**Practical 7**

**Distributed Operating System**

**Name - Ritesh Parkhi**

**Roll No. – 46**

**Batch – 3**

**AIM:**

Construct a program to implement two phase commit protocol.

**THEORY:**

Distributed two-phase commit reduces the vulnerability of one-phase commit protocols. The steps performed in the two phases are as follows −

**Phase 1: Prepare Phase**

* After each slave has locally completed its transaction, it sends a “DONE” message to the controlling site. When the controlling site has received “DONE” message from all slaves, it sends a “Prepare” message to the slaves.
* The slaves vote on whether they still want to commit or not. If a slave wants to commit, it sends a “Ready” message.
* A slave that does not want to commit sends a “Not Ready” message. This may happen when the slave has conflicting concurrent transactions or there is a timeout.

**Phase 2: Commit/Abort Phase**

* After the controlling site has received “Ready” message from all the slaves −
  + The controlling site sends a “Global Commit” message to the slaves.
  + The slaves apply the transaction and send a “Commit ACK” message to the controlling site.
  + When the controlling site receives “Commit ACK” message from all the slaves, it considers the transaction as committed.
* After the controlling site has received the first “Not Ready” message from any slave −
  + The controlling site sends a “Global Abort” message to the slaves.
  + The slaves abort the transaction and send a “Abort ACK” message to the controlling site.
  + When the controlling site receives “Abort ACK” message from all the slaves, it considers the transaction as aborted.

**PROGRAM:**

TPCServer.java

import java.io.\*;

import java.net.\*;

public class TPCServer

{

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

        {

            BufferedReader br;

            InetAddress lclhost;

            lclhost=InetAddress.getLocalHost();

            Server ser=new Server(lclhost);

            System.out.println("Server in sending mode…..");

*// Sending data to client 1*

            ser.setSendPort(9000);   *//recport=8000*

            ser.setRecPort(8001);   *//sendport=9001*

            System.out.println("Send request data to client1..");

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

        String s=br.readLine();

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

            ser.sendData();

            System.out.println("Waiting for response from client1….");

            ser.recData();

*// Sending data to client 2*

            ser.setSendPort(9002);  *//recport=8002*

            ser.setRecPort(8003);  *//senport=9003*

            System.out.println("Send request data to client2..");

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

        String s1=br.readLine();

            System.out.println("Data is "+s1);

            ser.sendData();

            System.out.println("Waiting for response from client2….");

            ser.recData();

*//Sending the final result to client 1*

            ser.setSendPort(9000);

            ser.sendData();

*//Sending the final result to client 2*

            ser.setSendPort(9002);

            ser.sendData();

        }

}

class Server

{

    InetAddress lclhost;

    int sendPort,recPort;

    int ssend =0;

    int scounter=0;

    Server(InetAddress lclhost)

    {

        this.lclhost=lclhost;

    }

    public void setSendPort(int sendPort)

    {

        this.sendPort=sendPort;

    }

    public void setRecPort(int recPort)

    {

        this.recPort=recPort;

    }

    public void sendData()throws Exception

    {

        DatagramSocket ds;

        DatagramPacket dp;

        String data="";

        if(scounter<2 && ssend<2)

            {

                data="VOTE\_REQUEST";

            }

        if(scounter<2 && ssend>1)

            {

                data="GLOBAL\_ABORT";

                data= data + " TRANSACTION ABORTED";

            }

        if(scounter==2 && ssend>1)

            {

                data="GLOBAL\_COMMIT";

                data= data + " TRANSACTION COMMITED";

            }

        ds=new DatagramSocket(sendPort);

        dp=new DatagramPacket(data.getBytes(),data.length(),lclhost,sendPort-1000);

        ds.send(dp);

        ds.close();

        ssend++;

    }

public void recData()throws Exception

        {

            byte buf[]=new byte[256];

            DatagramPacket dp=null;

            DatagramSocket ds=null;

