MATURI VENKATA SUBBA RAO ENGINEERING COLLEGE

(An Autonomous Institution)

(Affiliated to Osmania University & Recognized by AICTE) Nadergul, RangaReddyDist.

# CERTIFICATE

**Department of COMPUTER SCIENCE &ENGINEERING**

Certified that this is a bonafide work of lab experiments carried out by Mr/Ms._Bearing Roll.No.________________________under the course of **Distributed Systems** ___Laboratory prescribed by Osmania University for B.E. Sem-VII of Computer Science & Engineering during the academic year 2021–2022.

***Internal Examiner*** ***External Examiner***

## VISION AND MISSION

### VISION

- To impart technical education of the highest standards, producing competent and confident engineers with an ability to use computer science knowledge to solve societalproblems.

### MISSION

- To make learning process exciting, stimulating andinteresting.
- To impart adequate fundamental knowledge and soft skills tostudents.
- To expose students to advanced computer technologies in order to excel in engineering practices by bringing out the creativity instudents.
- To develop economically feasible and socially acceptablesoftware.

## PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Bachelor's program in Computer Science and Engineering is aimed at preparing graduates who will:-
**PEO-1:**Achieverecognitionthroughdemonstrationoftechnicalcompetenceforsuccessfulexecutionofsoftware projects to meet customer businessobjectives.
**PEO-2:** Practice life-long learning by pursuing professional certifications, higher education or research in the emerging areas of information processing and intelligent systems at a global level.
**PEO-3:** Contribute to society by understanding the impact of computing using a multidisciplinary and ethical approach.

## (A) PROGRAM OUTCOMES(POs)

At the end of the program the students (Engineering Graduates) will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineeringproblems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmentalconsiderations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environmentandsustainability:**Understandtheimpactoftheprofessionalengineeringsolutionsinsocietaland environmental contexts, and demonstrate the knowledge of, and need for sustainabledevelopment.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineeringpractice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community andwithsocietyatlarge,suchas,beingabletocomprehendandwriteeffectivereportsanddesigndocumentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principle and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinaryenvironments.
12. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technologicalchange.

## (B) PROGRAM SPECIFIC OUTCOMES(PSOs)

13. (PSO-1) Demonstrate competence to build effective solutions for computational real-world problems using software and hardware across multi-disciplinarydomains.
14. (PSO-2) Adapt to current computing trends for meeting the industrial and societal needs througha holistic professional development leading to pioneering careers orentrepreneurship

**Course Name: Distributed Systems Lab**

**Course Code: PC 752CS**

**Academic Year: 2021-2022**

**Semester:VII**

**Course Objectives**

➢ To implement client and server programs using sockets

➢ To learn about working of NFS

➢ Understanding Remote Communication and Interprocess Communication

➢ To use Map, reduce model for distributed processing

➢ To develop mobile applications

**Course Outcomes**

After completing this course, the student will be able to

➢ Write programs that communicate data between two hosts

➢ Configure NFS

➢ To implement inter process communication and remote communication

➢ Use distributed data processing frameworks and mobile application tool kits

# INDEX

| S.No. | Name of the Experiment | Experiment Date | Date of Submission | Page No. |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

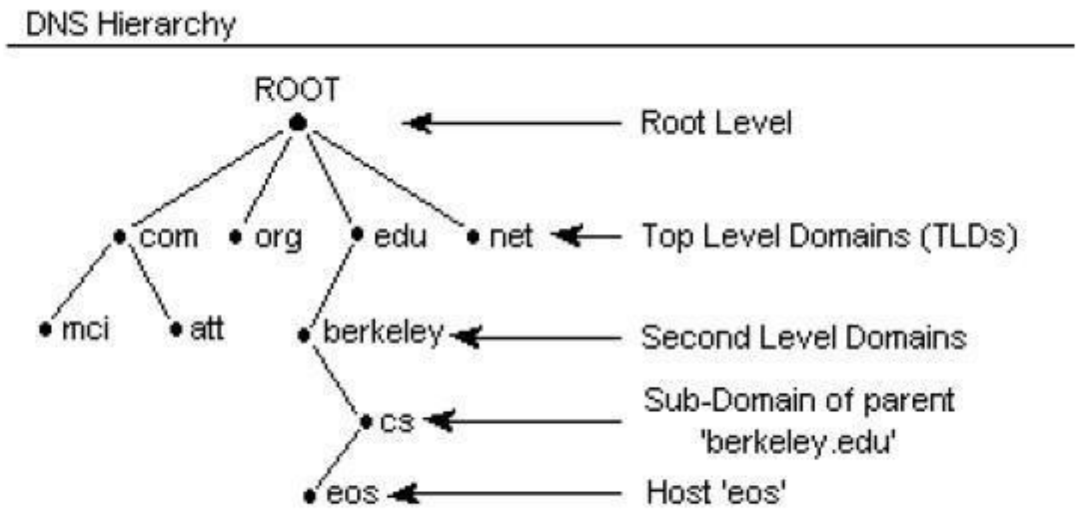
# 1.Domain Name Server Implementation

**Introduction:**

The **Domain Name System** (**DNS**) is a hierarchical naming system built on a distributed database for computers, services, or any resource connected to the Internet or a private network. The Domain Name System makes it possible to assign domain names to groups of Internet resources and users in a meaningful way, independent of each entity's physical location. Because of this, World Wide Web (WWW) hyperlinks and Internet contact information can remain consistent and constant even if the current Internet routing arrangements change.The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating iterative name servers for each domain.The Domain Name System is maintained by a distributed database system, which uses the clientserver model. The nodes of this database are the name servers. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root name servers, the servers to query when looking up (*resolving*) a top level domain name.

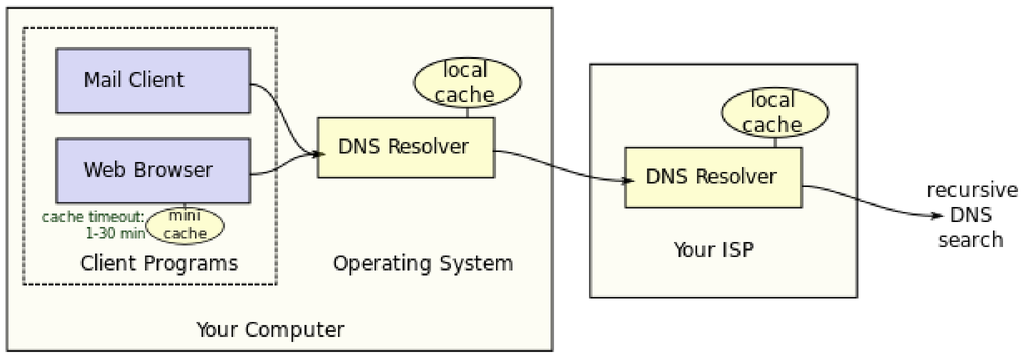
The Domain Name System also specifies the technical functionality of thedatabaseservice which is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS, as part of theInternet Protocol Suite. Historically, other directory services preceding DNS were not scalable to large or global directories as they were originally based on text files, prominently theHOSTS.TXTresolver. DNS has been in wide use since the 1980s. The most common types of records stored in the DNS database are forDNS zoneauthority (SOA),IP addresses(A and AAAA),SMTPmail exchangers(MX),name servers(NS), pointers forreverse DNS lookups(PTR), anddomain name aliases(CNAME).

Domain name resolvers determine the domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label.



**DNS resolvers**

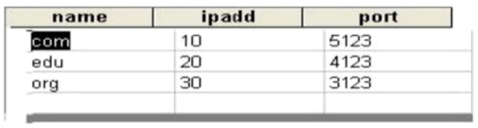
The client side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.



**Implementation:**

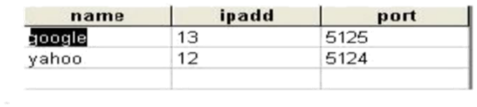
- Design the table with in the database as specified in the next section.
- Implement individual server for each domain with in DNS.
- Each server program is now in waiting state.
- Implement client program which accepts URL from user, divides the URL into different domain and fetch the IPs of the domains from the corresponding servers.
- Combine the IPs received from different server and display them to the user.

Table Name : **Client**



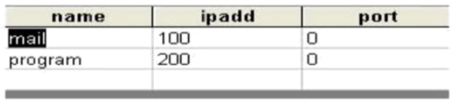
| name | ipadd | port |
|---|---|---|
| com | 10 | 5123 |
| edu | 20 | 4123 |
| org | 30 | 3123 |

Table Name: Root



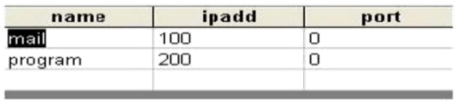
| name | ipadd | port |
|---|---|---|
| google | 13 | 5125 |
| yahoo | 12 | 5124 |

Table Name: Yahoo



| name | ipadd | port |
|---|---|---|
| mail | 100 | 0 |
| program | 200 | 0 |

Table Name: Google



| name | ipadd | port |
|---|---|---|
| mail | 100 | 0 |
| program | 200 | 0 |

**Server1.java**

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.sql.Connection;importjava.util.*;
class Server1 {
public static void main(String a[]) {
ServerSocket sock;
Socket client;
DataInputStream input;
PrintStreamps;
String url, u, s;
Connection con;
Statement          smt;
ResultSetrs;       try {
s = u = "\0";
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dns", "root", "");
smt = con.createStatement();           sock = new ServerSocket(5123);
while (true) {               client =
sock.accept();
input = new DataInputStream(client.getInputStream());
ps = new PrintStream(client.getOutputStream());              url
= input.readLine();
System.out.println("IN SERVER1 URL IS:" + url);
StringTokenizerst = new StringTokenizer(url, ".");
while (st.countTokens() > 1)                    s = s +
st.nextToken() + ".";                 s = s.substring(0, s.length()
- 1).trim();                u = st.nextToken();
rs = smt.executeQuery("select port,ipadd from root where name='" + u + "'");
if (rs.next()) {
ps.println(Integer.parseInt(rs.getString(1)));
ps.println(Integer.parseInt(rs.getString(2)));
ps.println(s);                } else {
ps.println("Illegal address pleasr check the spelling again");
con.close();
}
}
} catch (Exception e) {
System.err.println(e);
}
}
}
```

**Server2.java**

```
import java.io.*;
import java.net.*;
import java.sql.*;
import
java.sql.Connection;i
mportjava.util.*; class
Server2 {
public static void main(String a[]) {
ServerSocket sock;
Socket client;
DataInputStream input;
PrintStreamps;
String url, u, s;
Connection con;
Statement smt;
ResultSetrs;
try {
s = u = "\0";
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dns", "root", "");
smt = con.createStatement();
sock = new ServerSocket(5124);
while (true) {
client = sock.accept();
input = new DataInputStream(client.getInputStream());
ps = new PrintStream(client.getOutputStream());                url
= input.readLine();
System.out.println("IN SERVER2 URL IS:" + url);
StringTokenizerst = new StringTokenizer(url, ".");
while (st.countTokens() > 1)
s = s + st.nextToken() + ".";
s = s.substring(0, s.length() - 1).trim();
u = st.nextToken();
rs = smt.executeQuery("select port,ipadd from yahoo where name='" + u + "'");
if (rs.next()) {
ps.println(rs.getString(1));
ps.println(rs.getString(2));
ps.println(s);
} else {
ps.println("Illegal address pleasr check the spelling again");
con.close();
}
}
} catch (Exception e) {
System.err.println(e);
}
}
}
```

**Server3.java**

```
importjava.io.*;
import java.net.*;
import java.sql.*;
import java.sql.Connection;importjava.util.*;
 class Server3 {
public static void main(String a[]) {
ServerSocket sock;
Socket client;
DataInputStream input;
PrintStreamps;
String url, u, s;
Connection con;
Statement smt;
ResultSetrs;          try {
s = u = "\0";
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dns", "root", "");              smt
= con.createStatement();
sock = new ServerSocket(5125);
while (true) {                  client =
sock.accept();
input = new DataInputStream(client.getInputStream());
ps = new PrintStream(client.getOutputStream());                 url
= input.readLine();
System.out.println("IN SERVER3 URL IS:" + url);
StringTokenizerst = new StringTokenizer(url, ".");
while (st.countTokens() > 1)
s = s + st.nextToken() + ".";
s = s.substring(0, s.length() - 1).trim();
u = st.nextToken();
rs = smt.executeQuery("select port,ipadd from google where name='" + u + "'");
if (rs.next()) {
ps.println(rs.getString(1));
ps.println(rs.getString(2));
ps.println(s);
} else {
```

