

National Institute of Technology Raipur

Distributed Systems Lab File

By

Obulasetty Rama Akhil

Roll No. 19115054

(7th Semester, Computer Science & Engineering)

S. no.	Experiment/ Practical	Pg. no.
1.	Implement concurrent echo client-server application in JAVA	3
2.	Implement a Distributed Chat Server using TCP Sockets in JAVA	5
3.	Implement concurrent day -time client-server application in JAVA	9
4.	Con figure following options on server socket and tests them: SO_KEEPA LIV E, SO_LINGER, SO_SNDBUF, SO_RCV BUF, TCP_NODELAY	11
5.	Write a program to Incrementing a counter in shared memory in JAVA	14
6.	Write a program to Simulate the Distributed Mutual Exclusion.	17
7.	Write a program to Implement Java RMI mechanism for accessing method	19
8.	Write a program to Create CORBA based server-client application	23

Practical 1

Aim: Implement concurrent echo client-server application in java.

Theory: TCP stands for Transmission Control Protocol, a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks. TCP organizes data so that it can be transmitted between a server and a client. It guarantees the integrity of the data being communicated over a network. Before it transmits data, TCP establishes a connection between a source and its destination, which it ensures remains live until communication begins. It then breaks large amounts of data into smaller packets, while ensuring data integrity is in place throughout the process.

Code: We make two files.

Tcpserver.java:

```
import java.io.*;
import java.net.*;

public class TcpServer {
    public static void main(String[] args) throws Exception {
        ServerSocket ss=new ServerSocket(8088);
        System.out.println("server is ready!");
        Socket ls=ss.accept();
        while (true){
            System.out.println("Client Port is "+ls.getPort());
            //READING DATA FROM CLIENT
            InputStream is=ls.getInputStream();
            byte data[]=new byte[50];
            is.read(data);
            String mfc=new String(data);
            //mfc: message from client
            mfc=mfc.trim();
            String mfs="The message was:"+mfc;
            //mfs: message from server
            //SENDING MSG TO CLIENT
            OutputStream os=ls.getOutputStream();
            os.write(mfs.getBytes());
        }
    }
}
```

Tcpclient.java:

```
import java.net.*;
import java.io.*;

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

```

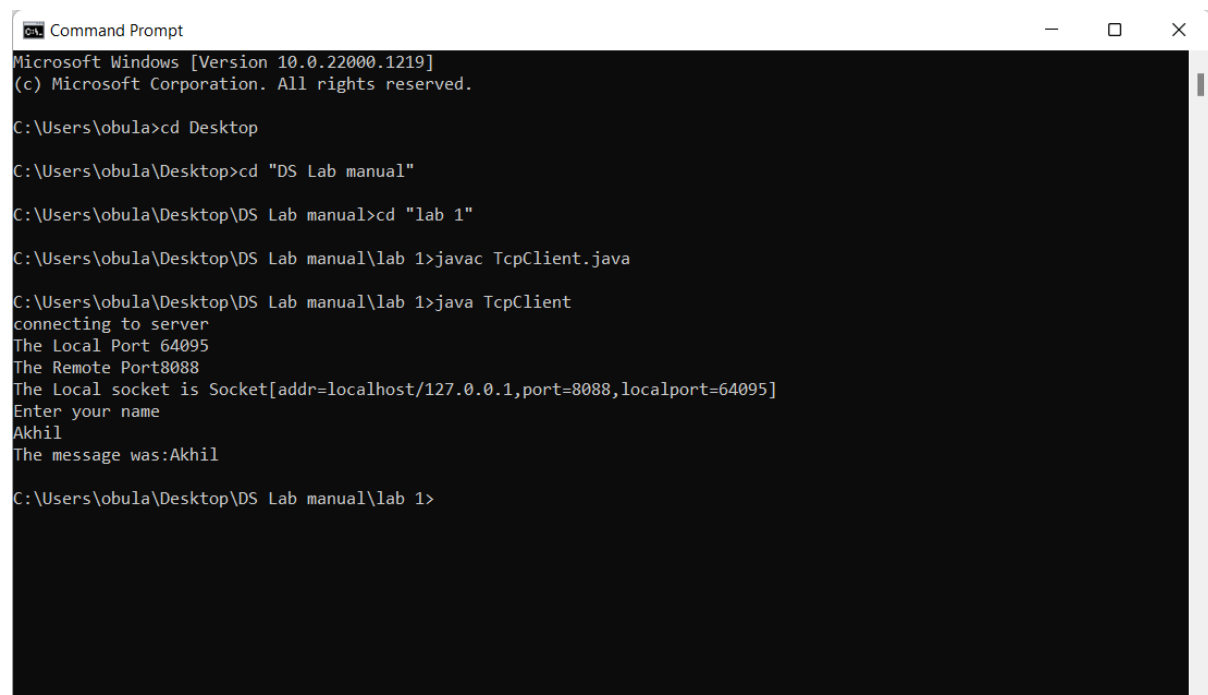
        System.out.println("connecting to server");
        Socket cs=new Socket("localhost",8088);