            String msgStr="";

            try{

            ds=new DatagramSocket(recPort);

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

            ds.receive(dp);

            ds.close();

            }

            catch(Exception e)

            {

                e.printStackTrace();

            }

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

System.out.println("String = "+msgStr);

if(msgStr.equalsIgnoreCase("VOTE\_COMMIT"))

                {

                    scounter++;

                }

        System.out.println("Counter value = "+scounter + "n Send value = "+ssend);

    }

};

TPCClient1.java

import java.io.\*;

import java.net.\*;

public class TPCClient1

{

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

        {

            InetAddress lclhost;

            lclhost=InetAddress.getLocalHost();

            Client clnt=new Client(lclhost);

            clnt.setSendPort(9001);   *//recport=8000*

            clnt.setRecPort(8000);   *//sendport=9001*

            clnt.recData();

            clnt.sendData();

            clnt.recData();

        }

}

class Client

{

InetAddress lclhost;

    int sendPort,recPort;

    Client(InetAddress lclhost)

    {

        this.lclhost=lclhost;

}

    public void setSendPort(int sendPort)

    {

        this.sendPort=sendPort;

    }

    public void setRecPort(int recPort)

    {

        this.recPort=recPort;

    }

    public void sendData()throws Exception

    {

        BufferedReader br;

        DatagramSocket ds;

        DatagramPacket dp;

        String data="";

        System.out.println("Enter the Response 'VOTE\_COMMIT' || 'VOTE\_ABORT' ");

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

        data = br.readLine();

        System.out.println("Data is "+data);

        ds=new DatagramSocket(sendPort);

        dp=new DatagramPacket(data.getBytes(),data.length(),lclhost,sendPort-1000);

        ds.send(dp);

        ds.close();

    }

public void recData()throws Exception

        {

            byte buf[]=new byte[256];

            DatagramPacket dp;

            DatagramSocket ds;

            ds=new DatagramSocket(recPort);

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

            ds.receive(dp);

            ds.close();

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

            System.out.println("Client1 data " +msgStr);

    }

};

TPCClient2.java

import java.io.\*;

import java.net.\*;

public class TPCClient2

{

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

        {

            InetAddress lclhost;

            lclhost=InetAddress.getLocalHost();

            Client client=new Client(lclhost);

*// Sending data to client 2*

            client.setSendPort(9003);  *//recport=8002*

            client.setRecPort(8002);  *//senport=9003*

            client.recData();

            client.sendData();

            client.recData();

    }

}

class Client

{

    InetAddress lclhost;

    int sendPort,recPort;

    Client(InetAddress lclhost)

    {

        this.lclhost=lclhost;

    }

    public void setSendPort(int sendPort)

    {

        this.sendPort=sendPort;

    }

    public void setRecPort(int recPort)

    {

        this.recPort=recPort;

    }

    public void sendData()throws Exception

    {

        BufferedReader br;

        DatagramSocket ds;

        DatagramPacket dp;

        String data="";

        System.out.println("Enter the Response 'VOTE\_COMMIT' || 'VOTE\_ABORT' ");

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

        data = br.readLine();

        System.out.println("Data is "+data);

        ds=new DatagramSocket(sendPort);

        dp=new DatagramPacket(data.getBytes(),data.length(),lclhost,sendPort-1000);

        ds.send(dp);

        ds.close();

    }

public void recData()throws Exception

        {

            byte buf[]=new byte[256];

            DatagramPacket dp;

            DatagramSocket ds;

            ds=new DatagramSocket(recPort);

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

            ds.receive(dp);

            ds.close();