```
ps.println("Illegal address pleasr check the spelling again");
con.close();
}
}
} catch (Exception e) {
System.err.println(e);
}
}
}
```

**Client.java**

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.sql.Connection;
import java.util.*;
class Client {
public static void main(String a[]) {
Socket clisock;
DataInputStream input;
PrintStreamps;
String url, ip, s, u, p, str;        int
pno = 5123;        Connection
con;
Statement smt;
ResultSetrs;        boolean
status = true;        try {
ip = s = p = u = "\0";
System.out.println("enter name to resolve");
BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));
url = br.readLine();
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dns", "root", "");
smt = con.createStatement();
while (status) {
s = "\0";
System.out.println("IN CLIENT URL IS:" + url);   StringTokenizerst =
new StringTokenizer(url, ".");
if (st.countTokens() == 1) {
status = false;
}
while (st.countTokens() > 1)
s = s + st.nextToken() + ".";
s = s.substring(0, s.length() - 1).trim();
u = st.nextToken();
System.out.println("u=" + u);
rs = smt.executeQuery("select port,ipadd from client where name='" + u + "'");
if (rs.next()) {
p = rs.getString(1);
pno = Integer.parseInt(p);
str = rs.getString(2);
url = s;
ip = str + "." + ip;
} else {
System.out.println("pno=" + pno);
clisock = new Socket("127.0.0.1", pno);
input = new DataInputStream(clisock.getInputStream());
ps = new PrintStream(clisock.getOutputStream());
ps.println(url);
p = input.readLine();
pno = Integer.parseInt(p);
str = input.readLine();
url = input.readLine();
ip = str + "." + ip;
smt.executeUpdate("insert into client values('" + u + "','" + str + "','" + p + "')");
}
System.out.println("ip=" + ip);
}
ip = ip.substring(0, ip.length() - 1).trim();
System.out.println("ip address is:" + ip);
con.close();
} catch (Exception e) {
System.err.println(e);
}

}

}
```

## Steps of Execution:

1. In one of the terminals, type the command **sudo/opt/lampp/manager***

   → Password :mvsr

2.Then the xamp server will be opened, click start all.

3.Go to https://locallhost.me/phpmyadmin in web browser and create 4 tables Client, Root,Yahoo,Google as shown in the “implementation” section above.

4. Put your database name as your roll number and make sure to replace dns (database name in the above programs) with your roll number.

5. Now, in Server1 terminal type the commands
   - javac -Xlint *va
   - javac -cp .:/share/java/mysql.jar Server1

6. In Server2 terminal, type the command
   - javac -cp .:/share/java/mysql.jar Server2

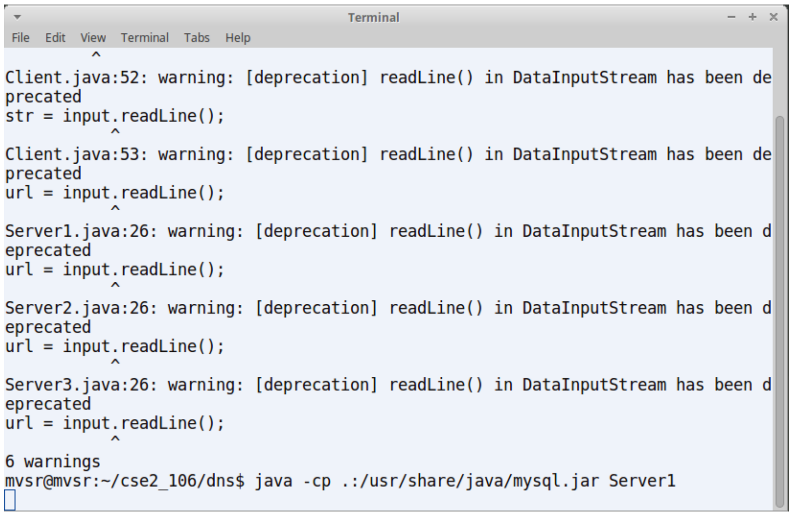
7. In Server3 terminal, type the command
   - javac -cp .:/share/java/mysql.jar Server3

8. In Client terminal, type the commands
   - javac -xlint *va
   - javac -cp .:/share/java/mysql.jar Client

   → Enter name to resolve : google.com
   → 13.10

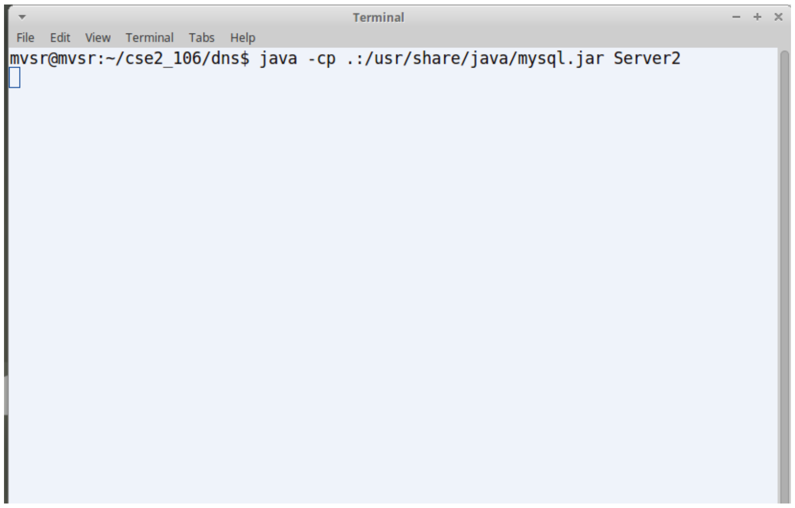
## Sample Outputs:

## Server1



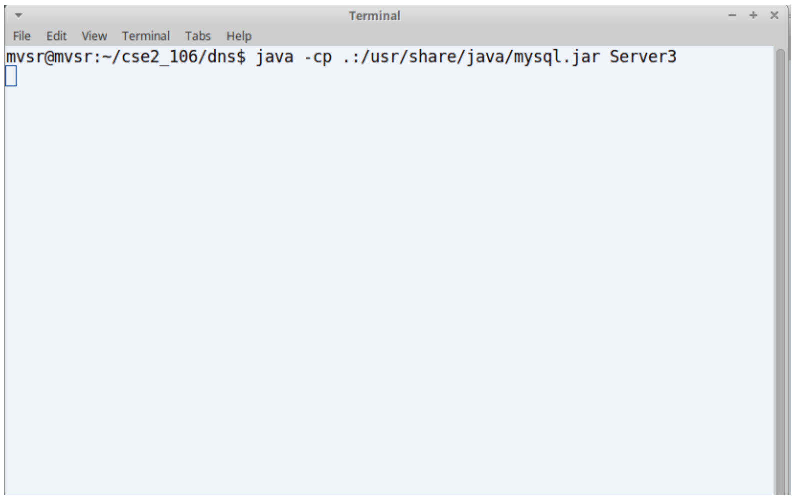
```
        ^
Client.java:52: warning: [deprecation] readLine() in DataInputStream has been de
precated
str = input.readLine();
           ^
Client.java:53: warning: [deprecation] readLine() in DataInputStream has been de
precated
url = input.readLine();
           ^
Server1.java:26: warning: [deprecation] readLine() in DataInputStream has been d
eprecated
url = input.readLine();
           ^
Server2.java:26: warning: [deprecation] readLine() in DataInputStream has been d
eprecated
url = input.readLine();
           ^
Server3.java:26: warning: [deprecation] readLine() in DataInputStream has been d
eprecated
url = input.readLine();
           ^
6 warnings
mvsr@mvsr:~/cse2_106/dns$ java -cp .:/usr/share/java/mysql.jar Server1
```

## Server2



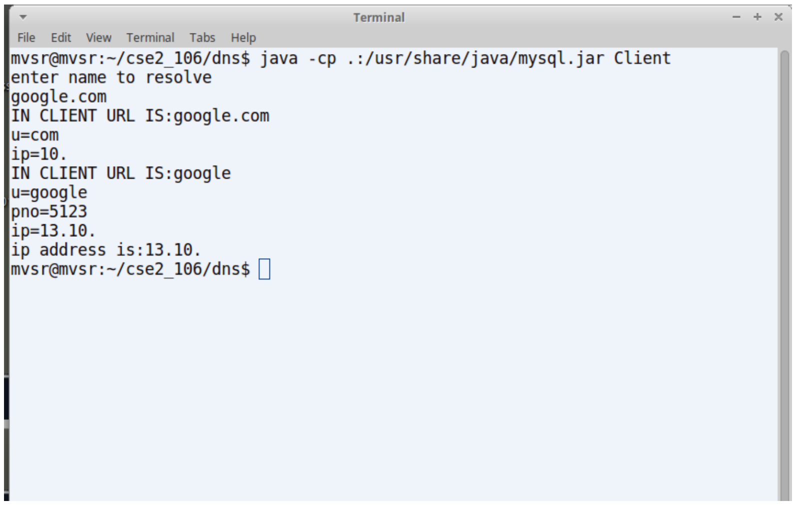
```
mvsr@mvsr:~/cse2_106/dns$ java -cp .:/usr/share/java/mysql.jar Server2
```

## Server3



```
mvsr@mvsr:~/cse2_106/dns$ java -cp .:/usr/share/java/mysql.jar Server3
```

## Client



```
mvsr@mvsr:~/cse2_106/dns$ java -cp .:/usr/share/java/mysql.jar Client
enter name to resolve
google.com
IN CLIENT URL IS:google.com
u=com
ip=10.
IN CLIENT URL IS:google
u=google
pno=5123
ip=13.10.
ip address is:13.10.
mvsr@mvsr:~/cse2_106/dns$ 
```

# 2.Implementation of DNS using RMI

## Introduction

This is a brief introduction to Java Remote Method Invocation (RMI). Java RMI is a mechanism that allows one to invoke a method on an object that exists in another address space. The other address space could be on the same machine or a different one. The RMI mechanism is basically an object-oriented RPC mechanism.

There are three processes that participate in supporting remote method invocation.

1. The *Client* is the process that is invoking a method on a remote object.
2. The *Server* is the process that owns the remote object. The remote object is an ordinary object in the address space of the server process.
3. The *Object Registry* is a name server that relates objects with names. Objects are *registered* with the

Object Registry. Once an object has been registered, one can use the Object Registry to obtain access to a remote object using the name of the object.

**RMI Stubs and Skeletons**

- RMI uses stub and skeleton objects to provide the connection between the client and the remote object
- A stub is a proxy for a remote object which is responsible for forwarding method invocations from the client to the server where the actual remote object implementation resides
- A client's reference to a remote object, therefore, is actually a reference to a local stub. The client has a local copy of the stub object.
- A skeleton is a server-side object which contains a method that dispatches calls to the actual remote object implementation
- A remote object has an associated local skeleton object to dispatch remote calls to it.
- The skeleton object is automatically provided on the server side. · A method can get a reference to a remote object
    - ➢ by looking up the remote object in some directory service. RMI provides a simple directory service called the RMI registry for this purpose
    - ➢ by receiving the remote object reference as a method argument or return value.

**Steps To Develop an RMI Application**

1. Design and implement the components of your distributed application
2. Define the remote interface(s)
3. Implement the remote object(s)
4. Implement the client(s)
5. Compile sources and generate stubs (and skeletons)
6. Make required classes network accessible.
7. Run the application.