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

        System.out.println("The Local Port "+cs.getLocalPort()+"\nThe Remote
Port"+cs.getPort());
        System.out.println("The Local socket is "+cs);
        System.out.println("Enter your name");
        String str=br.readLine();
        //SENDING DATA TO SERVER
        OutputStream os=cs.getOutputStream();
        os.write(str.getBytes());
        //READING DATA FROM SERVER
        InputStream is=cs.getInputStream();
        byte data[]=new byte[50];
        is.read(data);
        //PRINTING MESSAGE ON CLIENT CONSOLE
        String mfs=new String(data);
        mfs=mfs.trim();
        System.out.println(mfs);
    }
}

```

Output:



```

Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

C:\Users\obula>cd Desktop
C:\Users\obula\Desktop>cd "DS Lab manual"
C:\Users\obula\Desktop\DS Lab manual>cd "lab 1"
C:\Users\obula\Desktop\DS Lab manual\lab 1>javac TcpClient.java
C:\Users\obula\Desktop\DS Lab manual\lab 1>java TcpClient
connecting to server
The Local Port 64095
The Remote Port8088
The Local socket is Socket[addr=localhost/127.0.0.1,port=8088,localport=64095]
Enter your name
Akhil
The message was:Akhil
C:\Users\obula\Desktop\DS Lab manual\lab 1>

```

Practical 2

Aim: Implement a Distributed Chat Server using TCP Sockets in JAVA.

Theory: We first define a graphical user interface for the chat boxes in ChatGUI.java file. This is written using Java AWT. Next, we create the Server app, which implements the Runnable interface and has a server socket to connect to the client. It also creates the server-side chat GUI. Finally, we create ClientApp.java which uses the Chat GUI defined earlier and creates the chat box for the client.

Code:

ServerApp.java

```
import java.io.InputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerApp implements Runnable{

    public static Socket s=null;
    public static int i=1;
    public static String clientName = "";
    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub
        ServerSocket ss = new ServerSocket(8089);
        ServerApp sa = new ServerApp();
        Thread t;
        try{
            while(true){
                System.out.println("Waiting for client "+i);
                s = ss.accept();
                i++;
                t = new Thread(sa);
                t.start();
            }
        }catch (Exception e) {
            // TODO: handle exception
        }
        finally{
            ss.close();
        }
    }
    @Override
    public void run() {
        try
        {
            InputStream is = s.getInputStream();
            byte[] b = new byte[1024];
            is.read(b);
```

```

        clientName="";
        clientName = new String(b).trim();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    new ChatGUI(s,clientName);
}
}

```

ClientApp.java

```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;

public class ClientApp {

    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub

        System.out.print("Enter your name:");
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        String name = br.readLine();
        Socket s = new Socket("localhost",8089);
        OutputStream os = s.getOutputStream();
        os.write(name.getBytes());
        new ChatGUI(s,"Admin");
    }

}

```

ChatGUI.java:

```

import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import java.net.Socket;
import java.net.SocketException;

import javax.swing.JButton;
import javax.swing.JFrame;

```

```

import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class ChatGUI extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;
    Socket s;
    JButton button;
    JTextArea ta1, ta2;
    String msg = "", title;
    JScrollPane scrollPane1, scrollPane2;
    InputStream is;
    OutputStream os;

    ChatGUI(Socket x, String str) {
        s = x;
        title = str;
        button = new JButton("SEND");
        ta1 = new JTextArea(5, 20);
        ta2 = new JTextArea(5, 20);
        ta1.setEditable(false);
        scrollPane1 = new JScrollPane(ta1);
        scrollPane2 = new JScrollPane(ta2);
        setLayout(new FlowLayout());
        add(scrollPane1);
        add(scrollPane2);
        add(button);
        button.addActionListener(this);
        setSize(300, 300);
        setVisible(true);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setTitle("Messenger " + title);
        try {
            is = s.getInputStream();
            os = s.getOutputStream();
        } catch (IOException ioe) {
        }

        try {
            chat();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    @SuppressWarnings("deprecation")
    public void chat() throws Exception {
        while (true) {
            try {

```