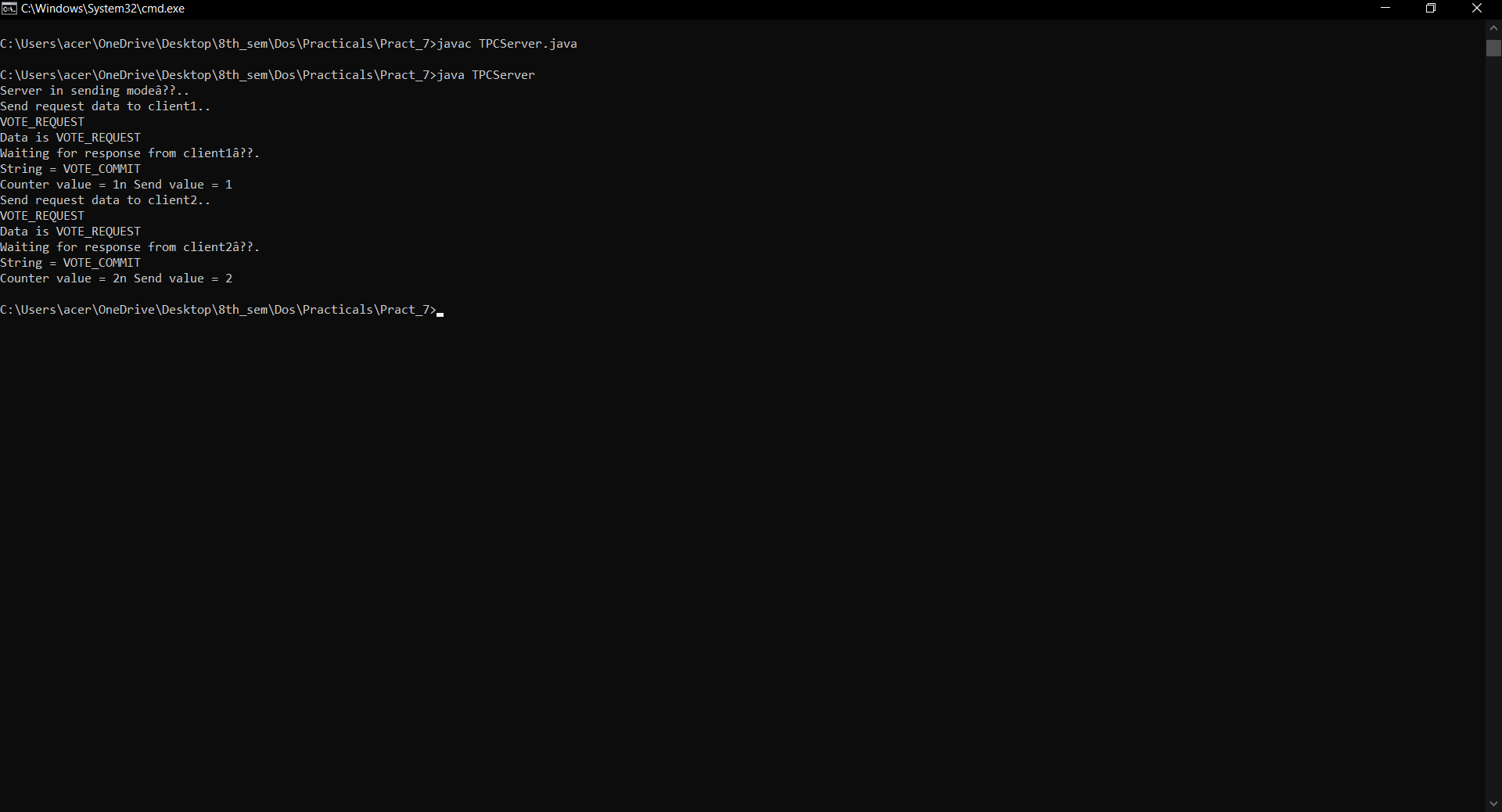
            String msgStr=new String(dp.getData(),0,dp.getLength());

            System.out.println(msgStr);

