

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION AFFILIATED TO VISHVESHWARYA
TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF
KARNATAKA)



Department of Computer Science and Engineering

Academic Year: 2017-2018

Application Development in Java Project on

‘MULTI-THRAEDED CLIENT-SERVER FILE SYSTEM’

Submitted by

ABHAY NAVADA	1NT15CS007
ANKIT DATTA	1NT15CS028
HARSHITH NARAHARI	1NT15CS064
ROSHAN BADRINATH	1NT15CS140

Under the able guidance of

Mr. Mohan B A

Associate Professor, Dept. of CSE, NMIT

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)

YELAHANKA, BANGALORE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the course project in Application Development using Java in VI semester entitled

Multi-Threaded Client-Server File System

is an authentic record of the project carried out by

Abhay Navada	(1NT15CS007)
Ankit Datta	(1NT15CS028)
Harshith Narahari	(1NT15CS064)
Roshan Badrinath	(1NT15CS140)

SIGNATURE OF GUIDE

.....

Mr. MOHAN B A

ASSOCIATE PROFESSOR

DEPT. OF CSE, NMIT

SIGNATURE OF HOD

.....

DR. THIPPESWAMY M N

HEAD OF DEPARTMENT

DEPT. OF CSE, NMIT

ACKNOWLEDGEMENT

We are extremely grateful to our HOD, **Dr. Thippeswamy M.N.** who extended his support towards our project. We remain indebted to our lecturer **Mr. Mohan B.A.** for his constant support in the Design, Implementation and Evaluation of the project. We are thankful to him for constructive criticism and valuable suggestions, which benefited us a lot while developing the project, also without whom we might not have been able to accomplish this project. Finally, we gratefully acknowledge the support, encouragement and patience of our friends.

CONTENTS OF THE PROJECT

Sl. No.	Title	Page No.
1	Introduction	1
2	Concepts Used	2
3	Code	5
4	Screenshots of Output	12
5	Bibliography	14
6	Plagiarism Check	15

INRODUCTION

In computer science, the Readers Writers Problem is a classic inter-process communication and synchronization problem between multiple operating system processes. The problem is analogous to that of multiple readers reading the file and a single writer writing into the file.

The file can be written by a single writer. If a writer is writing into a file, no other writer can write, no other reader can read. If a reader is reading the contents of the file, no other writer can write into the file. Multiple readers can read the file as the contents of the file are not changed.

If the client has to access the file system, he/she has to go through an authentication procedure. For authentication the client is needed to enter a pre-assigned username and password. This username and password is stored in the database which is present in the server.

Eclipse IDE is used to develop and build the project. An external .jar file which contains the JDBC driver information has to be added and configured for the project to work.

