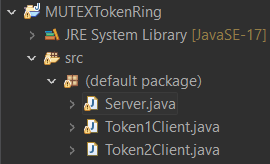
**Practical No. 05**

**Mutual Exclusion**

**Q.1 Write a java program to implement mutual exclusion using Token ring algorithm.**

****

**Code:**

**Server.java**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.io.\*;

class TokenServer{

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

while(true) {

Server sr = new Server();

sr.recPort(8000);

sr.recData();

}

}

}

public class Server {

boolean hasToken = false;

boolean sendData = false;

int recport;

void recPort(int recport) {

this.recport = recport;

}

void recData() throws Exception{

byte bu[] = new byte[256];

DatagramSocket ds;

DatagramPacket dp;

String str;

ds = new DatagramSocket(recport);

dp = new DatagramPacket(bu, bu.length);

ds.receive(dp);

ds.close();

str = new String(dp.getData(), 0, dp.getLength());

System.out.println("The message is " + str);

}

}

**Token1Client.java**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class Token1Client {

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

InetAddress lclhost;

BufferedReader br;

String str = "";

TokenClient12 tkcl, tkser;

boolean hasToken;

boolean setSendData;

while(true) {

lclhost = InetAddress.getLocalHost();

tkcl = new TokenClient12(lclhost);

tkser = new TokenClient12(lclhost);

//tkcl.setSendPort(9001);

tkcl.setSendPort(9004);

tkcl.setRecPort(8002);

lclhost = InetAddress.getLocalHost();

tkser.setSendPort(9000);

if(tkcl.hasToken == true) {

System.out.println("Do you want to Enter the Data -> Yes/No");

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

str = br.readLine();

if(str.equalsIgnoreCase("yes")) {

System.out.println("Ready to Send");

tkser.setSendData = true;

tkser.sendData();

tkser.setSendData = false;

}else if (str.equalsIgnoreCase("no")) {

System.out.println("I'm in else");

tkcl.hasToken = false;

tkcl.sendData();

tkcl.recData();

System.out.println("I'm Leaving");

}

}

else {

System.out.println("Entering Receiving Mode ...");

tkcl.recData();

tkcl.hasToken = true;

}

}

}

}

class TokenClient12{

InetAddress lclhost;

int sendport, recport;

boolean hasToken = true;

boolean setSendData = false;

TokenClient12 tkcl, tkser;

public TokenClient12(InetAddress lclhost) {

// TODO Auto-generated constructor stub

this.lclhost = lclhost;

}

public void setSendPort(int sendport) {

this.sendport = sendport;

}

public void setRecPort(int recport) {

this.recport = recport;

}

void sendData() throws Exception{

BufferedReader br;

String str = "Token";

DatagramSocket ds;

DatagramPacket dp;

if(setSendData == true) {

System.out.println("sending");

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

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

str = "Client One... " + br.readLine();

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

}

ds = new DatagramSocket(sendport);

dp = new DatagramPacket(str.getBytes(), str.length(), lclhost, sendport-1000);

ds.send(dp);

ds.close();

setSendData = false;

hasToken = false;

}

void recData() throws Exception{

String msgStr;

byte buffer[] = new byte[256];

DatagramPacket dp;

DatagramSocket ds;

ds = new DatagramSocket(recport);

dp = new DatagramPacket(buffer, buffer.length);

ds.receive(dp);

ds.close();

msgStr = new String(dp.getData(), 0, dp.getLength());

System.out.println("The data is " + msgStr);

if(msgStr.equals("Token")) {

hasToken = true;

}

}

}

**Token2Client.java**

import java.io.\*;

import java.net.\*;

public class Token2Client {

static boolean setSendData ; static boolean hasToken ;

public static void main(String arg[]) throws Exception

{

InetAddress lclhost; BufferedReader br;

String str1;

TokenClient21 tkcl;

TokenClient21 ser;

while(true)

{

lclhost=InetAddress.getLocalHost();

tkcl = new TokenClient21(lclhost);

tkcl.setRecPort(8004);

tkcl.setSendPort(9002);

lclhost=InetAddress.getLocalHost();

ser = new TokenClient21(lclhost);

ser.setSendPort(9000);

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

if(hasToken == true)

{

System.out.println("Do you want to enter the Data –> YES/NO");

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

str1=br.readLine();

if(str1.equalsIgnoreCase("yes"))

{

System.out.println("ignorecase");

ser.setSendData = true;

ser.sendData();

}

else if(str1.equalsIgnoreCase("no"))

{

tkcl.sendData();

tkcl.hasToken=false; tkcl.sendData(); tkcl.recData();

hasToken=false;

}

}

else

{

System.out.println("entering recieving mode");

tkcl.recData();

hasToken=true;

}

}

}

}

class TokenClient21

{

InetAddress lclhost;

int sendport,recport;

boolean setSendData = false;

boolean hasToken = false;

TokenClient21 tkcl;

TokenClient21 ser;

TokenClient21(InetAddress lclhost)

{

this.lclhost = lclhost;

}

void setSendPort(int sendport)

{

this.sendport = sendport;

}

void setRecPort(int recport)

{

this.recport = recport;

}

void sendData() throws Exception

{

System.out.println("case");

BufferedReader br;

String str="Token";

DatagramSocket ds;

DatagramPacket dp;

if(setSendData == true)

{

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

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

str ="ClientTwo….." + br.readLine();

}

ds = new DatagramSocket(sendport);

dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);

ds.send(dp);

ds.close();

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

setSendData = false; hasToken = false;

}

@SuppressWarnings("resource")

void recData()throws Exception

{

String msgstr;

byte buffer[] = new byte[256];

DatagramSocket ds;

DatagramPacket dp;

ds = new DatagramSocket(recport);

ds = new DatagramSocket(4000);

dp = new DatagramPacket(buffer,buffer.length);

ds.receive(dp);

ds.close();

msgstr = new String(dp.getData(),0,dp.getLength());

System.out.println("The data is "+msgstr);

if(msgstr.equals("Token"))

{

hasToken = true;

}

}

}

**Output:**