```

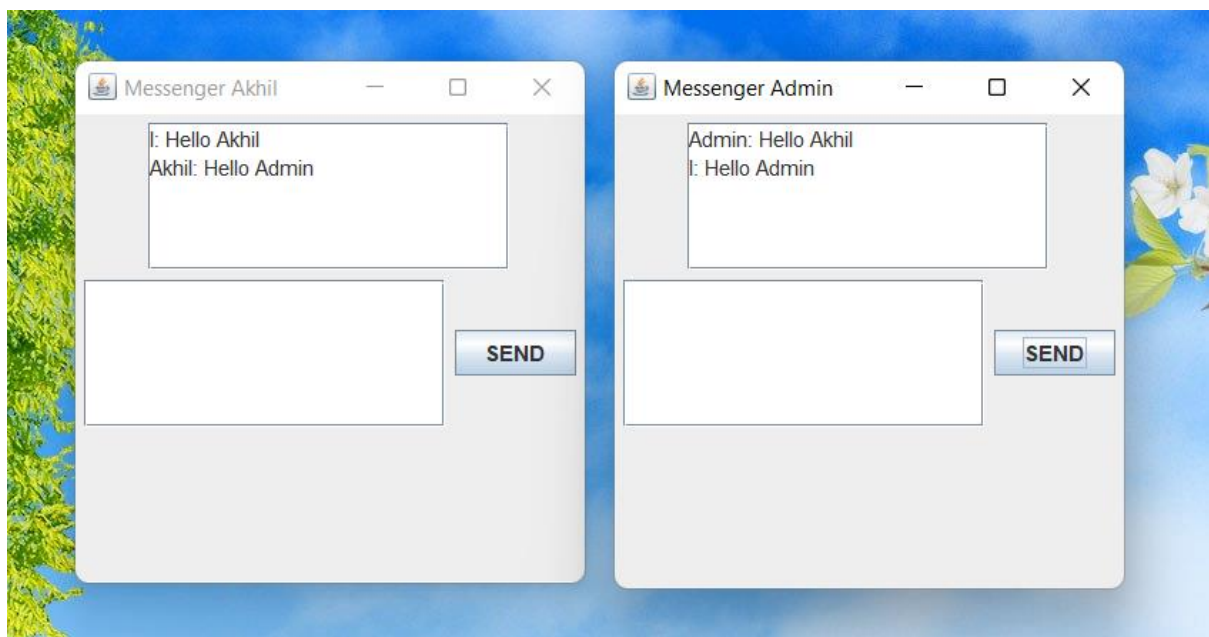
        byte data[] = new byte[50];
        is.read(data);
        msg = new String(data).trim();
        ta1.append(title+": " + msg + "\n");
    } catch (SocketException se) {
        JOptionPane.showMessageDialog(this, "Disconnected from
"+title);

        this.dispose();
        Thread.currentThread().stop();
    }
}

public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    msg = ta2.getText();
    try {
        os.write(msg.getBytes());
    } catch (IOException ioe) {
        // TODO Auto-generated catch block
        ioe.printStackTrace();
    }
    ta1.append("I: " + msg + "\n");
    ta2.setText("");
}
}

```

Output:



Practical 3

Aim: Implement concurrent day-time client-server application in JAVA

Theory: TCP stands for Transmission Control Protocol, a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks. TCP organizes data so that it can be transmitted between a server and a client. We use TCP to get server day time value to client.

Code:

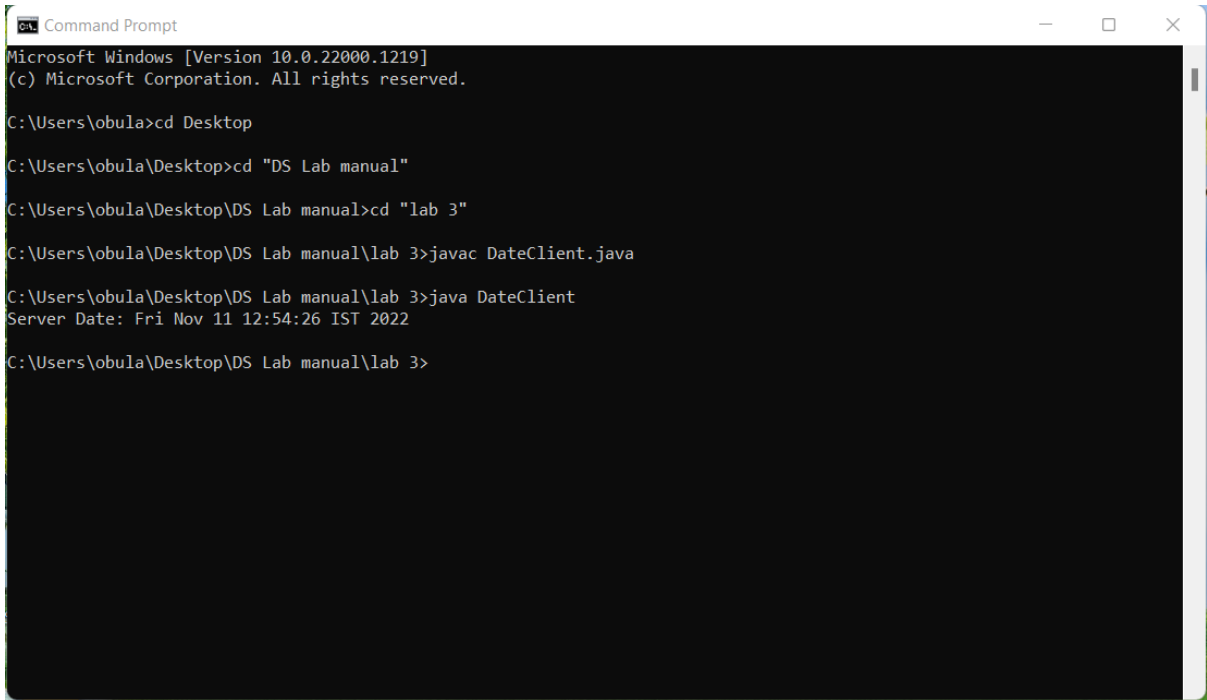
DateClient.java