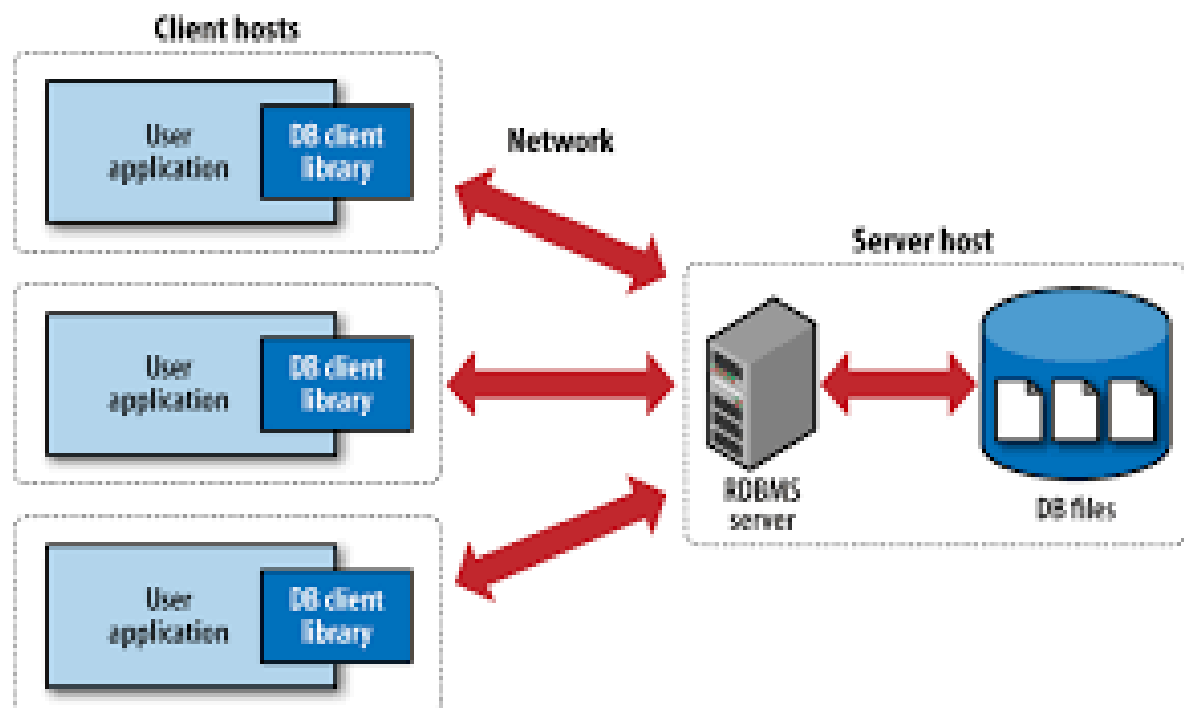


Fig 1: Client-Server Architecture

CONCEPTS USED

1. Multi-Threading:

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms:

1. Extending the Thread class
2. Implementing the Runnable Interface

```
class Class_Name extends Thread
{
    public void run()
    {
        .....
    }
}
Class_Name obj=new Class_name();
Thread t=new Thread();
t.start();
```

Fig 2: Extending Thread class to create a new thread

```
class Class_Name implement Runnable
{
    public void run()
    {
        .....
    }
}
Class_Name obj=new Class_name();
Thread t=new Thread();
t.start();
```

Fig 3: Implementing Runnable interface to create a new thread

2. Socket Programming:

Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard UNIX file descriptors. In UNIX, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

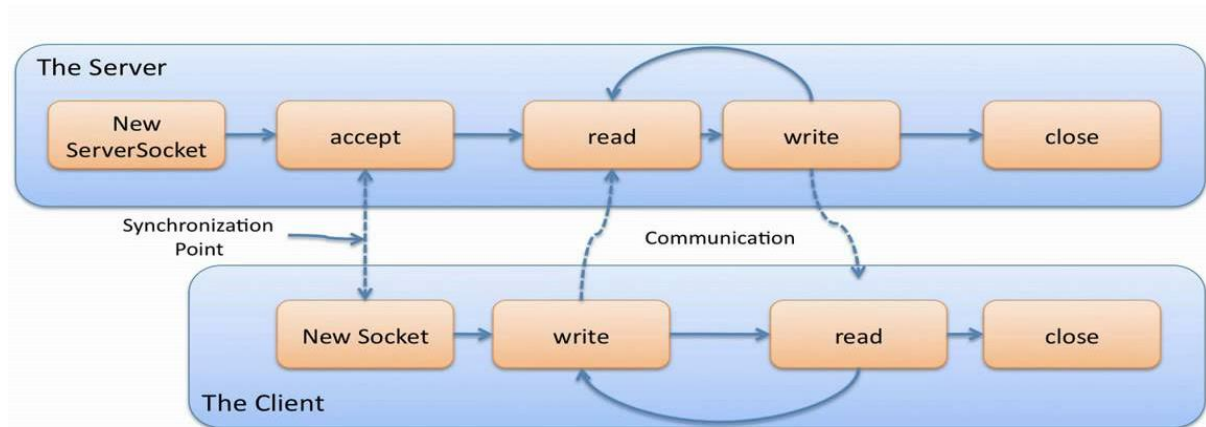


Fig 4: Java Socket Overview

3. Process Synchronization:

Process synchronization problem arises in the case of Cooperative process because resources are shared in Cooperative processes. Critical section is a code segment that can be accessed by only one process at a time. Critical section contains shared variables which need to be synchronized to maintain consistency of data variables. A Semaphore is an integer variable, which can be accessed only through two operations wait() and signal().

```

do {
    wait(wrt);
    . . .
    // writing is performed
    . . .
    signal(wrt);
}while (TRUE);
  