**DNSServer.java**

```
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
public class DNSServer extends UnicastRemoteObject implements DNSServerIntf{
public DNSServer() throws RemoteException
{        super();        }
public String DNS(String s1) throws RemoteException   {
  if(s1.equals("www.osmania.ac.in"))                return
"50.32.24.29";
  if(s1.equals("www.mvsrec.edu.in"))
return "90.82.44.89";
 if(s1.equals("www.jntu.ac.in"))
return "150.32.64.20";
  if(s1.equals("www.yahoo.com"))
return "88.39.124.129";
else
return "No Info about this address";
}
public static void main(String args[])throws Exception  {
            Registry r=LocateRegistry.createRegistry(1234);
            r.rebind("mvsrserver",new DNSServer());
            System.out.println("server started...");
            }
}
```

**DNSClient.java**

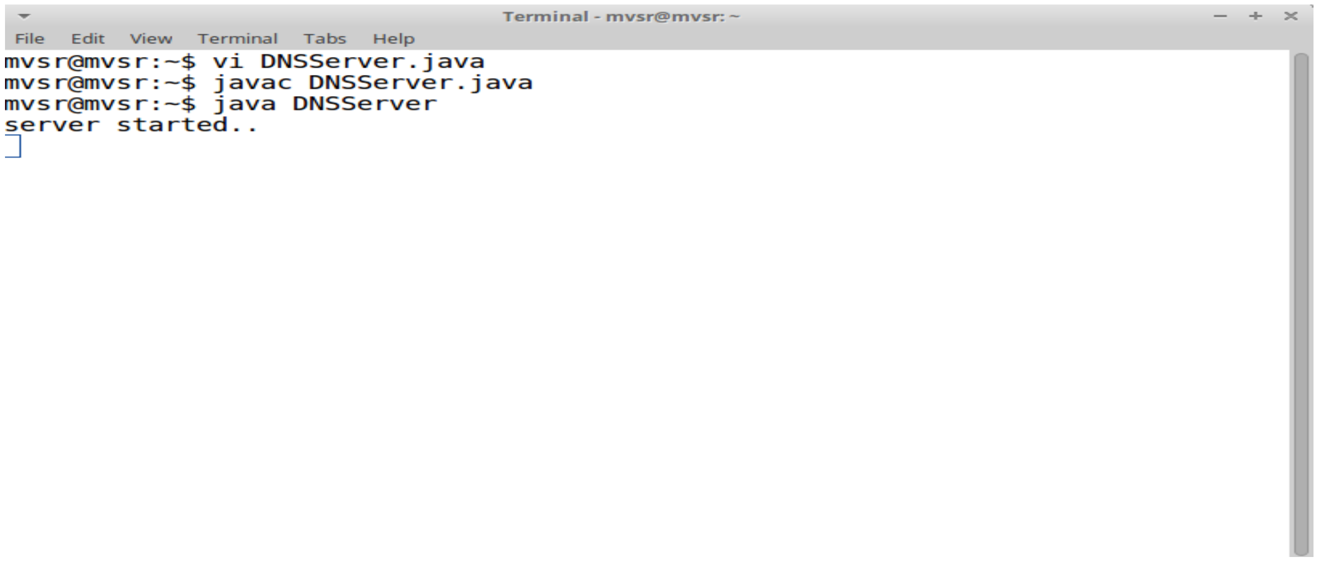
```
import java.rmi.*;
import java.rmi.registry.*;
public class DNSClient{
public static void main(String args[])throws Exception{
                    Registry r=LocateRegistry.getRegistry("localhost",1234);
                    DNSServerIntf d=(DNSServerIntf)r.lookup("mvsrserver");
                       String str=args[0];
                     System.out.println("The website name is:"+str);
                     System.out.println("The IP Address is:"+d.DNS(str));
                                    }
}
```

**DNSServerIntf.java**

```
import java.rmi.*;
public interface DNSServerIntf extends Remote
{
public String DNS(String s1) throws RemoteException;
}
```

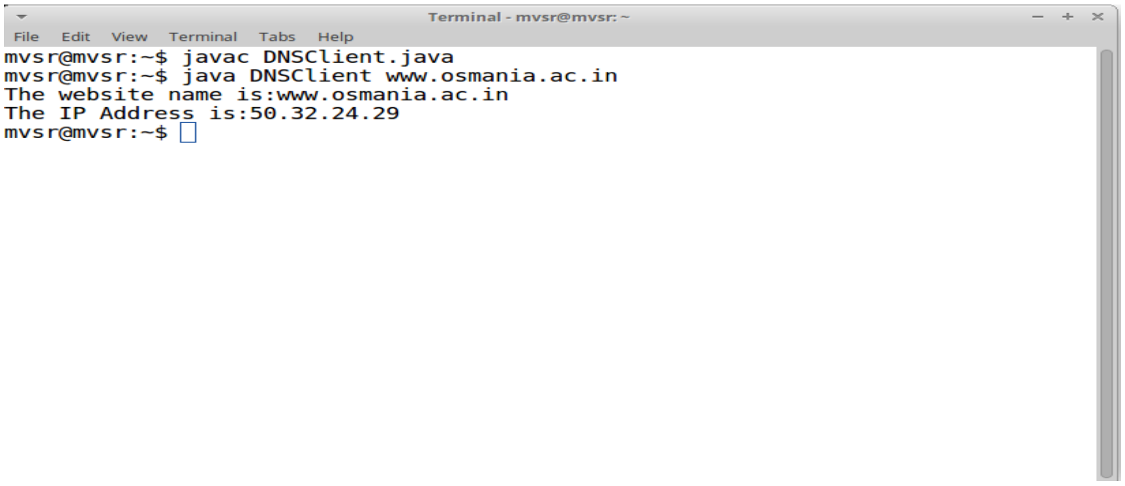
## Sample Outputs:

**DNSServer**



```
mvsr@mvsr:~$ vi DNSServer.java
mvsr@mvsr:~$ javac DNSServer.java
mvsr@mvsr:~$ java DNSServer
server started..
```

**DNSClient**



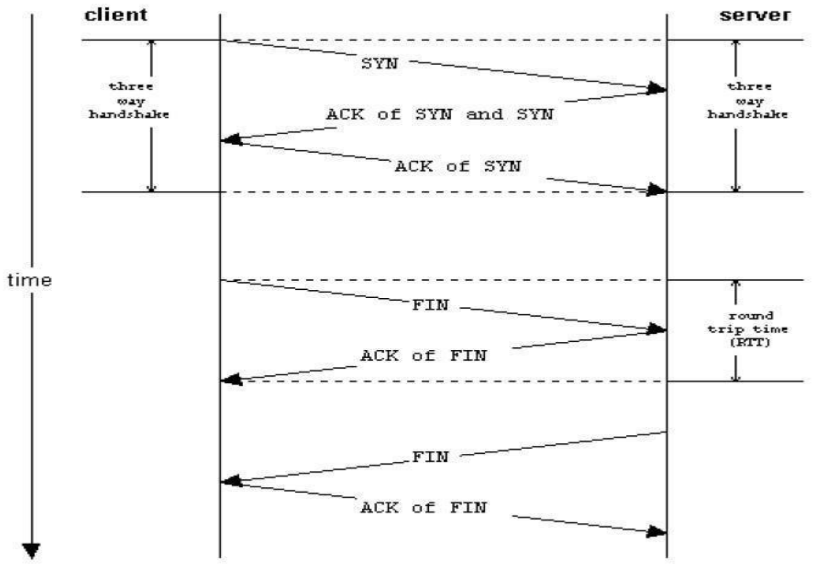
```
mvsr@mvsr:~$ javac DNSClient.java
mvsr@mvsr:~$ java DNSClient www.osmania.ac.in
The website name is:www.osmania.ac.in
The IP Address is:50.32.24.29
mvsr@mvsr:~$ 
```

## 3.Implementing client-server communication using TCP

**Introduction:**

TCP provides the service of exchanging data directly between two network hosts, whereas IP handles addressing and routing message across one or more networks. In particular, TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer.

TCP is the protocol that major Internet applications rely on, applications such as the World Wide Web, email, and file transfer. TCP connection is managed by an operating system through a programming interface that represents the local end-point for communications, the *Internet socket*. During the lifetime of a TCP connection it undergoes a series of state changes:



1. LISTEN: In case of a server, waiting for a connection request from any remote client.
2. SYN-SENT: waiting for the remote peer to send back a TCP segment with the SYN and ACK flags set. (Usually set by TCP clients)
3. SYN-RECEIVED: waiting for the remote peer to send back an acknowledgment after having sent back a connection acknowledgment to the remote peer. (usually set by TCP servers) 4. ESTABLISHED: the port is ready to receive/send data from/to the remote peer.
5. FIN-WAIT-1
6. FIN-WAIT-2
7. CLOSE-WAIT
8. CLOSING
9. LAST-ACK
10. TIME-WAIT: represents waiting for enough time to pass to be sure the remote peer received the acknowledgment of its connection termination request. According to RFC 793 a connection can stay in TIME-WAIT for a maximum of four minutes.
11. CLOSED

TCP uses the notion of port numbers to identify sending and receiving application end-points on a host, or *Internet sockets*. Each side of a TCP connection has an associated 16-bit unsigned port number (0-65535) reserved by the sending or receiving application. Arriving TCP data packets are identified as belonging to a specific TCP connection by its sockets, that is, **the combination of source host address, source port, destination host address, and destination port**. This means that a server computer can provide several clientswith several services simultaneously, as long as a client takes care of initiating any simultaneous connections toone destination port from different source ports.

## TCPServer.java

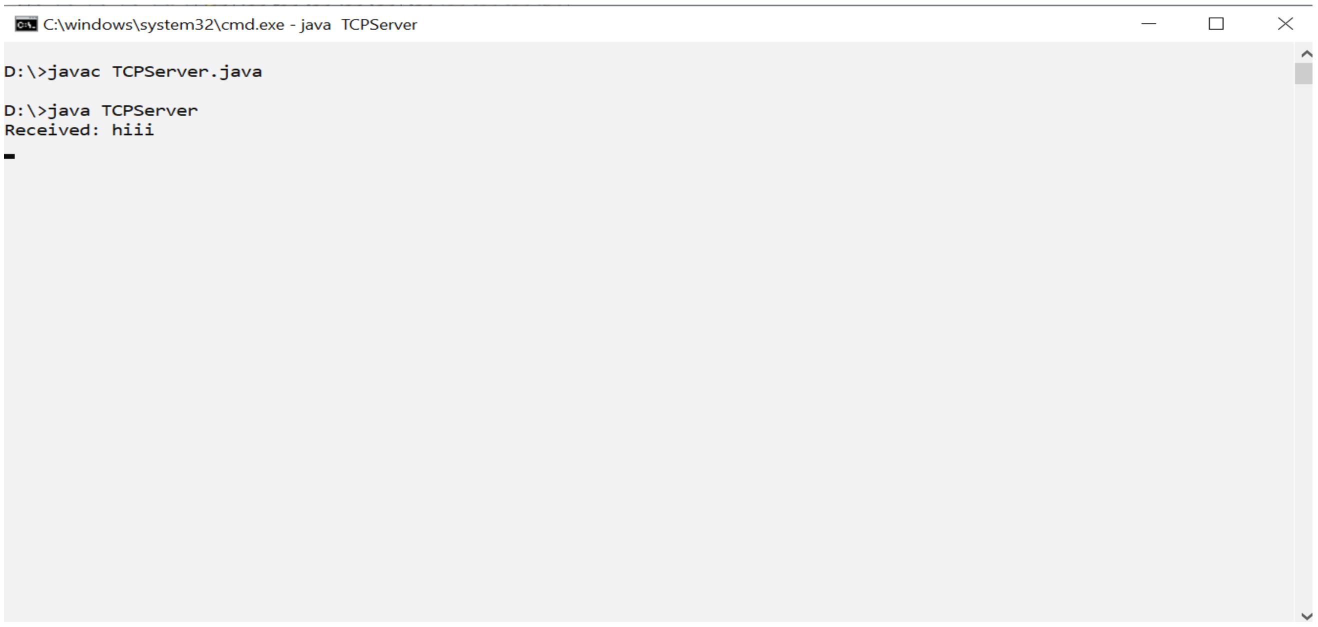
```
import java.net.*;
import java.io.*;
public class TCPServer {
public static void main(String a[]) {
Socket s = null;
try {
int serverPort = 8117;
ServerSocketlistenSocket = new ServerSocket(serverPort);
while (true) {
Socket clientSocket = listenSocket.accept();
Connection c = new Connection(clientSocket);
}
} catch (IOException e) {
System.out.println("Listen:" + e.getMessage());
}
}
}
class Connection extends Thread {
DataInputStreamin ;
DataOutputStreamout;
Socket clientSocket;
public Connection(Socket aClientSocket) {
try {
clientSocket = aClientSocket; in = new DataInputStream(clientSocket.getInputStream());
out = new DataOutputStream(clientSocket.getOutputStream());            this.start();
} catch (IOException e) {
System.out.println("Connection:" + e.getMessage());
}
}
public void run() {
try {
String data = in .readUTF();
out.writeUTF(data);
System.out.println("Received: " + data);
}
catch(EOFException e) {
System.out.println("EOF:" + e.getMessage());
}
catch (IOException e) {
System.out.println("IO:" + e.getMessage());
}

finally {
try {
clientSocket.close();            }
catch (IOException e) {
System.out.println("IO:" + e.getMessage());
}
}
}

}
```

**TCPClient.java**

```
import java.net.*;
import java.io.*;
public class TCPClient {
public static void main(String args[]) { // arguments supply message and hostname
Socket s = null;
try {
int serverPort = 8117;
s = new Socket(args[1], serverPort);
DataInputStream in = new DataInputStream(s.getInputStream());
DataOutputStream out = new DataOutputStream(s.getOutputStream());
out.writeUTF(args[0]); // UTF is a string encoding
String data = in .readUTF(); // read a line of data from the stream
System.out.println("Received: " + data);
} catch (UnknownHostException e) {
System.out.println("Socket:" + e.getMessage());
} catch (EOFException e) {
System.out.println("EOF:" + e.getMessage());
} catch (IOException e) {
System.out.println("readline:" + e.getMessage());
} finally {
if (s != null) try {
s.close();
} catch (IOException e) {
System.out.println("close:" + e.getMessage());
}
}
}

}
```