```
import java.io.*;
import java.net.*;
class DateClient
{
    public static void main(String args[]) throws Exception
    {
        Socket soc =new Socket(InetAddress.getLocalHost(),5217);
        BufferedReader in=new BufferedReader(new InputStreamReader(soc.getInputStream()
));
        System.out.println(in.readLine());
    }
}
```

DateServer.java

```
import java.net.*;
import java.io.*;
import java.util.*;
class DateServer
{
    public static void main(String args[]) throws Exception
    {
        ServerSocket s=new ServerSocket(5217);
        while(true)
        {
            System.out.println("Waiting For Connection ...");
            Socket soc=s.accept();
            DataOutputStream out=new DataOutputStream(soc.getOutputStream());
            out.writeBytes("Server Date: " + (new Date()).toString() + "\n");
            out.close();
            soc.close();
        }
    }
}
```

Output:



```
Command Prompt
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

C:\Users\obula>cd Desktop
C:\Users\obula\Desktop>cd "DS Lab manual"
C:\Users\obula\Desktop\DS Lab manual>cd "lab 3"
C:\Users\obula\Desktop\DS Lab manual\lab 3>javac DateClient.java
C:\Users\obula\Desktop\DS Lab manual\lab 3>java DateClient
Server Date: Fri Nov 11 12:54:26 IST 2022
C:\Users\obula\Desktop\DS Lab manual\lab 3>
```

Practical 4

Aim: Configure following options on server socket and tests them: SO_KEEPALIVE, SO_LINGER, SO_SNDBUF, SO_RCVBUF, TCP_NODELAY.

Theory:

- The setKeepAlive() method of Java Socket class returns a Boolean value 'true' if the write-half of the socket connection has been closed successfully else it returns false.
- The setSoLinger() method of Java Socket class enables or disables the SO_LINGER option with the given linger time in seconds. It is used to specify how the close() method affects socket using a connection-oriented protocol. The timeout value is platform-specific, and this setting only affects the socket close.
- The setSendBufferSize () method of Java Socket class sets the SO_SNDBUF option to the given value for this socket. The size value should be greater than 0.
- The setReceiveBufferSize() method of Java Socket class sets the SO_RCVBUF option to the given value for the specified socket. The buffer size should be greater than zero else, it will throw an IllegalArgumentException.
- The setTcpNoDelay () method of Java Socket class enables or disables the TCP_NODELAY option.

Code:

We make 2 files:

Tcpserver.java

```
import java.io.*;
import java.net.*;
import java.util.Date;

public class TcpServer {
    public static void main(String[] args) throws Exception {
        ServerSocket ss=new ServerSocket(8088);
        System.out.println("server is ready!");
        Socket ls=ss.accept();
        while (true){
//            System.out.println("Client Port is "+ls.getPort());
//READING DATA FROM CLIENT
            InputStream is=ls.getInputStream();
            byte data[]=new byte[50];
            is.read(data);
            String mfc=new String(data);
            //mfc: message from client
            mfc=mfc.trim();
            Date d=new Date();
```

```

        String mfs=d.toString();
        //mfs: message from server
        //SENDING MSG TO CLIENT
        OutputStream os=ls.getOutputStream();
        os.write(mfs.getBytes());
    }
}

```

Tcpclient.java:

```

import java.net.*;
import java.io.*;

class TcpClient {
    public static void main(String[] args) throws Exception    {
        System.out.println("connecting to server");
        Socket cs=new Socket("localhost",8088);
        boolean on=true;
        int timeout=90;
        cs.setKeepAlive(on);
        cs.setSoLinger(on,timeout);
        cs.setSendBufferSize(67);
        cs.setReceiveBufferSize(789);
        cs.setTcpNoDelay(on);

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

        System.out.println("The Local Port "+cs.getLocalPort()+"\n\nThe Remote
Port"+cs.getPort());
        System.out.println("The Local socket is "+cs);
        System.out.println("Enter your name");
        String str=br.readLine();
        //SENDING DATA TO SERVER
        OutputStream os=cs.getOutputStream();
        os.write(str.getBytes());
        //READING DATA FROM SERVER
        InputStream is=cs.getInputStream();
        byte data[]=new byte[50];
        is.read(data);
        //PRINTING MESSAGE ON CLIENT CONSLOE
        String mfs=new String(data);
        mfs=mfs.trim();
        System.out.println(mfs);