```

Fig 5: Structure of Writers process

```
do {
    wait(mutex);
    readcount++;
    if (readcount == 1)
        wait(wrt);
    signal(mutex);
    . . .
    // reading is performed
    . . .
    wait(mutex);
    readcount--;
    if (readcount == 0)
        signal(wrt);
    signal(mutex);
}while (TRUE);
```

Fig 6: Structure of Readers process

4. Java Database Connectivity (JDBC):

Java JDBC is a java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database. There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

- Register the driver class
- Creating connection
- Creating statement
- Executing queries
- Closing connection

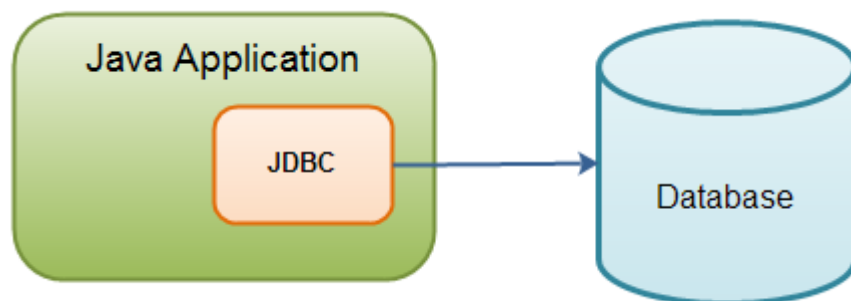


Fig 7: JDBC Overview

CODE

Server Code:

```
import java.net.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.io.*;
import java.util.concurrent.Semaphore;

public class Server {

    public static void main(String[] args) throws Exception {
        try {
            ServerSocket server = new ServerSocket(8888);
            int counter = 0;
            System.out.println("Server has started..");
            while (true) {
                counter++;
                Socket client = server.accept();
                System.out.println(">> " + "Client No:" + counter + "
started!");

                ReaderWriter rw = new ReaderWriter(client, counter);
                rw.start();
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

class ReaderWriter extends Thread {

    public static Semaphore wrt[] = {
        new Semaphore(1),
        new Semaphore(1),
        new Semaphore(1),
        new Semaphore(1),
        new Semaphore(1),
    };
};
```

```
public static Semaphore mutex[] = {
    new Semaphore(1),
    new Semaphore(1),
    new Semaphore(1),
    new Semaphore(1),
    new Semaphore(1),
};
private String clM = "", srM = "", tc, line;
Socket client;
int clientNo;
public static int readcount[] = {0,0,0,0,0};

ReaderWriter(Socket inSocket, int counter) {

    client = inSocket;
    clientNo = counter;
}

public void run() {
    try {
        DataInputStream inStream = new
DataInputStream(client.getInputStream());
        DataOutputStream outStream = new
DataOutputStream(client.getOutputStream());

        String uname = inStream.readUTF();
        String passw = inStream.readUTF();

        Class.forName("com.mysql.jdbc.Driver");
        Connection c =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306", "root", "root");
        Statement s = c.createStatement();
        String qry = "select Fullname from jee.login where Username = '" +
uname + "' and Password = '" + passw + "'";
        ResultSet r = s.executeQuery(qry);
        if (r.next()) {
            String name = r.getString("Fullname");
            System.out.println("Welcome " + name);
            outStream.writeInt(1);
            outStream.flush();
        } else {
            System.out.println("Invalid Username or Password");
            outStream.writeInt(0);
            outStream.flush();
        }
    }
}
```

```
    }
    c.close();
    s.close();
    r.close();

    int d = inStream.readInt();
    int f = inStream.readInt();
    String filename="";
    int ch = 0;

    switch (d) {
    case 1:
        switch (f) {
        case 1: filename = "Dir1/File1.txt";
                ch=1;
                break;
        case 2: filename = "Dir1/File2.txt";
                ch=2;
                break;
        }
        break;

    case 2:
        switch (f) {
        case 1: filename = "Dir2/File1.txt";
                ch=3;