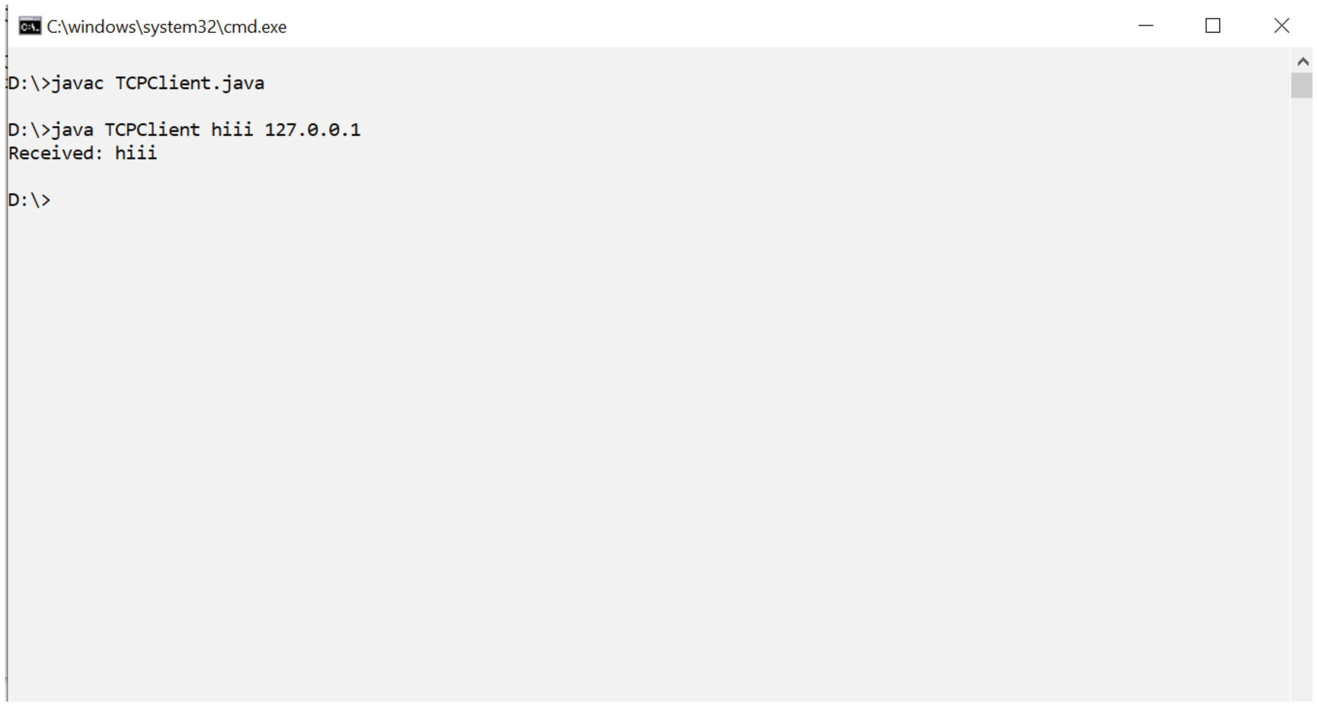
## **<u>Sample Outputs:</u>**

### TCPServer



```
D:\>javac TCPServer.java

D:\>java TCPServer
Received: hiii
```

**TCPClient**



```
D:\>javac TCPClient.java

D:\>java TCPClient hiii 127.0.0.1
Received: hiii

D:\>
```

## 4.Implementation of client-server communication using UDP

**Introduction:**

With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network without requiring prior communications to set up special transmission channels or data paths. UDP uses a simple transmission model without implicit hand-shaking for providing reliability, ordering, or data integrity. UDP's stateless nature is also useful for servers answering small queries from huge numbers of clients. Unlike TCP, UDP is compatible with packet broadcast (sending to all on local network) and multicasting (send to all subscribers).Common network applications that use UDP include the Domain Name System (DNS).

UDP applications use **datagram sockets** to establish host-to-host communications. A **Datagram socket** is a type of Internet socket, which is the sending or receiving point for packet delivery services.[1] Eachpacket sent or received on a Datagram socket is individually addressed and routed. Multiple packets sent fromone machine to another may arrive in any order and might not arrive at the receiving computer.UDP broadcastssends are always enabled on a Datagram Socket. In order to receive broadcast packets a Datagram Socket mustbe bound to a more specific address.

An application binds a socket to its endpoint of data transmission, which is a combination of anIP address and a service port. A port is a software structure that is identified by the port number, a 16 bit integervalue, allowing for port numbers between 0 and 65535.

Port numbers are divided into three ranges:

- Port numbers 0 through 1023 are used for common, well-known services.
- Port numbers 1024 through 49151 are the registered ports
- Ports 49152 through 65535 are dynamic ports that are not officially used for any specific service, and can be used for any purpose. They are also used as ephemeral ports , from which software running on the host may randomly choose a port in order to define itself.[2] In effect, they are used as temporary ports primarily by clients when communicating with servers.

**UDPServer.Java**

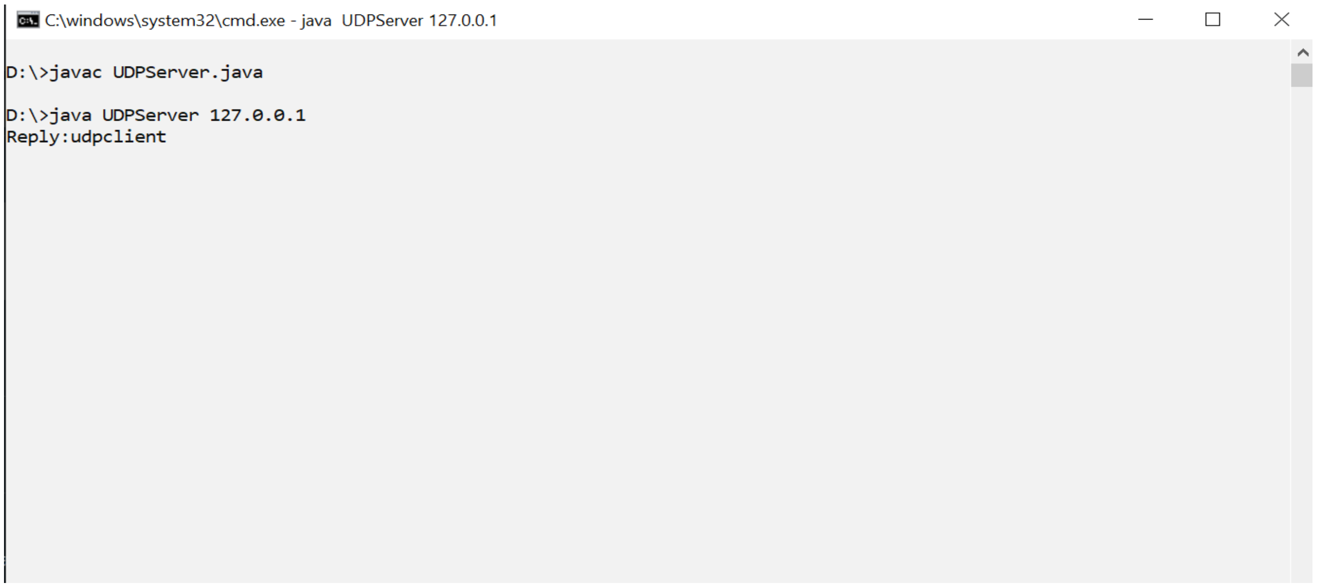
```
import java.net.*;
import java.io.*;
public class UDPServer {
public  static  void  main(String  args[])  {
DatagramSocketaSocket = null;
try {
aSocket = new DatagramSocket(8117);
byte[] buffer = new byte[1000];
while (true) {
DatagramPacket request = new DatagramPacket(buffer, buffer.length);
aSocket.receive(request);
DatagramPacket reply = new DatagramPacket(request.getData(), request.getLength(), request.getAddress(),
request.getPort());
aSocket.send(reply);
System.out.println("Reply:" + new String(reply.getData()));
}
} catch (SocketException e) {
System.out.println("Socket: " + e.getMessage());
} catch (IOException e) {
System.out.println("IO: " + e.getMessage());
}
Finally {
if (aSocket != null) aSocket.close();
}
}
}
```

**UDPClient.java**

```
import java.net.*;
import java.io.*;
public class UDPClient {
public static void main(String args[]) {
DatagramSocketaSocket = null;
try {
aSocket = new DatagramSocket();
byte[] m = args[0].getBytes();
InetAddressaHost =InetAddress.getByName(args[1]);
int serverPort = 8117;
DatagramPacket request = new DatagramPacket(m, args[0].length(), aHost, serverPort);
aSocket.send(request);
byte[] buffer = new byte[1000];
DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
aSocket.receive(reply);
System.out.println("Reply:" + new String(reply.getData()));
} catch (SocketException e) {
System.out.println("Socket:" + e.getMessage());
} catch (IOException e) {
System.out.println("IO:" + e.getMessage());
} finally {             if
(aSocket != null)
aSocket.close();
}
}

}
```

## Sample Outputs:

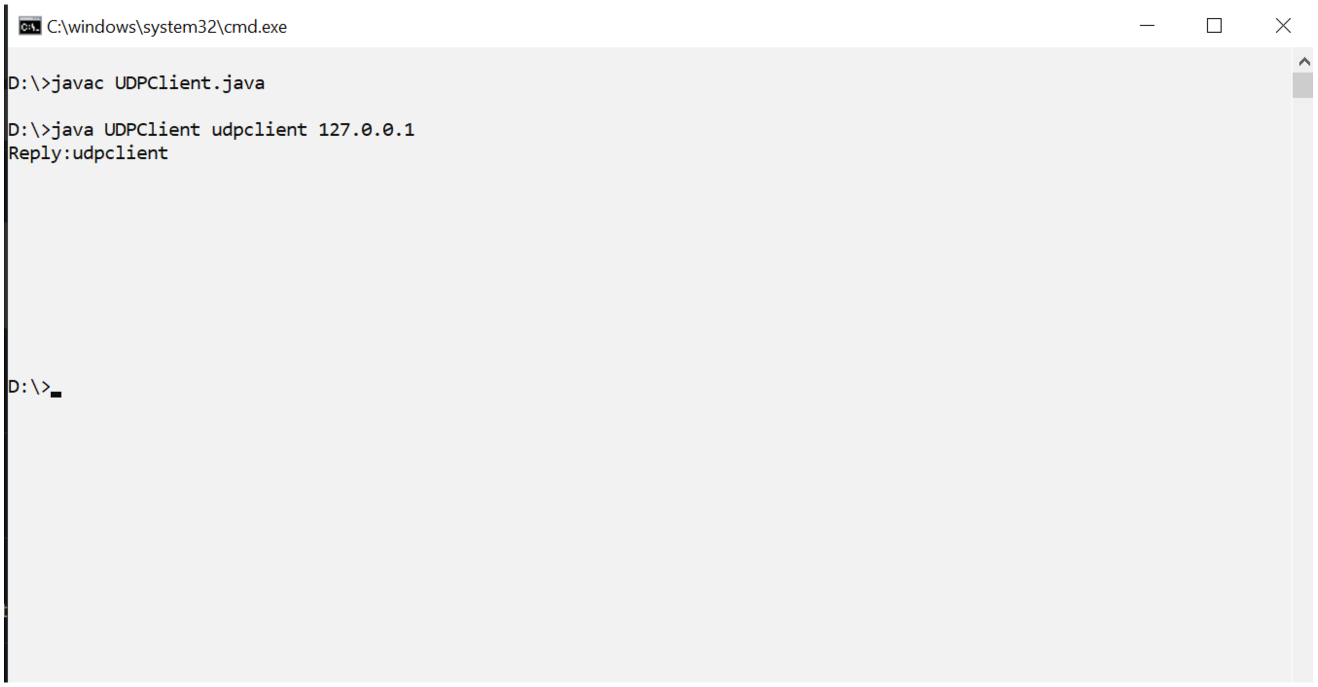
### UDPServer



C:\windows\system32\cmd.exe - java  UDPServer 127.0.0.1

```
D:\>javac UDPServer.java

D:\>java UDPServer 127.0.0.1
Reply:udpclient
```

### UDPClient


C:\windows\system32\cmd.exe

```
D:\>javac UDPClient.java

D:\>java UDPClient udpclient 127.0.0.1
Reply:udpclient






D:\>_
```

## 5.Chat Server Implementation

### Introduction:

Chat server is a standalone application that is made of combination of two-applications, server application (which runs on server side) and client application (which runs on client side). This application is using for chatting in LAN. To start chatting client must be connected with the server after which, messages can be broadcast between each and every client.