        System.out.println("SO_KEEPALIVE is enabled: "+cs.getKeepAlive());
    }
}

```

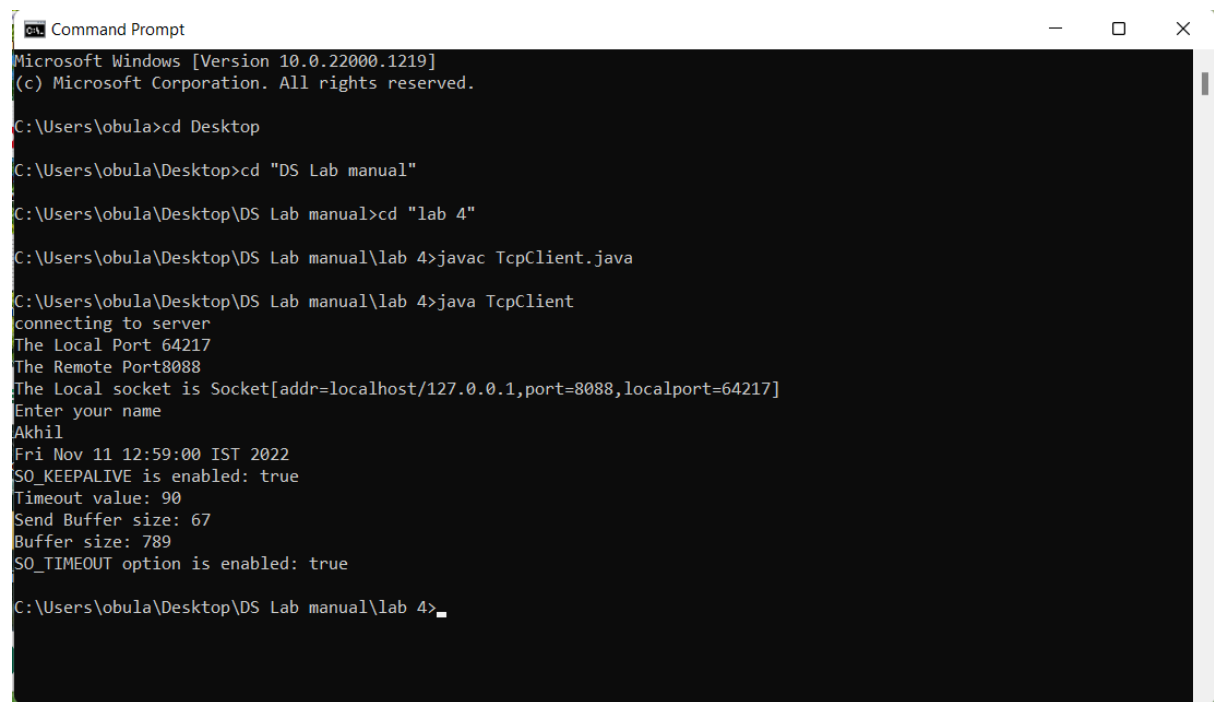
```

        System.out.println("Timeout value: "+cs.getSoLinger());
        System.out.println("Send Buffer size: "+cs.getSendBufferSize());
        System.out.println("Buffer size: "+cs.getReceiveBufferSize());
        System.out.println("SO_TIMEOUT option is enabled: "+cs.getTcpNoDelay());

    }
}

```

Output:



```

C:\Users\obula>cd Desktop
C:\Users\obula\Desktop>cd "DS Lab manual"
C:\Users\obula\Desktop\DS Lab manual>cd "lab 4"
C:\Users\obula\Desktop\DS Lab manual\lab 4>javac TcpClient.java
C:\Users\obula\Desktop\DS Lab manual\lab 4>java TcpClient
connecting to server
The Local Port 64217
The Remote Port8088
The Local socket is Socket[addr=localhost/127.0.0.1,port=8088,localport=64217]
Enter your name
Akhil
Fri Nov 11 12:59:00 IST 2022
SO_KEEPALIVE is enabled: true
Timeout value: 90
Send Buffer size: 67
Buffer size: 789
SO_TIMEOUT option is enabled: true
C:\Users\obula\Desktop\DS Lab manual\lab 4>

```

Practical 5

Aim: Write a program to Incrementing a counter in shared memory.

Theory: To increment a counter in shared memory, we use unsynchronised threads to update it. Multiple threads are created, each of which increments the shared static counter variable a certain number of times. Due to the unsynchronised nature, the actual increment is less than expected value.

Code:

```
import java.util.Scanner;
public class UnsynchronizedCounterTest {

    static class Counter {
        int count;
        void inc() {
            count = count+1;
        }
        int getCount() {
            return count;
        }
    }

    static Counter counter;          // The counter that will be incremented.
    static int numberOfIncrements; // Number of times each thread will increment it.

    static class IncrementerThread extends Thread {
        public void run() {
            for (int i = 0; i < numberOfIncrements; i++) {
                counter.inc();
            }
        }
    }

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in); // For reading the user's inputs.

        while (true) {

            /* Get number of threads and number of increments per thread
             * from the user. Exit if number of threads is <= 0. */

            System.out.println();
            System.out.print("How many threads do you want to run (Enter 0 to end)? ");
            int numberOfThreads = in.nextInt();
            if (numberOfThreads <= 0)
                break;
        }
    }
}
```

