Ass=5

import java.util.Scanner;

class Tring {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

// Decides the number of nodes forming the ring

int token = 0;

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

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

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

try {

while (true) {

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

int s = sc.nextInt();

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

int r = sc.nextInt();

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

String d = sc.next();

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

//current token not equal to sender, increment i by 1 and j by j+1%n

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: " + d);

// start forwarding from node after sender until it becomes equal to receiver and increment by i+1%n

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

System.out.println("Data " + d + " forwarded by " + i);

}

System.out.println("Receiver " + r + " received data: " + d);

token = s;

}

} catch (Exception e) {

System.out.println("Error occurred: " + e.getMessage());

}

}

}

Ass=4

import java.util.\*;

public class BerkeleyAlgo {

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

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

int numNodes = sc.nextInt();

int[] localClocks = new int[numNodes];

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

System.out.println("Enter the clock values of clock "+i+" : ");

localClocks[i] = sc.nextInt();

}

int master = localClocks[0];

int diff = 0;

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

diff = diff + (localClocks[i] - master);

}

System.out.println("Every clock to be synchronized by : "+(diff/numNodes)+" minutes ");

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

System.out.println("New clock timings are : "+(localClocks[i]+(diff/numNodes)));

}

}

}

Ass=6

Bully=

import java.util.\*;

public class Bully {

int coordinator;

int max\_processes;

boolean processes[];

public Bully(int max) {

max\_processes = max;

processes = new boolean[max\_processes];

coordinator = max;

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

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

processes[i] = true;

System.out.println("P"+ (i+1) + " created");

}

System.out.println("Process P" + coordinator + " is the coordinator");

}

void displayProcesses() {

for(int i = 0; i < max\_processes; i++) {

if(processes[i]) {

System.out.println("P" + (i+1) + " is up");

} else {

System.out.println("P" + (i+1) + " is down");

}

}

System.out.println("Process P" + coordinator + " is the coordinator");

}

void upProcess(int process\_id) {

if(!processes[process\_id - 1]) {

processes[process\_id - 1] = true;

System.out.println("Process " + process\_id + " is now up.");

} else {

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

}

}

void downProcess(int process\_id) {

if(!processes[process\_id - 1]) {

System.out.println("Process " + process\_id + " is already down.");

} else {

processes[process\_id - 1] = false;

System.out.println("Process " + process\_id + " is down.");

}

}

void runElection(int process\_id) {

coordinator = process\_id;

boolean keepGoing = true;

for(int i = process\_id; i < max\_processes && keepGoing; i++) {

System.out.println("Election message sent from process " + process\_id + " to process " + (i+1));

if(processes[i]) {

keepGoing = false;

runElection(i + 1);

}

}

}

public static void main(String args[]) {

Bully bully = null;

int max\_processes = 0, process\_id = 0;

int choice = 0;

Scanner sc = new Scanner(System.in);

while(true) {

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

System.out.println("1. Create processes");

System.out.println("2. Display processes");

System.out.println("3. Up a process");

System.out.println("4. Down a process");

System.out.println("5. Run election algorithm");

System.out.println("6. Exit Program");

System.out.print("Enter your choice:- ");

choice = sc.nextInt();

switch(choice) {

case 1:

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

max\_processes = sc.nextInt();

bully = new Bully(max\_processes);

break;

case 2:

bully.displayProcesses();

break;

case 3:

System.out.print("Enter the process number to up:- ");

process\_id = sc.nextInt();

bully.upProcess(process\_id);

break;

case 4:

System.out.print("Enter the process number to down:- ");

process\_id = sc.nextInt();

bully.downProcess(process\_id);

break;

case 5:

System.out.print("Enter the process number which will perform election:- ");

process\_id = sc.nextInt();

bully.runElection(process\_id);

bully.displayProcesses();

break;

case 6:

System.exit(0);

break;

default:

System.out.println("Error in choice. Please try again.");

break;

}

}

}

}

Ring=

import java.util.\*;

public class Ring {

int max\_processes;

int coordinator;

boolean processes[];

ArrayList<Integer> pid;

public Ring(int max) {

coordinator = max;

max\_processes = max;

pid = new ArrayList<Integer>();

processes = new boolean[max];

