**Assignment No. 1**

**Server file:**

package Assign1;

import java.io.\*;

import java.net.\*;

// Server class

class Server {

public static void main(String[] args)

{

ServerSocket server = null;

try {

// server is listening on port 1234

server = new ServerSocket(1234);

server.setReuseAddress(true);

// running infinite loop for getting

// client request

while (true) {

// socket object to receive incoming client

// requests

Socket client = server.accept();

// Displaying that new client is connected

// to server

System.out.println("New client connected"

+ client.getInetAddress()

.getHostAddress());

// create a new thread object

ClientHandler clientSock

= new ClientHandler(client);

// This thread will handle the client

// separately

new Thread(clientSock).start();

}

}

catch (IOException e) {

e.printStackTrace();

}

finally {

if (server != null) {

try {

server.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

}

}

// ClientHandler class

private static class ClientHandler implements Runnable {

private final Socket clientSocket;

// Constructor

public ClientHandler(Socket socket)

{

this.clientSocket = socket;

}

public void run()

{

PrintWriter out = null;

BufferedReader in = null;

try {

// get the outputstream of client

out = new PrintWriter(

clientSocket.getOutputStream(), true);

// get the inputstream of client

in = new BufferedReader(

new InputStreamReader(

clientSocket.getInputStream()));

String line;

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

// writing the received message from

// client

System.out.printf(

" Sent from the client: %s\n",

line);

out.println(line);

}

}

catch (IOException e) {

e.printStackTrace();

}

finally {

try {

if (out != null) {

out.close();

}

if (in != null) {

in.close();

clientSocket.close();

}

}

catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

**Client File:**

package Assign1;

import java.io.\*;

import java.net.\*;

import java.util.\*;

// Client class

class Client {

// driver code

public static void main(String[] args)

{

// establish a connection by providing host and port

// number

try (Socket socket = new Socket("localhost", 1234)) {

// writing to server

PrintWriter out = new PrintWriter(

socket.getOutputStream(), true);

// reading from server

BufferedReader in

= new BufferedReader(new InputStreamReader(

socket.getInputStream()));

// object of scanner class

Scanner sc = new Scanner(System.in);

String line = null;

while (!"exit".equalsIgnoreCase(line)) {

// reading from user

line = sc.nextLine();

// sending the user input to server

out.println(line);

out.flush();

// displaying server reply

System.out.println("Server replied "

+ in.readLine());

}

// closing the scanner object

sc.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

}

**Assignment No. 4**

**BerkeleyClockSync File:**

**package** assign4;

**import** java.util.ArrayList;

**public** **class** BerkeleyClockSync {

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

// Initialize the system clocks

**int**[] systemClocks = { 10, 12, 13, 11, 14 };

**int** masterClock = 0;

// Print the initial system clocks

System.***out***.print("System clocks: ");

**for** (**int** clock : systemClocks) {

System.***out***.print(clock + " ");

}

System.***out***.println();

// Calculate the average system clock

**int** sum = 0;

**for** (**int** clock : systemClocks) {

sum += clock;

}

**int** averageClock = sum / systemClocks.length;

// Calculate the time difference for each system clock

ArrayList<Integer> timeDifferences = **new** ArrayList<>();

**for** (**int** clock : systemClocks) {

timeDifferences.add(averageClock - clock);

}

// Calculate the time adjustment for the master clock

**int** timeAdjustment = 0;

**for** (**int** difference : timeDifferences) {

timeAdjustment += difference;

}

timeAdjustment /= timeDifferences.size();

// Update the master clock

masterClock = averageClock - timeAdjustment;

// Print the updated system clocks and master clock

System.***out***.print("Updated system clocks: ");

**for** (**int** clock : systemClocks) {

System.***out***.print((clock - timeAdjustment) + " ");

}

System.***out***.println();

System.***out***.println("Master clock: " + masterClock);

}

}

**Assignment No.5**

**Tokenring File:**

package assign5;

import java.io.\*;

import java.util.\*;

class tokenring {

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

Scanner scan = new Scanner(System.in);

System.out.println("Enter the num of nodes:");

int n = scan.nextInt();

int m = n - 1;

// Decides the number of nodes forming the ring

int token = 0;

int ch = 0, flag = 0;