### SERVER:

Server side application is used to get the message from any client and broadcast to each and every client. And this application is also used to maintain the list of users and broadcast this list to everyone.

· Firstly server program creates a new server socket by the

**ServerSocket ss = new ServerSocket(Port number);**

- After creating the ServerSocket it accepts the client socket and adds this socket into an arraylist.
- After getting the client socket it creates a thread and enables the DataInputStream for this socket.
- After creating the input stream it reads the user name and adds it to arraylist and this arraylist object writes into the ObjectOutputStream of each client by using an iterator.
- After this process, it creates a new thread and creates one DataInputStream for reading the messages which is sent by the client and after reading the message it creates the DataOutputStream for each socket and writes this message in each client output stream through the iterator. If any client logs out,the server receives the client name and removes it from the arraylist. Further ,it sends this updated arraylist to all clients.

**CLIENT:**

In the client side, first the program creates a new socket and specifies the address and port of the server and establishes the connection with the Server.The client, then, creates a new thread and DataInputStream, ObjectInputStream and DataOutputStream for sending the user name and retrieving the list of all users and adds all the user names into its list box through the iterator.Then new thread is created for sending and receiving the messages from the server. This task is done by using DataInputStream and DataOutputStream.When the client logs out it sends its’ name and message i.e. “User_Name has Logged out” and terminates the chatting.

**ChatServer.java**i

```
mport java.net.*;

import java.io.*;
public class ChatServer implements Runnable {
private ChatServerThread clients[] = new ChatServerThread[50];
private ServerSocketserver = null;
private Thread thread = null;
private int clientCount = 0;
public ChatServer(int port) {
try {
System.out.println("binding to port" + port + ",please wait ...");
server = new ServerSocket(port);
System.out.println("server started:" + server);
thread = new Thread(this);
thread.start();
} catch (IOException e) {
System.out.println("cannot bind to port" + port + ":" + e.getMessage());
}
}
public void run() {
while (thread != null) {
try {
System.out.println("waiting for a client...");
addThread(server.accept());
} catch (IOException e) {
System.out.println("server accept error:" + e);
if (thread != null) {
thread.stop();
thread = null;
}
}
}
}
public void stop() {
```

```
if (thread != null) {             thread.stop();
thread = null;
}
}
private int findClient(int ID) {
for (int i = 0; i<clientCount; i++)
if (clients[i].getID() == ID)
return i;
return -1;
}
public synchronized void handle(int ID, String input) {
if (input.equals("quit")) {
clients[findClient(ID)].send("quit");
remove(ID);
} else
System.out.println(ID + ":" + input);         for
(int i = 0; i<clientCount; i++)
clients[i].send(ID + ":" + input);
}
public synchronized void remove(int ID) {
        int pos = findClient(ID);
if (pos>= 0) {
ChatServerThread closing = clients[pos];
System.out.println("removing client thread:" + ID + "at" + pos);             if
(pos<clientCount - 1)
for (int i = pos + 1; i<clientCount; i++)
clients[i - 1] = clients[i];
clientCount--;
try {
closing.close();
} catch (IOException e) {
System.out.println("error closing thread;" + e);
}
closing.stop();
}
}
private void addThread(Socket
socket) {
if (clientCount<clients.length) {
System.out.println("client      accepted:"      +      socket);
clients[clientCount]    =    new    ChatServerThread(this,
socket);
try {
clients[clientCount].open();
clients[clientCount].start();
clientCount++;
} catch (IOException e) {
System.out.println("error opening thread;" + e);
}
} else
System.out.println("client refused;maximum" + clients.length + "reached");
}
public static void main(String a[]) {
```

```
ChatServer server = null;
if (a.length != 1)
System.out.println("Usage:javaChatServer Port");          else
server = new ChatServer(Integer.parseInt(a[0]));
}
}
```

**<u>ChatServerThread.java</u>**

```
import java.net.*;
import java.io.*;
public class ChatServerThread extends Thread {
private ChatServer server = null;
private Socket socket = null;
private DataInputStream In = null;     private
int ID = -1;
private PrintStream Out = null;
public ChatServerThread(ChatServerserv, Socket sock) {
super();
server = serv;
socket = sock;
ID = socket.getPort();
}
public void send(String msg) {
Out.println(msg);
        Out.flush();
}
public int getID() {
return ID;
}
public void run() {
System.out.println("Server thread" + ID + "running");
while (true) {
try {
server.handle(ID, In.readLine());
} catch (IOException e) {
System.out.println(ID + "error reading" + e.getMessage());
server.remove(ID);
stop();
}
}
}
public void open() throws IOException {
In = new DataInputStream(socket.getInputStream());
Out = new PrintStream(socket.getOutputStream());
}
public void close() throws IOException {
if (socket != null)
socket.close();
if (In != null)
In.close();
if (Out != null)
Out.close();
}
}
```

**ChatClient.java**

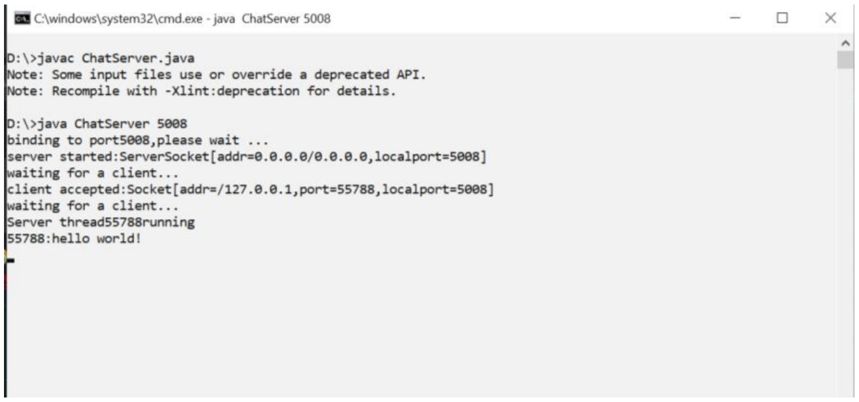
```
import java.net.*;
import java.io.*;
public class ChatClient implements Runnable {
private ChatClientThread client = null;
    private Socket socket = null;
private DataInputStream console = null;
    private Thread thread = null;
private PrintStream Out = null;
public ChatClient(String serverName, int serverPort) {
System.out.println("Establishing connection please wait...");
try {
socket = new Socket(serverName, serverPort);
System.out.println("connected" + socket.toString());
console = new DataInputStream(System.in);
Out = new PrintStream(socket.getOutputStream());              if
(thread == null) {
client = new ChatClientThread(this, socket);
thread = new Thread(this);
thread.start();
}
} catch (UnknownHostException e) {
System.out.println("Host unknown" + e.getMessage());
} catch (IOExceptionioe) {
System.out.println("Unexcepted exception" + ioe.getMessage()); }
}
public void run() {
while (thread != null) {
try {
Out.println(console.readLine());
Out.flush();
} catch (IOException e) {
System.out.println("sending error" + e.getMessage());
stop();
}
}
}
public void handle(String msg) {
if (msg.equals("quit")) {
System.out.println("good bye, press RETURN to exit...");
stop();
} else
System.out.println(msg);
}
public void stop() {
if (thread != null) {
thread.stop();
thread = null;
}
try {
 if (console != null)
console.close();             if
(Out != null)
Out.close();
if (socket != null)
socket.close();
} catch (IOException e) {
System.out.println("error closing...");
}   client.close();
client.stop();
}
public static void main(String args[]) {
ChatClient client = null;
if (args.length != 2)
System.out.println("usage:javaChatClient host port");
else
client = new ChatClient(args[0], Integer.parseInt(args[1]));
}
}
```

**ChatClientThread.java**

```
import java.net.*;
import java.io.*;
public class ChatClientThread extends Thread {
private ChatClient client = null;
private Socket socket = null;
private DataInputStream In = null;
public ChatClientThread(ChatClient cli, Socket sock) {
client = cli;
        socket = sock;
try {
In = new DataInputStream(socket.getInputStream());
} catch (IOException e) {
System.out.println("error geting input stream:" + e);
client.stop();
}        start();
}
public void close() {
try {
if (In != null)
In.close();
} catch (IOException e) {
System.out.println("error closing input stream:" + e);
}
}
public void run() {
while (true) {
try {
client.handle(In.readLine());
} catch (IOException e) {
System.out.println("Listening error" + e.getMessage());
client.stop();
}
}
}
}
```

## Sample Outputs:

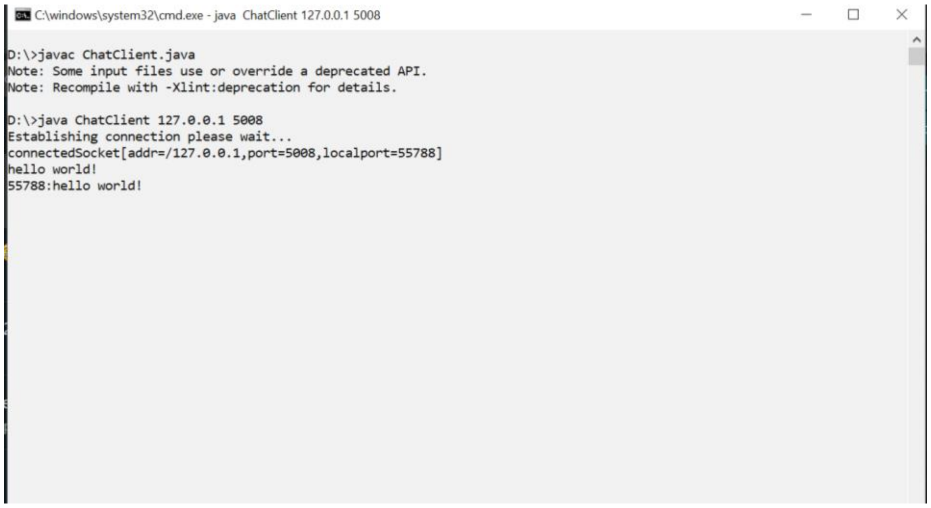
### ChatServer



C:\windows\system32\cmd.exe - java  ChatServer 5008

```
D:\>javac ChatServer.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\>java ChatServer 5008
binding to port5008,please wait ...
server started:ServerSocket[addr=0.0.0.0/0.0.0.0,localport=5008]
waiting for a client...
client accepted:Socket[addr=/127.0.0.1,port=55788,localport=5008]
waiting for a client...
Server thread55788running
55788:hello world!
```

### ChatClient



C:\windows\system32\cmd.exe - java  ChatClient 127.0.0.1 5008

```
D:\>javac ChatClient.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\>java ChatClient 127.0.0.1 5008
Establishing connection please wait...
connectedSocket[addr=/127.0.0.1,port=5008,localport=55788]
hello world!
55788:hello world!
```