                break;
        case 2: filename = "Dir2/File2.txt";
                ch=4;
                break;
        }
        break;
    }
    while (!clM.equals("exit")) {

        clM = inStream.readUTF();
        System.out.println("Client " + clientNo);

        if (clM.equals("reader")) {

            tc = "reader";
            mutex[ch].acquire();
            readcount[ch]++;
            System.out.println(readcount);
```

```
        if (readcount[ch] == 1)
            wrt[ch].acquire();
        mutex[ch].release();

        outputStream.writeUTF("Reader Ready");
        outputStream.flush();
        outputStream.writeUTF("Contents of file :");
        outputStream.flush();
        srM = "";
        System.out.println("Reader..!!");

        FileReader fr = new FileReader(filename);
        BufferedReader bfr = new BufferedReader(fr);

        while ((line = bfr.readLine()) != null) {
            srM += line;
            srM += "\n";
        }
        bfr.close();
        outputStream.writeUTF(srM);
        outputStream.flush();
    } else if (clM.equals("writer")) {
        tc = "writer";
        wrt[ch].acquire();
        srM = "Writer Ready";
        outputStream.writeUTF(srM);
        outputStream.flush();
        srM = inputStream.readUTF();
        System.out.println(srM);
        FileWriter fw = new FileWriter(filename);
        BufferedWriter bfw = new BufferedWriter(fw);
        bfw.write(srM);
        bfw.close();
    } else {
        System.out.println(clM);
    }
}

if (tc.equals("reader")) {
    System.out.println(tc);
    mutex[ch].acquire();
    readcount[ch]--;
    if (readcount[ch] == 0)
        wrt[ch].release();
}
```

```
        mutex[ch].release();
    } else if (tc.equals("writer")) {
        System.out.println(tc);
        wrt[ch].release();
    } else {
        System.out.println(tc);
    }

    inStream.close();
    outStream.close();
    client.close();
} catch (Exception ex) {
    System.out.println(ex);
} finally {
    System.out.println("Client -" + clientNo + " exit!! ");\
}
}
```

Client Code:

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

public class Client {
    public static void main(String[] args) throws Exception {
        String ip = "127.0.0.1";
        if(args.length > 0) {
            ip = args[0];
        }
        try {
            Socket socket = new Socket(ip, 8888);
            DataInputStream inStream = new
DataInputStream(socket.getInputStream());
            DataOutputStream outStream = new
DataOutputStream(socket.getOutputStream());
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

            System.out.println("Enter Username");
            String uname = br.readLine();
            outStream.writeUTF(uname);
            outStream.flush();
        }
    }
}
```

```
System.out.println("Enter Password");
String passw = br.readLine();
outStream.writeUTF(passw);
outStream.flush();

int st = inStream.readInt();
if(st == 0) {
    System.out.println("Invalid Username or Password");
    System.exit(0);
}

String clM = "", srM = "";
StringBuilder txt = new StringBuilder("");

System.out.println("Enter Directory number\n1. Dir1\t2. Dir2");
int d = Integer.parseInt(br.readLine());
outStream.writeInt(d);
outStream.flush();

System.out.println("Enter File number\n1. File1\t2. File2");
int f = Integer.parseInt(br.readLine());
outStream.writeInt(f);
outStream.flush();

System.out.println("Reader/writer :");

while (!clM.equals("exit")) {
    clM = br.readLine();
    outStream.writeUTF(clM);
    outStream.flush();