```

do {
    System.out.println();
    System.out.println("How many times should each thread increment the counter? ");
    numberOfIncrements = in.nextInt();
    if (numberOfIncrements <= 0) {
        System.out.println("Number of increments must be positive.");
    }
} while (numberOfIncrements <= 0);

System.out.println();
System.out.println("Using " + numberOfThreads + " threads.");
System.out.println("Each thread increments the counter "
    + numberOfIncrements + " times.");

/* Create the threads and start them. */

System.out.println();
System.out.println("Working...");
System.out.println();
IncrementerThread[] workers = new IncrementerThread[numberOfThreads];
counter = new Counter();
for (int i = 0; i < numberOfThreads; i++)
    workers[i] = new IncrementerThread();
for (int i = 0; i < numberOfThreads; i++)
    workers[i].start();

/* Wait for all threads to terminate. */

for (int i = 0; i < numberOfThreads; i++) {
    try {
        workers[i].join();
    }
    catch (InterruptedException e) {
    }
}

/* Display the results. */

System.out.println("The final value of the counter should be "
    + (numberOfIncrements*numberOfThreads));
System.out.println("Actual final value of counter is: " + counter.getCount());
System.out.println();
System.out.println();

} // end while

} // end main()

} // end class UnsynchronizedCounterTest

```

Output:

```
Command Prompt - java UnsynchronizedCounterTest
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

C:\Users\obula>cd Desktop

C:\Users\obula\Desktop>cd "DS Lab manual"

C:\Users\obula\Desktop\DS Lab manual>cd "lab 5"

C:\Users\obula\Desktop\DS Lab manual\lab 5>javac UnsynchronizedCounterTest.java

C:\Users\obula\Desktop\DS Lab manual\lab 5>java UnsynchronizedCounterTest

How many threads do you want to run (Enter 0 to end)? 3

How many times should each thread increment the counter?
50000

Using 3 threads.
Each thread increments the counter 50000 times.

Working...

The final value of the counter should be 150000
Actual final value of counter is: 53220

How many threads do you want to run (Enter 0 to end)?
```


Practical 6

Aim: Write a program to Simulate the Distributed Mutual Exclusion.

Theory: Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process cannot enter its critical section while another concurrent process is currently present or executing in its critical section i.e. only one process is allowed to execute the critical section at any given instance of time. In Distributed systems, we neither have shared memory nor a common physical clock and there for we cannot solve mutual exclusion problem using shared variables. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<time.h>
void main()
{
    int cs=0,pro=0;
    double run=5;
    char key='a';
    time_t t1,t2;
    clrscr();
    printf("Press a key(except q) to enter a process into critical section.");
    printf(" \nPress q at any time to exit.");
    t1 = time(NULL) - 5;
    while(key!='q')
    {
        while(!kbhit())
        if(cs!=0)
        {
            t2 = time(NULL);
            if(t2-t1 > run)
            {
                printf("Process%d ",pro-1);
                printf(" exits critical section.\n");
                cs=0;
            }
        }
        key = getch();
        if(key!='q')
        {
            if(cs!=0)
                printf("Error: Another process is currently executing critical section Please wait till its execution is over.\n");
            else
            {
                printf("Process %d ",pro);
```

```
printf(" entered critical section\n");
cs=1;
pro++;
t1 = time(NULL);
}
}
}
}
```

Output:

```
Press a key(except q) to enter a process into critical section.
Press q at any time to exit.Process 0  entered critical section
Process0  exits critical section.
Process 1  entered critical section
Error: Another process is currently executing critical section Please wait till
its execution is over.
Process1  exits critical section.

Press any key to continue.
```

Practical 7

Aim: Write a program to Implement Java RMI mechanism for accessing methods of remote system

Theory: The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM. The RMI provides remote communication between the applications using two objects *stub* and *skeleton*. The is given the 6 steps to write the RMI program.

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmi tool.
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