## 6. FTP Implementation

**ftp.java**

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

```
public class ftp {
static Socket DataSocket;
static DataInputStreamipstream;
static PrintStream outstream;
public static void main(String args[]) throws IOException {
try {
BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the FTP host address:");
String host = br.readLine();
DataSocket = new Socket(host, 21);
ipstream = new DataInputStream(DataSocket.getInputStream());
outstream = new PrintStream(DataSocket.getOutputStream());
String s = ipstream.readLine();
if (s.startsWith("220")) {
System.out.println("Connected to server " + host);
System.out.println("\n Enter Username:\t");
String uname = br.readLine();
outstream.print("USER" + uname + "\r\n");
s = ipstream.readLine();
if (s.startsWith("331")) {
System.out.println("\n Enter password:\t");
String pass = br.readLine();
outstream.print("PASS" + pass + "\r\n");
s = ipstream.readLine();
if (s.startsWith("230")) {
System.out.println("Login successful!!!HAVE A NICE DAY!!!\n");
while (true) {
System.out.println("Press Enter to continue . . . ");
String name = br.readLine();
System.out.println("Enter your choice: \n");
System.out.println("1. Read a file \n");
System.out.println("2. Store a file \n");
System.out.println("3. List files \n");
System.out.println("4. Change directory \n");
System.out.println("5. Change to Parent Directory \n");
System.out.println("6. Create Directory \n");
System.out.println("7. Print Current Directory \n");
System.out.println("8. Logout \n");
int option = Integer.parseInt(br.readLine());
switch (option) {
case 1:
read();
break;

case 2:
store();
break;
case 3:
list();
break;
case 4:
System.out.println("Enter Directory: \n");
name = br.readLine();
outstream.print("CWD " + name + "\r\n");
break;
case 5:
outstream.print("CDUP" + "\r\n");
break;
case 6:
System.out.println("Enter Directory: \n");
name = br.readLine();
outstream.print("MKD " + name + "\r\n");
break;
case 7:
outstream.print("PWD" + "\r\n");
break;
case 8:
outstream.print("QUIT" + "\r\n");
System.exit(1);
break;
default:
System.out.println("Choose a valid option!!!\n");
break;
}
}
}
}
} else {
System.out.println("Error connecting to server" + host);
}
} catch (UnknownHostException e) {
System.err.println(e);
} catch (IOException e) {
System.err.println(e);
}
}
static void store() {
try {
BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter Filename: \n");
String filename = br.readLine();
FileInputStreaminputStream = new FileInputStream(filename);
outstream.print("PASV" + "\r\n");
String s = ipstream.readLine();
String ip = null;
int port = -1;
int opening = s.indexOf('(');
int closing = s.indexOf(')', opening + 1);
if (closing > 0) {
String dataLink = s.substring(opening + 1, closing);
```

```
StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "." + tokenizer.nextToken() +
"." + tokenizer.nextToken();
port =Integer.parseInt(tokenizer.nextToken());
outstream.print("STOR" + filename + "\r\n");
Socket dataSocket = new Socket(ip, port);
s = ipstream.readLine();
if (!s.startsWith("150")) {
throw new IOException("Simple FTP was not allowed to send the file:" + s);
}
BufferedInputStream input = new BufferedInputStream(inputStream);
BufferedOutputStream output = new
BufferedOutputStream(DataSocket.getOutputStream());
byte[] buffer = new byte[4096];
int bytesRead = 0;
while ((bytesRead = input.read(buffer)) != -1) {
output.write(buffer, 0, bytesRead);
}
output.flush();
output.close();
input.close();
}
} catch (UnknownHostException e) {
System.err.println(e);
} catch (IOException e) {
System.err.println(e);
}
}
static void read() {
try {
BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter Filename \n");
String filename = br.readLine();
outstream.print("PASV" + "\r\n");
String s = ipstream.readLine();
System.out.println(s);
String ip = null;
int port = -1;
int opening = s.indexOf('(');
int closing = s.indexOf(')', opening + 1);
if (closing > 0) {
String dataLink = s.substring(opening + 1, closing);
StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "." + tokenizer.nextToken() +
"." + tokenizer.nextToken();
port = Integer.parseInt(tokenizer.nextToken()) * 256 +
Integer.parseInt(tokenizer.nextToken());
outstream.print("RETR" + filename + "\r\n");
Socket dataSocket = new Socket(ip, port);
s = ipstream.readLine();
```

```
if (!s.startsWith("150")) {
throw new IOException("Simple FTP was not allowed to send the file:" + s);
}
BufferedInputStream input = new BufferedInputStream(dataSocket.getInputStream());
FileOutputStream output = new FileOutputStream("New" + filename);
byte[] buffer = new byte[4096];
int bytesRead = 0;
while ((bytesRead = input.read(buffer)) != -1) {
output.write(buffer, 0, bytesRead);
}
s = ipstream.readLine();
System.out.println("Finished!" + s);
output.close();
input.close();
}
} catch (UnknownHostException e) {
System.err.println(e);
} catch (IOException e) {
System.err.println(e);
}
}
static void list() {
try {
outstream.print("PASV" + "\r\n");
String s = ipstream.readLine();
System.out.println(s);
String ip = null;
int port = -1;
int opening = s.indexOf('(');
int closing = s.indexOf(')', opening + 1);
if (closing > 0) {
String dataLink = s.substring(opening + 1, closing);
StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "." + tokenizer.nextToken() +
"." + tokenizer.nextToken();
port = Integer.parseInt(tokenizer.nextToken()) * 256 +
Integer.parseInt(tokenizer.nextToken());
outstream.print("LIST" + "\r\n");
Socket dataSocket = new Socket(ip, port);
s = ipstream.readLine();
DataInputStream input = new DataInputStream(dataSocket.getInputStream());
String line;
while ((line = input.readLine()) != null)
System.out.println(line);
input.close();
}
} catch (UnknownHostException e) {
System.err.println(e);
} catch (IOException e) {
System.err.println(e);
```

```
}
}
}
```

## Steps of Execution:

1. After compiling the java program open a new terminal and type the commands

- sudo /opt/lamp/manager*

  $\rightarrow$ Password :mvsr

2. It connects to the lampp.

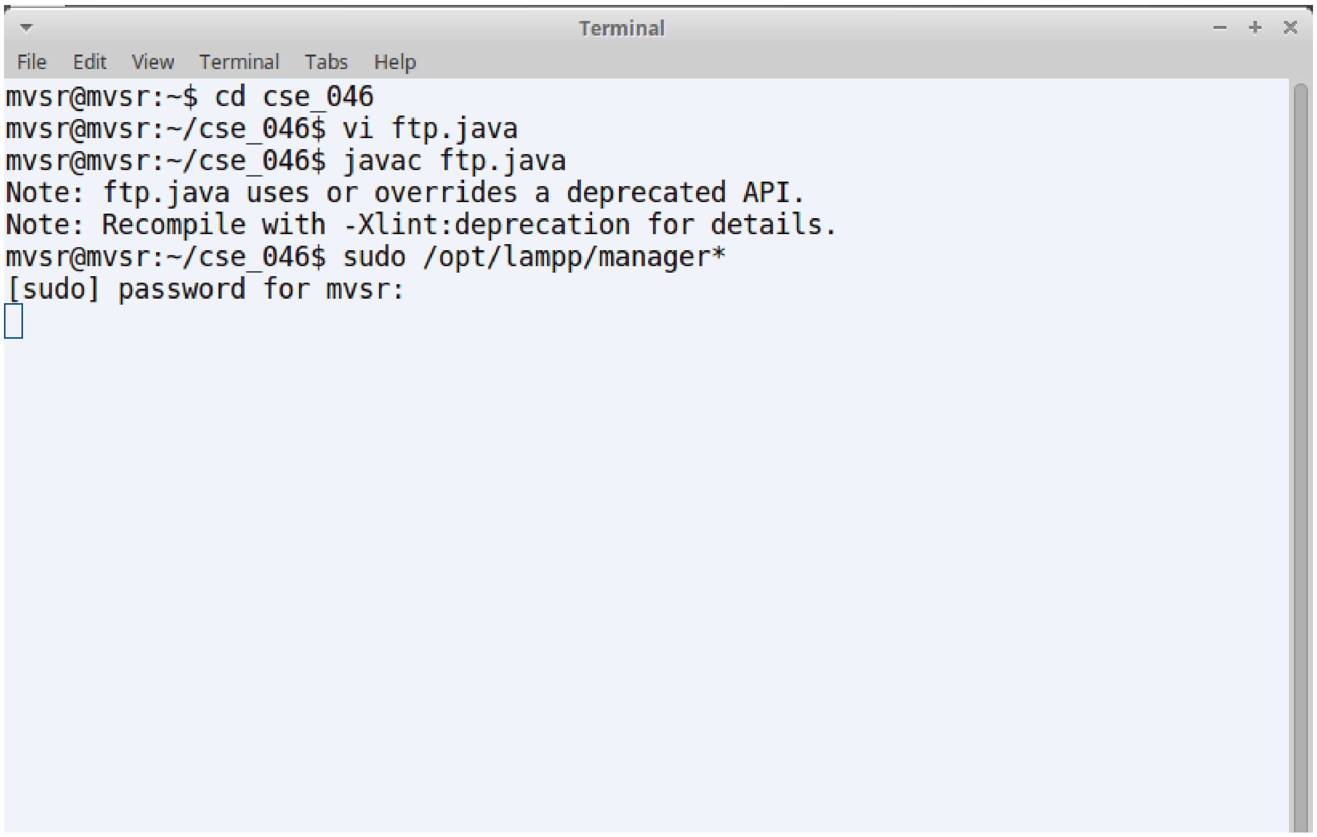
3.Select Manage servers and ProFTPD

4.Click start.

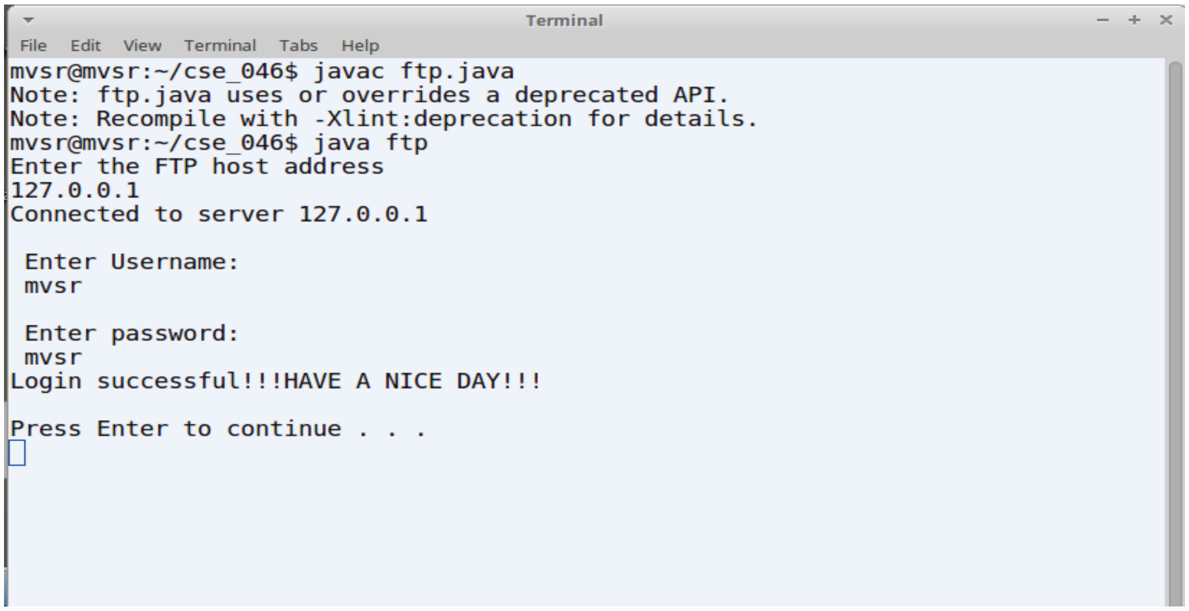
5.Now run the java program and enter FTP host address as loop back address i.e. 127.0.0.1

And rest of the steps as mentioned in the below picture.