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

processes[i] = true;

System.out.println("P" + (i+1) + " created.");

}

System.out.println("P" + (coordinator) + " is the coordinator");

}

void displayProcesses() {

for(int i = 0; i < max\_processes; i++) {

if(processes[i])

System.out.println("P" + (i+1) + " is up.");

else

System.out.println("P" + (i+1) + " is down.");

}

System.out.println("P" + (coordinator) + " is the coordinator");

}

void upProcess(int process\_id) {

if(!processes[process\_id-1]) {

processes[process\_id-1] = true;

System.out.println("Process P" + (process\_id) + " is up.");

} else {

System.out.println("Process P" + (process\_id) + " is already up.");

}

}

void downProcess(int process\_id) {

if(!processes[process\_id-1]) {

System.out.println("Process P" + (process\_id) + " is already down.");

} else {

processes[process\_id-1] = false;

System.out.println("Process P" + (process\_id) + " is down.");

}

}

void displayArrayList(ArrayList<Integer> pid) {

System.out.print("[ ");

for(Integer x : pid) {

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

}

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

}

void initElection(int process\_id) {

if(processes[process\_id-1]) {

pid.add(process\_id);

int temp = process\_id;

System.out.print("Process P" + process\_id + " sending the following list:- ");

displayArrayList(pid);

while(temp != process\_id - 1) {

if(processes[temp]) {

pid.add(temp+1);

System.out.print("Process P" + (temp + 1) + " sending the following list:- ");

displayArrayList(pid);

}

temp = (temp + 1) % max\_processes;

}

coordinator = Collections.max(pid);

System.out.println("Process P" + process\_id + " has declared P" + coordinator + " as the coordinator");

pid.clear();

}

}

public static void main(String args[]) {

Ring ring = null;

int max\_processes = 0, process\_id = 0;

int choice = 0;

Scanner sc = new Scanner(System.in);

while(true) {

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

System.out.println("1. Create processes");

System.out.println("2. Display processes");

System.out.println("3. Up a process");

System.out.println("4. Down a process");

System.out.println("5. Run election algorithm");

System.out.println("6. Exit Program");

System.out.print("Enter your choice:- ");

choice = sc.nextInt();

switch(choice) {

case 1:

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

max\_processes = sc.nextInt();

ring = new Ring(max\_processes);

break;

case 2:

ring.displayProcesses();

break;

case 3:

System.out.print("Enter the process to up:- ");

process\_id = sc.nextInt();

ring.upProcess(process\_id);

break;

case 4:

System.out.print("Enter the process to down:- ");

process\_id = sc.nextInt();

ring.downProcess(process\_id);

break;

case 5:

System.out.print("Enter the process which will initiate election:- ");

process\_id = sc.nextInt();

ring.initElection(process\_id);

break;

case 6:

System.exit(0);

break;

default:

System.out.println("Error in choice. Please try again.");

break;

}

}

}

}

**1st assignment**

Hello=

import java.rmi.\*;

import java.rmi.server.\*;

public class Hello extends UnicastRemoteObject implements HelloInterface

{

private String message;

public Hello (String msg) throws RemoteException

{

message = msg;

}

public String say() throws RemoteException

{

return message;

}

}

HelloClient=

import java.rmi.\*;

import java.io.\*;

public class HelloClient

{

public static void main (String[] argv)

{

try

{

HelloInterface hello = (HelloInterface) Naming.lookup("//priyanka-Inspiron-3593

/Hello");

System.out.println(hello.say());

}

catch (Exception e)

{

System.out.println ("HelloClient exception: " + e);

}

}

}

HelloInterface=

import java.rmi.\*;

public interface HelloInterface extends Remote

{

public String say() throws RemoteException;

}

HelloServer=

import java.io.\*;

import java.rmi.\*;

public class HelloServer

{

public static void main (String[] argv)

{

try

{

Hello robj = new Hello ("Hello, world!");

Naming.rebind ("Hello", robj);

System.out.println ("Hello Server is ready.");

}

catch (Exception e)

{

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

}

}

}