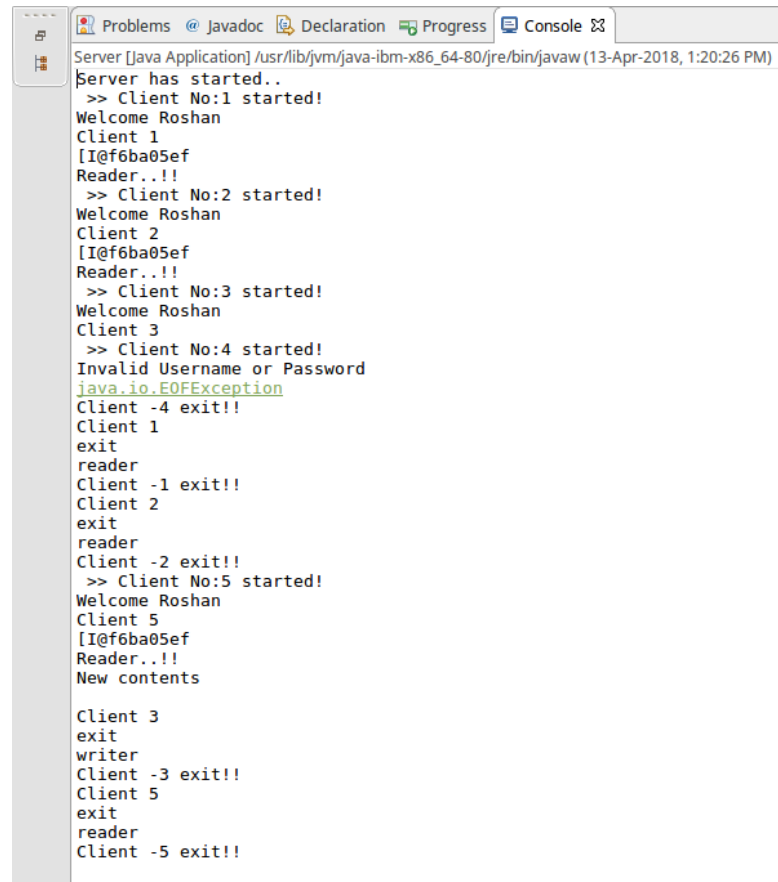
    if (clM.equals("reader")) {
        System.out.println("Busy..!");
        srM = inStream.readUTF();
        System.out.println(srM);
        srM = inStream.readUTF();
        System.out.println(srM);
        srM = inStream.readUTF();
        System.out.println(srM);
    } else if (clM.equals("writer")) {
        System.out.println("Busy..!");
        srM = inStream.readUTF();
        System.out.println(srM);
    }
}
```

```
send );");

        System.out.println("Enter Text ( enter * in new line to

        clM = br.readLine();
        while (!(clM.equals("*"))) {
            txt.append(clM);
            txt.append("\n");
            clM = br.readLine();
        }
        clM = txt.toString();
        outputStream.writeUTF(clM);
        outputStream.flush();
    } else {
        System.out.println("Bye");
    }
}
outStream.close();
outStream.close();
socket.close();
} catch (Exception e) {
    System.out.println(e);
}
}
}
```

SCREENSHOTS OF THE OUTPUT

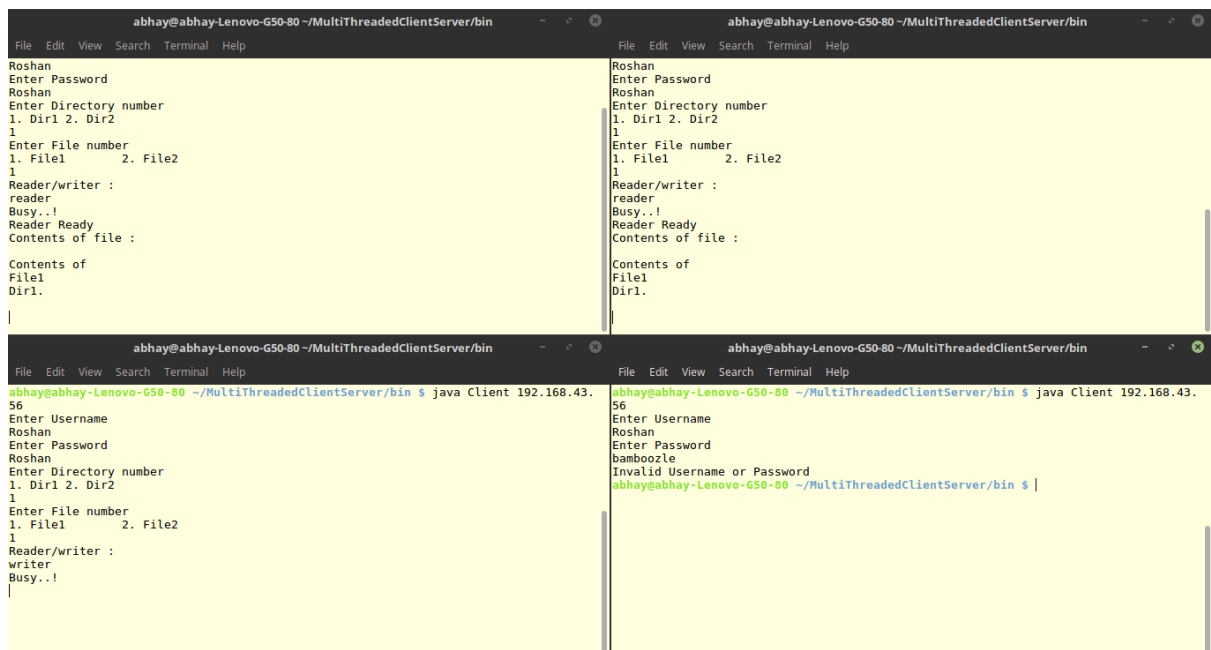