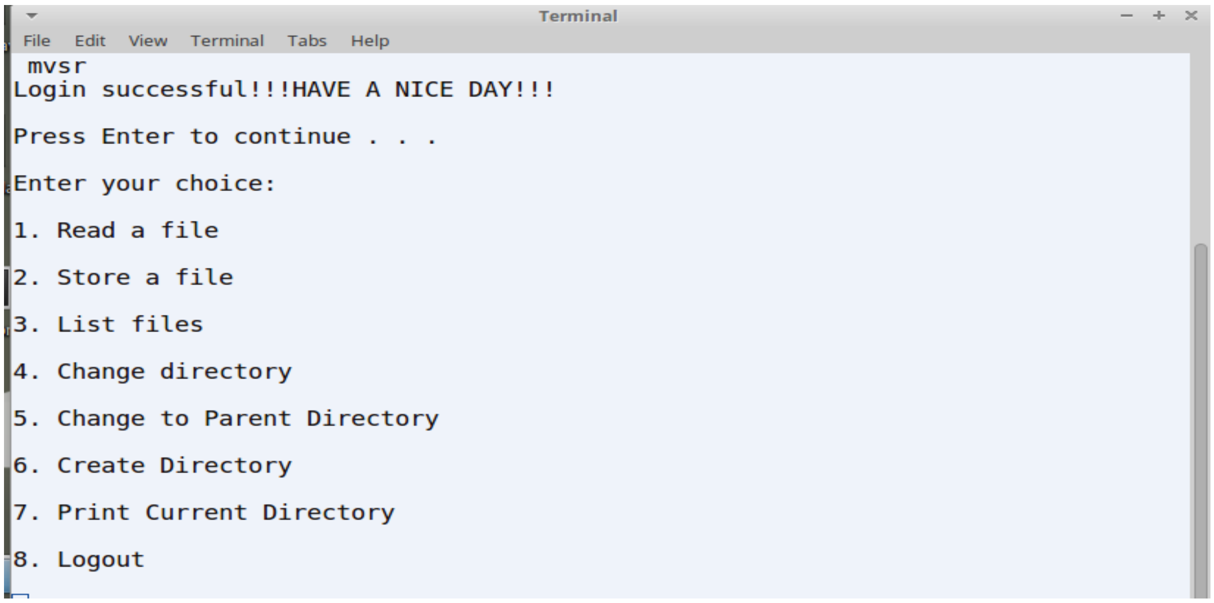
**OUTPUT:**



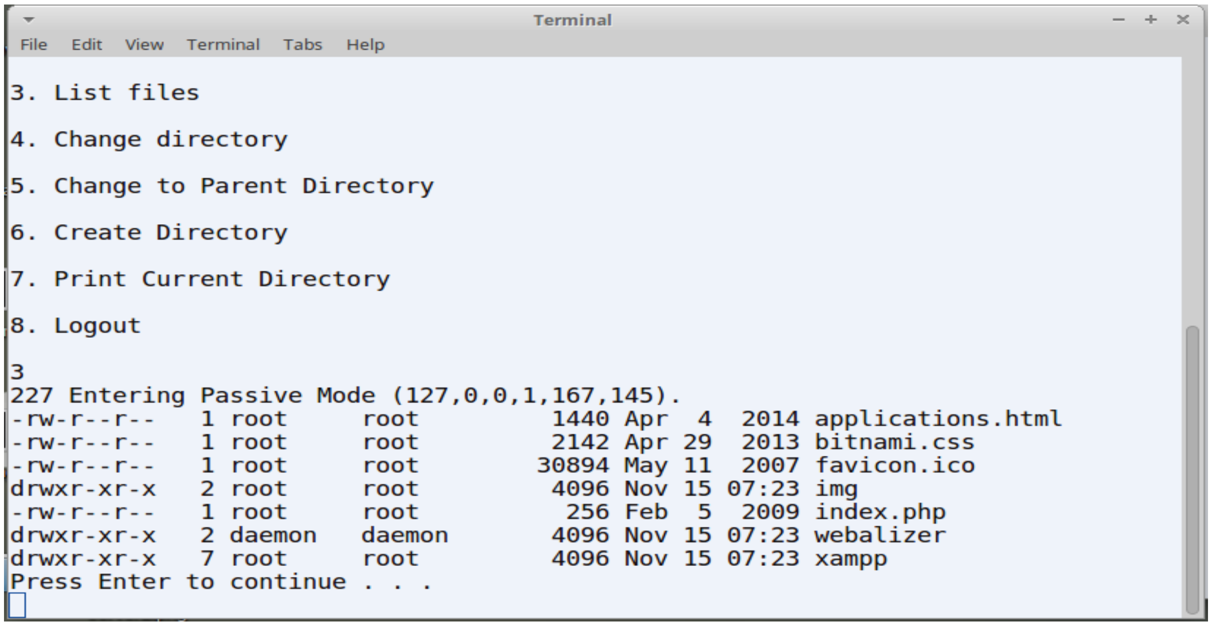
```
mvsr@mvsr:~$ cd cse_046
mvsr@mvsr:~/cse_046$ vi ftp.java
mvsr@mvsr:~/cse_046$ javac ftp.java
Note: ftp.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
mvsr@mvsr:~/cse_046$ sudo /opt/lampp/manager*
[sudo] password for mvsr:
```



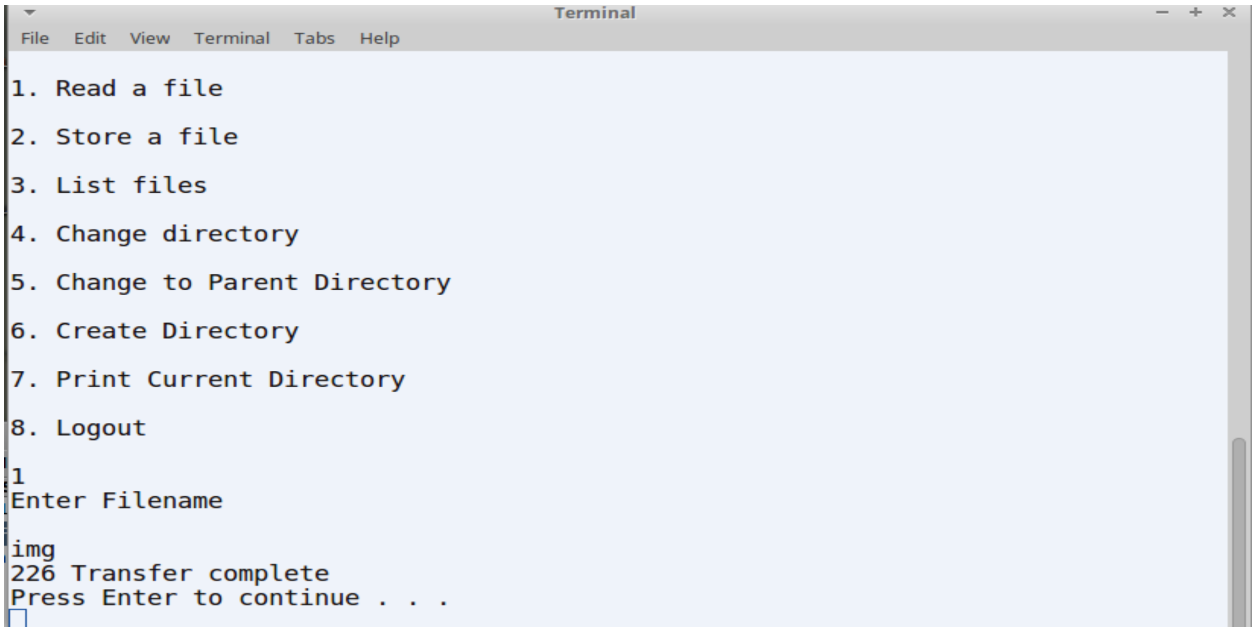
```
mvsr@mvsr:~/cse_046$ javac ftp.java
Note: ftp.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
mvsr@mvsr:~/cse_046$ java ftp
Enter the FTP host address
127.0.0.1
Connected to server 127.0.0.1

 Enter Username:
 mvsr

 Enter password:
 mvsr
Login successful!!!HAVE A NICE DAY!!!

Press Enter to continue . . .
```



```
 mvsr
Login successful!!!HAVE A NICE DAY!!!

Press Enter to continue . . .

Enter your choice:

1. Read a file

2. Store a file

3. List files

4. Change directory

5. Change to Parent Directory

6. Create Directory

7. Print Current Directory

8. Logout
```



```
3. List files

4. Change directory

5. Change to Parent Directory

6. Create Directory

7. Print Current Directory

8. Logout

3
227 Entering Passive Mode (127,0,0,1,167,145).
-rw-r--r--   1 root     root         1440 Apr  4  2014 applications.html
-rw-r--r--   1 root     root         2142 Apr 29  2013 bitnami.css
-rw-r--r--   1 root     root        30894 May 11  2007 favicon.ico
drwxr-xr-x   2 root     root         4096 Nov 15 07:23 img
-rw-r--r--   1 root     root          256 Feb  5  2009 index.php
drwxr-xr-x   2 daemon   daemon       4096 Nov 15 07:23 webalizer
drwxr-xr-x   7 root     root         4096 Nov 15 07:23 xampp
Press Enter to continue . . .
```



```
1. Read a file

2. Store a file

3. List files

4. Change directory

5. Change to Parent Directory

6. Create Directory

7. Print Current Directory

8. Logout

1
Enter Filename

img
226 Transfer complete
Press Enter to continue . . .
```

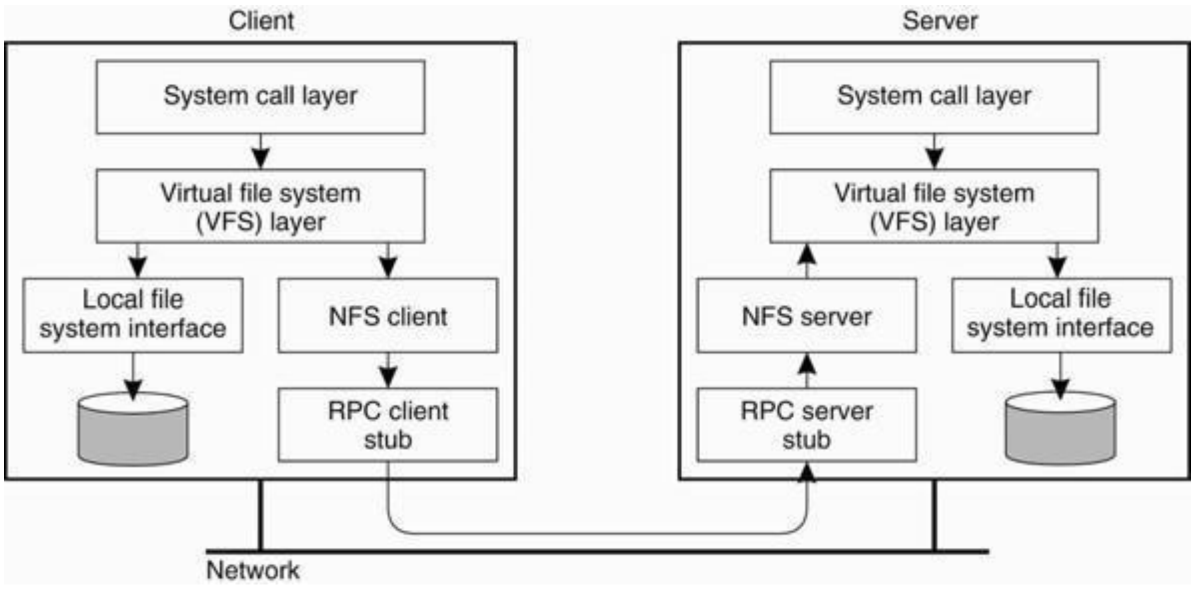
## 7.Understanding Working of NFS

NFS is a protocol for remote access to a file system developed by Sun
– It does not implement a file system per se
– Remote access is transparent to applications
• File system and OS independent
• Client/server architecture
– Client file system requests are forwarded to a remote server; NFS follows the remote
access model – Requests are implemented as remote procedure calls (RPCs)

Servers:
File sharing Server's:
**1. NFS: Network File System:** These servers are used in LAN
**2. FTP: File Transfer Protocol:** These servers are used in WAN & LAN
**3. Samba Server:** These servers are used in Linux & Window heterogeneous networks.

Note: 1. NFS is used in LAN because bandwidth for NFS is more than compared with FTP and Samba.
2. STP is used in WAN because FTP requires very less bandwidth when compared to NFS.

NFS (Network File System):-
**Configuration Steps:**
**1)** Copy all the shared files to a single location
/nfssh
are aa
bb cc
**2)** Create new file inside the share directory aa bb cc **3)** Packages:
i) nfs-utils: to un-install the nfs-utils 60
ii) portmap (please do not uninstall portmap
**4)**
services
nfsport
map**5)**
Daemon
s:
nfsdq
uotad
mount
d
6) Main Configuration File: /etc/exports


Client
Server
System call layer
Virtual file system (VFS) layer
Local file system interface
NFS client
RPC client stub
System call layer
Virtual file system (VFS) layer
NFS server
Local file system interface
RPC server stub
Network

**Experiment:**

Make a directory named nfsshare with files aa bb cc:

```
#mkdir /nfsshare
#cd /nfsshare
#touch aa bb cc
#ls
aa
bb
cc
```

Check if the nfs-utils package is installed or not:

#rpm –q nfs-utils (Prints a message whether package is installed or not)

#rpm –q portmap

To reinstall the package first remove it with the following commands:

#servicesnfs stop

#rpm –e nfs-utils

#rm –rf /var/lib/nfs/xtab-------- remove

If package is not installed then there are two ways to install:

61

1. Download from FTP and install

#ping the server

#rpm –ivhftp://192.168.0.250:/pub.RedHat/RPMS/nfs* --force –aid

2. Install from CD

```
#mount /dev/cdrom/mnt
#cd /mnt
#ls
#cd Fedora
```

```
#cd RPMS
#rpm –ivhnfs-utils* --force –aid
#rpm –ivhportmap* --force –aid
```

After installing the nfs-utils package create file as below:

```
#vi /etc/exports
/var/ftp/pub 192.168.0.0./24(ro,sync)
/nfsshare 192.168.0.0.24(rw,sync)
```

Note:in vi-editor write this content (/nfsshare server ipaddress and no. of systems that are connected in network)

After installing services enter the command to restart

```
#servicesnfs restart
```

Note: execute this command twice because first it will show failed second time it will show ok.

Access to NFS share from client:

```
#mount –t nfs 10.10.12.114:/nfsshare/mnt
```

Note: In client machine enter Server Ipaddress

**Note:**

1) #ping 192.168.0.3 –b

Broadcasts the address in the network only.