Code: We make three files

RMInterfaceDemo.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface RMIDemoInterface extends Remote{
    public String sayHello() throws RemoteException;
    public int add(int a, int b) throws RemoteException;
    public int subtract(int a, int b) throws RemoteException;
    public int multiply(int a, int b) throws RemoteException;
    public int divide(int a, int b) throws RemoteException;
}
```

RMIserverDemo.java:

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

class RMIDemoImpl extends UnicastRemoteObject implements RMIDemoInterface{
    private static final long serialVersionUID = 1L;

    protected RMIDemoImpl() throws RemoteException {
        super();
        // TODO Auto-generated constructor stub
    }
}
```

```

@Override
public String sayHello() throws RemoteException {
    // TODO Auto-generated method stub
    return "Hello Client! Welcome:";
}

@Override
public int add(int a, int b) throws RemoteException {
    // TODO Auto-generated method stub
    return a+b;
}

@Override
public int subtract(int a, int b) throws RemoteException {
    // TODO Auto-generated method stub
    return a-b;
}

@Override
public int multiply(int a, int b) throws RemoteException {
    // TODO Auto-generated method stub
    return a*b;
}

@Override
public int divide(int a, int b) throws RemoteException {
    // TODO Auto-generated method stub
    return a/b;
}
}

public class RMIDemoServer {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            RMIDemoInterface rmiDemoObject = new RMIDemoImpl();
            LocateRegistry.createRegistry(1099);
            Naming.rebind("rmiDemoObject",rmiDemoObject);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

RMIClientDemo.java:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

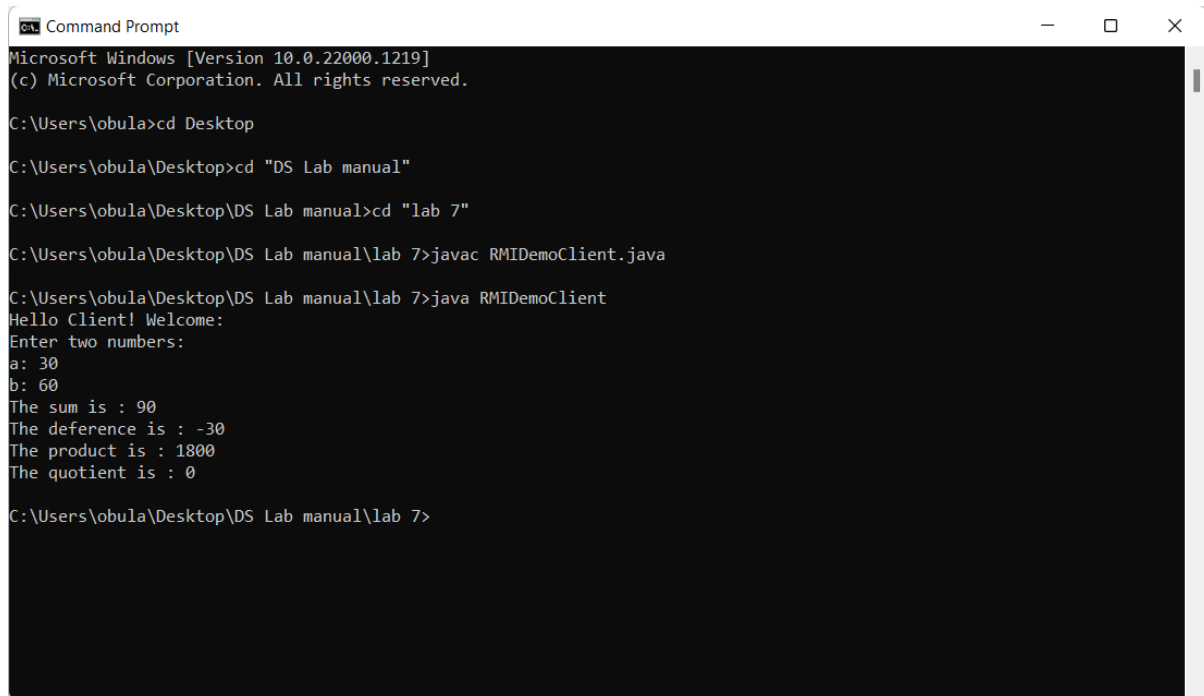
public class RMIDemoClient {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String url= "rmi://localhost:1099/rmiDemoObject";

        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        try {
            RMIDemoInterface remoteIntf = (RMIDemoInterface) Naming.lookup(url);
            System.out.println(remoteIntf.sayHello());
            System.out.println("Enter two numbers:");
            System.out.print("a: ");
            int a = Integer.parseInt(br.readLine());
            System.out.print("b: ");
            int b = Integer.parseInt(br.readLine());
            int sum = remoteIntf.add(a, b);
            int deference = remoteIntf.subtract(a, b);
            int product = remoteIntf.multiply(a, b);
            int quo = remoteIntf.divide(a, b);

            System.out.println("The sum is : "+sum);
            System.out.println("The deference is : "+deference);
            System.out.println("The product is : "+product);
            System.out.println("The quotient is : "+quo);

        } catch (MalformedURLException | RemoteException | NotBoundException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (NumberFormatException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Output:



```
Command Prompt
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

C:\Users\obula>cd Desktop
C:\Users\obula\Desktop>cd "DS Lab manual"
C:\Users\obula\Desktop\DS Lab manual>cd "lab 7"
C:\Users\obula\Desktop\DS Lab manual\lab 7>javac RMIDemoClient.java
C:\Users\obula\Desktop\DS Lab manual\lab 7>java RMIDemoClient
Hello Client! Welcome:
Enter two numbers:
a: 30
b: 60
The sum is : 90
The deference is : -30
The product is : 1800
The quotient is : 0
C:\Users\obula\Desktop\DS Lab manual\lab 7>
```

Practical 8

Aim: Write a program to Create CORBA based server-client application

Theory: The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.

CORBA is a standard for distributing objects across networks so that operations on those objects can be called remotely. CORBA is not associated with a particular programming language, and any language with a CORBA binding can be used to call and implement CORBA objects. Objects are described in a syntax called Interface Definition Language (IDL).

Code: We make 3 files:

Calc.idl