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

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

}

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

do{

System.out.println("Enter sender:");

int s = scan.nextInt();

System.out.println("Enter receiver:");

int r = scan.nextInt();

System.out.println("Enter Data:");

int a;

a = scan.nextInt();

System.out.print("Token passing:");

for (int i = token, j = token; (i % n) != s; i++, j = (j + 1) % n) {

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

}

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

System.out.println("Sender " + s + " sending data: " + a);

for (int i = s + 1; i != r; i = (i + 1) % n) {

System.out.println("data " + a + " forwarded by " + i);

}

System.out.println("Receiver " + r + " received data: " + a +"\n");

token = s;

do{

try {

if( flag == 1)

System.out.print("Invalid Input!!...");

System.out.print("Do you want to send again?? enter 1 for Yes and 0 for No : ");

ch = scan.nextInt();

if( ch != 1 && ch != 0 )

flag = 1;

else

flag = 0;

} catch (InputMismatchException e){

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

}

}while( ch != 1 && ch != 0 );

}while( ch == 1 );

}

}

**Assignment No.6**

**Bully File:**

package assign6;

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)//4

{

if (state[up - 1])// 0 1 2 3 4

{

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);

}

}

**Ring File:**

**package** assign6;

**import** java.util.Scanner;

**public** **class** Ring

{

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

{

// **TODO** Auto-generated method stub

**int** temp, i, j;

**char** str[] = **new** **char**[10];

Rr proc[] = **new** Rr[10];

// object initialisation

**for** (i = 0; i < proc.length; i++)

proc[i] = **new** Rr();

// scanner used for getting input from console

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

System.***out***.println("Enter the number of process : ");

**int** num = in.nextInt();

// getting input from users

**for** (i = 0; i < num; i++)

{

proc[i].index = i;

System.***out***.println("Enter the id of process : ");

proc[i].id = in.nextInt();

proc[i].state = "active";

proc[i].f = 0;

}

// sorting the processes from on the basis of id

**for** (i = 0; i < num - 1; i++)

{

**for** (j = 0; j < num - 1; j++)

{

**if** (proc[j].id > proc[j + 1].id)

{

temp = proc[j].id;

proc[j].id = proc[j + 1].id;

proc[j + 1].id = temp;

}

}

}

**for** (i = 0; i < num; i++)

{

System.***out***.print(" [" + i + "]" + " " + proc[i].id);

}

**int** init;

**int** ch;

**int** temp1;

**int** temp2;

**int** ch1;

**int** arr[] = **new** **int**[10];

proc[num - 1].state = "inactive";

System.***out***.println("\n process " + proc[num - 1].id + "select as co-ordinator");

**while** (**true**)

{

System.***out***.println("\n 1.election 2.quit ");

ch = in.nextInt();

**for** (i = 0; i < num; i++)

{

proc[i].f = 0;

}

**switch** (ch)

{

**case** 1:

System.***out***.println("\n Enter the Process number who initialsied election : ");

init = in.nextInt();

temp2 = init;

temp1 = init + 1;

i = 0;

**while** (temp2 != temp1)

{

**if** ("active".equals(proc[temp1].state) && proc[temp1].f ==

0)

{

System.***out***.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);

proc[temp1].f = 1;

init = temp1;

arr[i] = proc[temp1].id;

i++;

}

**if** (temp1 == num)

{

temp1 = 0;

}

**else**

{

temp1++;

}

}

System.***out***.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);

arr[i] = proc[temp1].id;

i++;

**int** max = -1;

// finding maximum for co-ordinator selection

**for** (j = 0; j < i; j++)

{

**if** (max < arr[j])

{

max = arr[j];

}

}

// co-ordinator is found then printing on console

System.***out***.println("\n process " + max + "select as co-ordinator");

**for** (i = 0; i < num; i++)

{

**if** (proc[i].id == max)

{

proc[i].state = "inactive";

}

}

**break**;

**case** 2:

System.***out***.println("Program terminated ...");

**return** ;

**default**:

System.***out***.println("\n invalid response \n");

**break**;

}

}

}

}

**class** Rr

{

**public** **int** index; // to store the index of process

**public** **int** id; // to store id/name of process

**public** **int** f;

String state; // indiactes whether active or inactive state of node

}