```

Server [Java Application] /usr/lib/jvm/java-ibm-x86_64-80/jre/bin/javaw (13-Apr-2018, 1:20:26 PM)
Server has started..
>> Client No:1 started!
Welcome Roshan
Client 1
[I@f6ba05ef
Reader...!!
>> Client No:2 started!
Welcome Roshan
Client 2
[I@f6ba05ef
Reader...!!
>> Client No:3 started!
Welcome Roshan
Client 3
>> Client No:4 started!
Invalid Username or Password
java.io.EOFException
Client -4 exit!!
Client 1
exit
reader
Client -1 exit!!
Client 2
exit
reader
Client -2 exit!!
>> Client No:5 started!
Welcome Roshan
Client 5
[I@f6ba05ef
Reader...!!
New contents

Client 3
exit
writer
Client -3 exit!!
Client 5
exit
reader
Client -5 exit!!
  
```

Fig 8: Server Process Output



```

abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin
File Edit View Search Terminal Help
Roshan
Enter Password
Roshan
Enter Directory number
1. Dir1 2. Dir2
1
Enter File number
1. File1 2. File2
1
Reader/writer :
reader
Busy..!
Reader Ready
Contents of file :

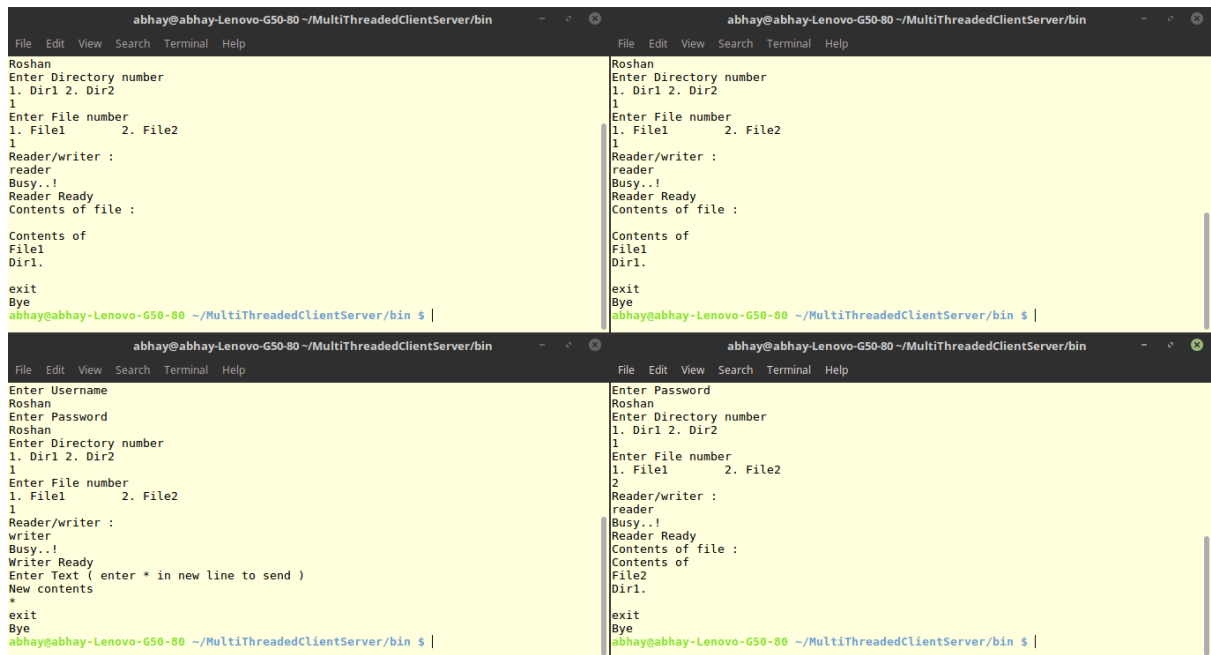
Contents of
File1
Dir1.

abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin
File Edit View Search Terminal Help
Roshan
Enter Password
Roshan
Enter Directory number
1. Dir1 2. Dir2
1
Enter File number
1. File1 2. File2
1
Reader/writer :
reader
Busy..!
Reader Ready
Contents of file :

Contents of
File1
Dir1.

abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin $ java Client 192.168.43.56
abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin $ java Client 192.168.43.56
Enter Username
Roshan
Enter Password
Roshan
bamboozle
Invalid Username or Password
abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin $
  
```

Fig 9.1: Client Process Output



```
abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin
File Edit View Search Terminal Help
Roshan
Enter Directory number
1. Dir1 2. Dir2
1
Enter File number
1. File1 2. File2
1
Reader/writer :
reader
Busy..!
Reader Ready
Contents of file :

Contents of
File1
Dir1.

exit
Bye
abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin $ |

abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin
File Edit View Search Terminal Help
Roshan
Enter Directory number
1. Dir1 2. Dir2
1
Enter File number
1. File1 2. File2
1
Reader/writer :
reader
Busy..!
Reader Ready
Contents of file :

Contents of
File1
Dir1.

exit
Bye
abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin $ |

abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin
File Edit View Search Terminal Help
Enter Username
Roshan
Enter Password
Roshan
Enter Directory number
1. Dir1 2. Dir2
1
Enter File number
1. File1 2. File2
1
Reader/writer :
writer
Busy..!
Writer Ready
Enter Text ( enter * in new line to send )
New contents
*
exit
Bye
abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin $ |

abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin
File Edit View Search Terminal Help
Enter Password
Roshan
Enter Directory number
1. Dir1 2. Dir2
1
Enter File number
1. File1 2. File2
2
Reader/writer :
reader
Busy..!
Reader Ready
Contents of file :
Contents of
File2
Dir1.

exit
Bye
abhay@abhay-Lenovo-G50-80 ~/MultiThreadedClientServer/bin $ |
```

Fig 9.2: Client Process Output

BIBLIOGRAPHY

1. The Complete Reference Java – Herbert Schildt
2. Operating System Concepts – Abraham Silberschatz, Peter Baer Galvin, Greg Gagne
3. <https://www.geekforgeeks.com>
4. <https://www.tutorialspoint.com>

90.1% Unique

Multi-Threaded Client-Server File System

Sources:	Similarity index:	View in the text:
https://www.mkyong.com/java/java-thread-mutex-and-semaphore-example/	7.8	Show
https://www.admfactory.com/thread-semaphore-and-mutex-in-java/	7.4	Show
https://www.geeksforgeeks.org/semaphore-in-java/	6.8	Show
http://www.examclouds.com/ava/java-core-russian/dbc-work	5.0	Show