```
module CalcApp
{
    interface Calc
    {
        exception DivisionByZero {};

        float sum(in float a, in float b);
        float div(in float a, in float b) raises (DivisionByZero);
        float mul(in float a, in float b);
        float sub(in float a, in float b);
    };
};
```

CalcClient.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import CalcApp.*;
import CalcApp.CalcPackage.DivisionByZero;

import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import static java.lang.System.out;

public class CalcClient {

    static Calc calcImpl;
    static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```

public static void main(String args[]) {

    try {
        // create and initialize the ORB
        ORB orb = ORB.init(args, null);

        // get the root naming context
        org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
        // Use NamingContextExt instead of NamingContext. This is
        // part of the Interoperable naming Service.
        NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

        // resolve the Object Reference in Naming
        String name = "Calc";
        calcImpl = CalcHelper.narrow(ncRef.resolve_str(name));

//          System.out.println(calcImpl);

        while (true) {
            out.println("1. Sum");
            out.println("2. Sub");
            out.println("3. Mul");
            out.println("4. Div");
            out.println("5. exit");
            out.println("--");
            out.println("choice: ");

            try {
                String opt = br.readLine();
                if (opt.equals("5")) {
                    break;
                } else if (opt.equals("1")) {
                    out.println("a+b= " + calcImpl.sum(getFloat("a"), getFloat("b")));
                } else if (opt.equals("2")) {
                    out.println("a-b= " + calcImpl.sub(getFloat("a"), getFloat("b")));
                } else if (opt.equals("3")) {
                    out.println("a*b= " + calcImpl.mul(getFloat("a"), getFloat("b")));
                } else if (opt.equals("4")) {
                    try {
                        out.println("a/b= " + calcImpl.div(getFloat("a"), getFloat("b")));
                    } catch (DivisionByZero de) {
                        out.println("Division by zero!!!");
                    }
                }
            } catch (Exception e) {
                out.println("===");
                out.println("Error with numbers");
                out.println("===");
            }
        }
    }
}

```



```

    }
    out.println("");

    }
    //calcImpl.shutdown();
} catch (Exception e) {
    System.out.println("ERROR : " + e);
    e.printStackTrace(System.out);
}
}

static float getFloat(String number) throws Exception {
    out.print(number + ": ");
    return Float.parseFloat(br.readLine());
}
}

```

CalcServer.java

```

import CalcApp.*;
import CalcApp.CalcPackage.DivisionByZero;

import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;

import java.util.Properties;

class CalcImpl extends CalcPOA {

    @Override
    public float sum(float a, float b) {
        return a + b;
    }

    @Override
    public float div(float a, float b) throws DivisionByZero {
        if (b == 0) {
            throw new CalcApp.CalcPackage.DivisionByZero();
        } else {
            return a / b;
        }
    }

    @Override
    public float mul(float a, float b) {
        return a * b;
    }
}

```

```

@Override
public float sub(float a, float b) {
    return a - b;
}
private ORB orb;

public void setORB(ORB orb_val) {
    orb = orb_val;
}
}

public class CalcServer {

    public static void main(String args[]) {
        try {
            // create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // get reference to rootpoa & activate the POAManager
            POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootpoa.the_POAManager().activate();

            // create servant and register it with the ORB
            CalcImpl helloImpl = new CalcImpl();
            helloImpl.setORB(orb);

            // get object reference from the servant
            org.omg.CORBA.Object ref = rootpoa.servant_to_reference(helloImpl);
            Calc href = CalcHelper.narrow(ref);

            // get the root naming context
            // NameService invokes the name service
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
            // Use NamingContextExt which is part of the Interoperable
            // Naming Service (INS) specification.
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            // bind the Object Reference in Naming
            String name = "Calc";
            NameComponent path[] = ncRef.to_name(name);
            ncRef.rebind(path, href);

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

            // wait for invocations from clients
            orb.run();
        } catch (Exception e) {
            System.err.println("ERROR: " + e);
            e.printStackTrace(System.out);
        }
    }
}

```

```
        System.out.println("Exiting ...");
    }
}
```

Output:

```
243 idlj -fall Calc.idl
244 javac *.java Calc/*.java
245 mkdir Calc
246 javac *.java Calc/*.java
247 javac *.java *.java
248 idlj -fall Calc.idl
249 javac *.java Calc/*.java
250 idlj -fall Calc.idl
251 orbd -ORBInitialPort 1050&
252 java CalcServer.java -ORBInitialPort 1050 -ORBInitialHost localhost&
253 ls
254 java CalcServer -ORBInitialPort 1050 -ORBInitialHost localhost&
255 javac CalcClient.java CalcServer.java
256 java CalcServer -ORBInitialPort 1050 -ORBInitialHost localhost&
257 test
258 java CalcServer -ORBInitialPort 1050 -ORBInitialHost localhost&
```

```
test@tests-MacBook-Pro 3[CORBA] % java CalcClient -ORBInitialPort 1050 -ORBInitialHost localhost
1. Sum
2. Sub
3. Mul
4. Div
5. exit
--
choice:
3
a: 78
b: 89
a*b= 6942.0
```