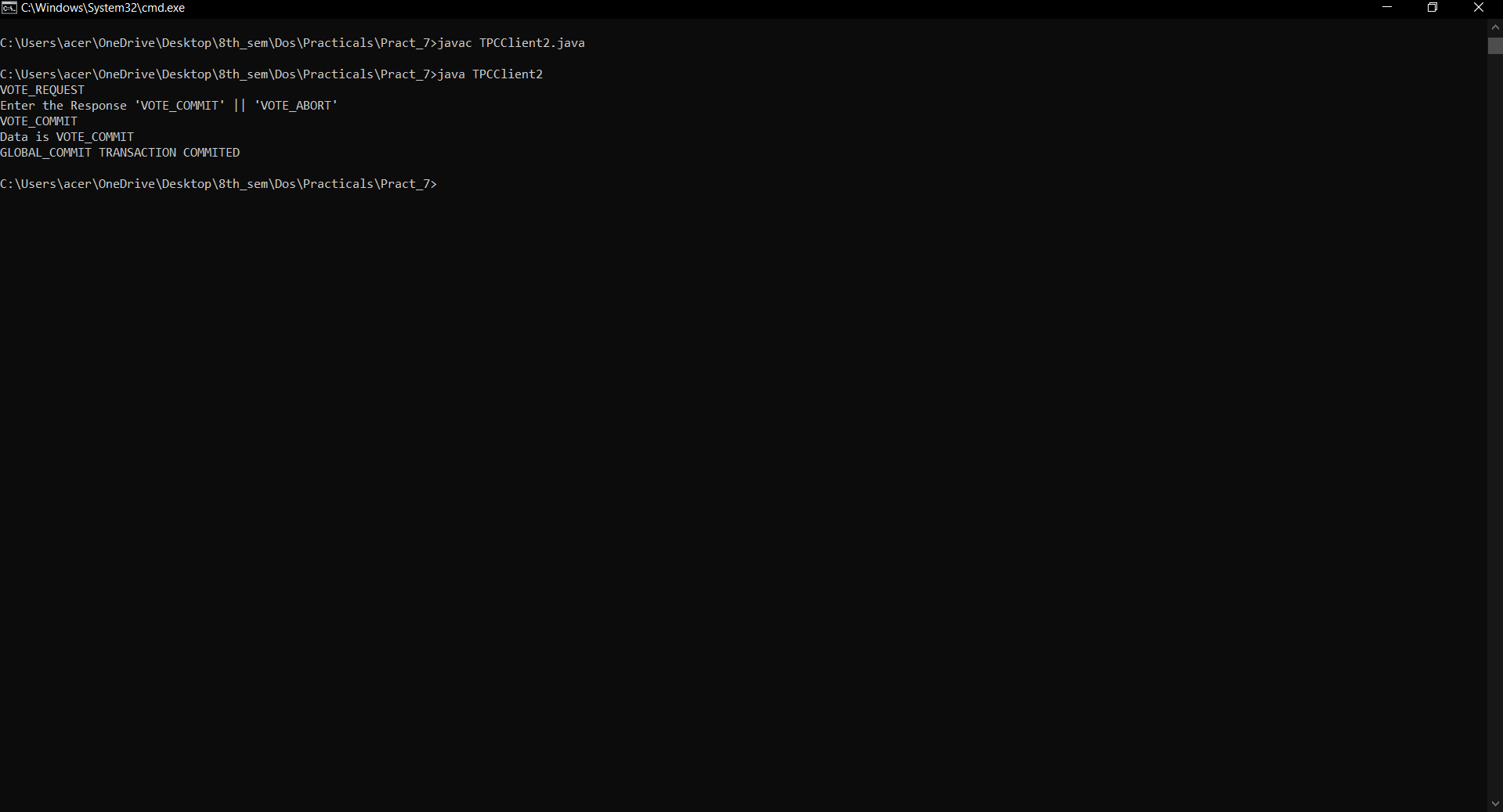
    }

};

**OUTPUT:**







**CONCLUSION:**

Hence we have successfully built a program to implement two phase commit protocol.