

temp = sc.nextInt();

numProcesses += temp;

System.out.println(temp + " processes added successfully!");

break;

case 4:

System.out.print("Enter number of processes to be removed: ");

temp = sc.nextInt();

numProcesses -= temp;

System.out.println(temp + " processes removed successfully!");

break;

case 5:

System.out.println("Exiting");

sc.close();

return;

default:

break;

}

}

}

static void displayMenu() {

System.out.println("1. Add Server");

System.out.println("2. Remove Server");

System.out.println("3. Add Processes");

System.out.println("4. Remove Processes");

System.out.println("5. Exit");

System.out.print("> ");

}

static void printServerLoad(int numServers, int numProcesses) {

int processesPerServer = numProcesses / numServers;

int extraProcesses = numProcesses % numServers;

int i = 0;

for (i = 0; i< extraProcesses; i++){

System.out.println("Server " + (i + 1) + " has " + (processesPerServer + 1 )+ " processes");

}

for(;i<numServers; i++){

System.out.println("Server " + (i + 1) + " has " + processesPerServer + " processes");

}

}

}