2) #ssh 192.168.0.8

Connects the PC to another PC just like Terminal connection in Windows.

3) In NFS all files & Directory are by default in read only mode.

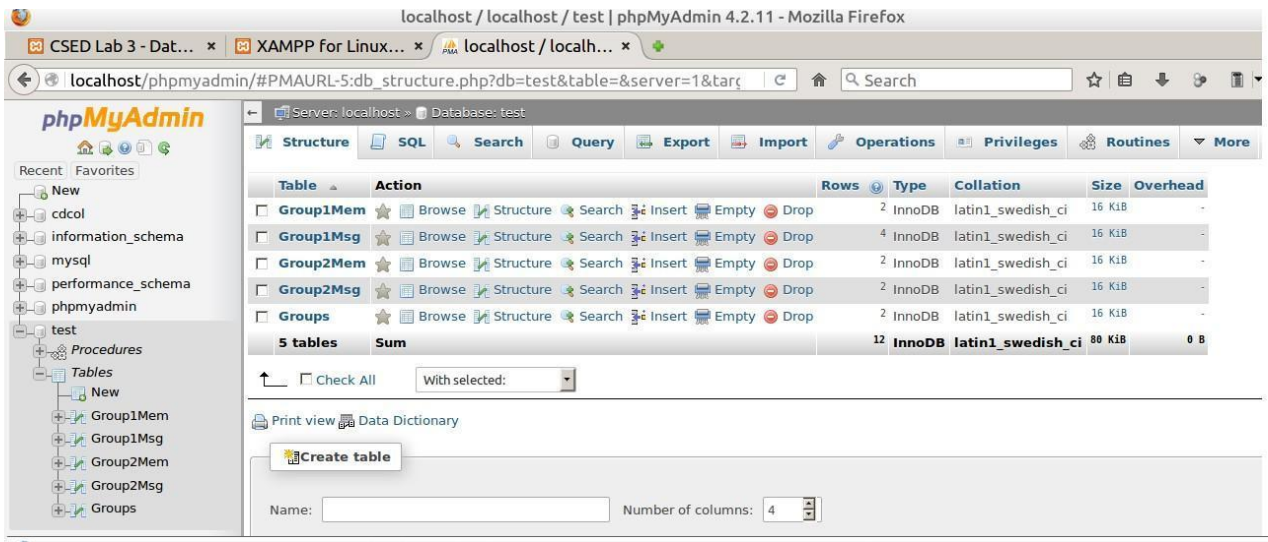
Common KVM Switch: Using a KVM switch a monitor, keyboard, and a mouse can be connected to two computers.

## 8.Implementation of Bulletin Board

```
import java.io.*;
import java.sql.*;
import java.sql.Connection;
public class bulletin {
static Connection con;
static Statement st;
static ResultSetrs;
public static void main(String[] args) throws IOException {
try {
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/BulletinBoard", "root","");
st = con.createStatement();
}
catch (ClassNotFoundException ex) {
ex.printStackTrace();
} catch (SQLException ex) {
ex.printStackTrace();
```

```
}
if (con != null) {
System.out.println("Connected to server...\n");
System.out.println("please Enter your group name:\n");
try {
rs = st.executeQuery("select*from Groups");
while (rs.next()) {
System.out.println(rs.getString("Group"));
}
} catch (SQLException ex) {
ex.printStackTrace();
}
String grpname;
BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));
grpname = br.readLine();
System.out.print("Enter username:\t");
String usr = br.readLine();
System.out.println("\nenter password:\t");
String pass = br.readLine();
try {
String query = "select * from " + grpname.trim() + "Mem where User=\"" + usr + "\"";
System.out.println(query);
rs = st.executeQuery(query);
while (rs.next()) {
if (rs.getString("password").equals(pass)) {
System.out.println("login successfull!Have A NICE DAY!!!");
while (true) {
System.out.println("enter your choice:\n");
System.out.println("1.read message:\n");
System.out.println("2.write message:\n");
System.out.println("3.logout:\n");
int option = Integer.parseInt(br.readLine());
switch (option) {
case 1: query = "select * from " + grpname + "Msg";
System.out.println(query);
rs = st.executeQuery(query);
while (rs.next())
System.out.println(rs.getString("user") + ":" + rs.getString("message") + "\n\n");
break;
case 2: System.out.println("enter your message :\n");
String msg = br.readLine();
query = "insert into " + grpname + "Msgvalues('" + msg + "','" + usr + "')";
System.out.println(query);
st.executeUpdate(query);
break;
case 3: System.exit(1);
default: System.out.println("choose a valid option!!\n");
break;
}
}
} else {
System.out.println("please login again\n");
System.exit(1);
}
}
} catch (SQLException ex) {
ex.printStackTrace();
}
}
}
}
```

**OUTPUT:**


localhost / localhost / test | phpMyAdmin 4.2.11 - Mozilla Firefox

localhost/phpmyadmin/#PMAURL-5:db_structure.php?db=test&table=&server=1&tarç

Server: localhost » Database: test

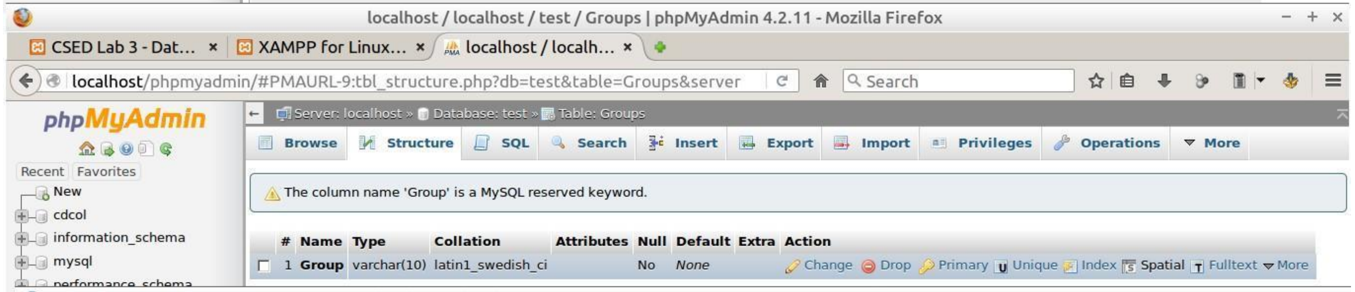
| Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|
| Group1Mem | Browse Structure Search Insert Empty Drop | 2 | InnoDB | latin1_swedish_ci | 16 KiB | - |
| Group1Msg | Browse Structure Search Insert Empty Drop | 4 | InnoDB | latin1_swedish_ci | 16 KiB | - |
| Group2Mem | Browse Structure Search Insert Empty Drop | 2 | InnoDB | latin1_swedish_ci | 16 KiB | - |
| Group2Msg | Browse Structure Search Insert Empty Drop | 2 | InnoDB | latin1_swedish_ci | 16 KiB | - |
| Groups | Browse Structure Search Insert Empty Drop | 2 | InnoDB | latin1_swedish_ci | 16 KiB | - |
| 5 tables | Sum | 12 | InnoDB | latin1_swedish_ci | 80 KiB | 0 B |

Check All    With selected:

Print view  Data Dictionary

Create table

Name:    Number of columns: 4

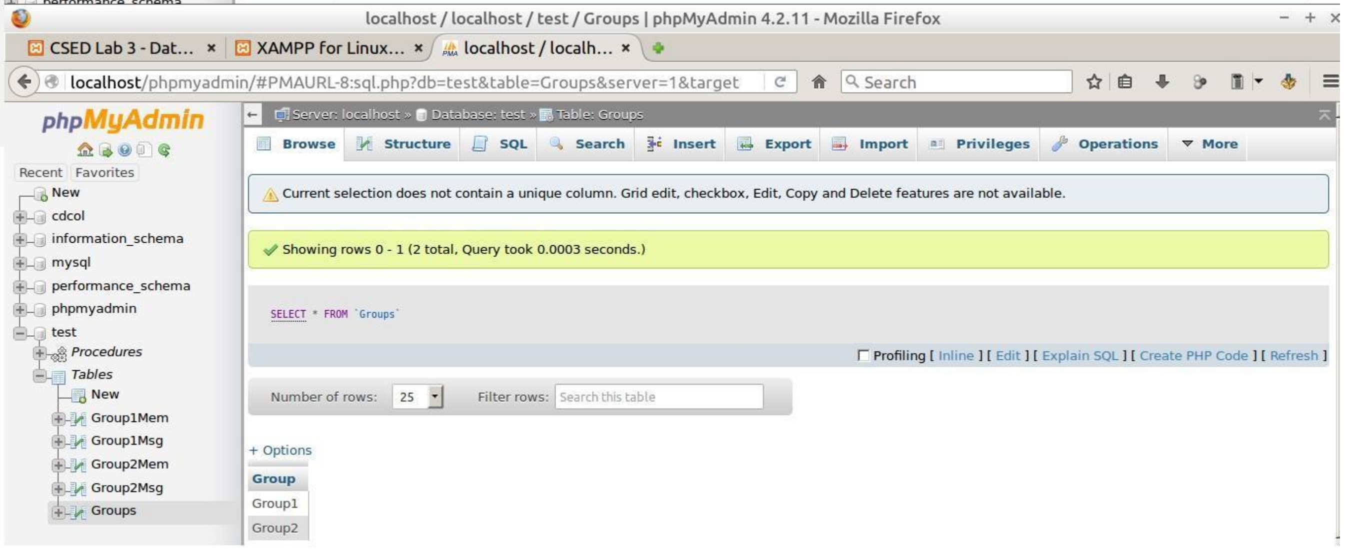

localhost / localhost / test / Groups | phpMyAdmin 4.2.11 - Mozilla Firefox

localhost/phpmyadmin/#PMAURL-9:tbl_structure.php?db=test&table=Groups&server

Server: localhost » Database: test » Table: Groups

The column name 'Group' is a MySQL reserved keyword.

| # | Name | Type | Collation | Attributes | Null | Default | Extra | Action |
|---|---|---|---|---|---|---|---|---|
| 1 | Group | varchar(10) | latin1_swedish_ci |  | No | None |  | Change Drop Primary Unique Index Spatial Fulltext More |


localhost / localhost / test / Groups | phpMyAdmin 4.2.11 - Mozilla Firefox

localhost/phpmyadmin/#PMAURL-8:sql.php?db=test&table=Groups&server=1&target

Server: localhost » Database: test » Table: Groups

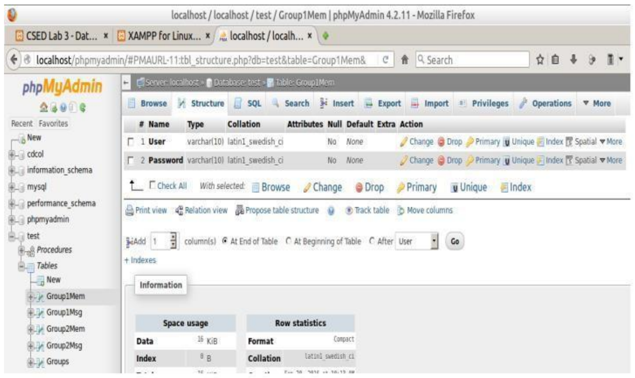
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)

```
SELECT * FROM `Groups`
```

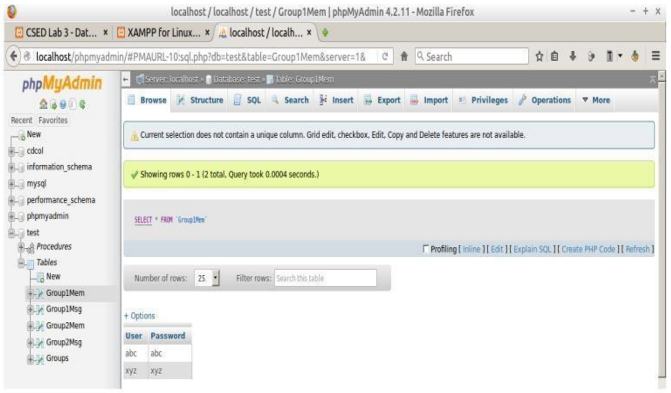
Number of rows: 25    Filter rows: Search this table

| Group |
|---|
| Group1 |
| Group2 |


localhost / localhost / test / Group1Mem | phpMyAdmin 4.2.11 - Mozilla Firefox

localhost/phpmyadmin/#PMAURL-11:tbl_structure.php?db=test&table=Group1Mem&

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | User | varchar(10) | latin1_swedish_ci | | No | None | |
| 2 | Password | varchar(10) | latin1_swedish_ci | | No | None | |


localhost / localhost / test / Group1Mem | phpMyAdmin 4.2.11 - Mozilla Firefox

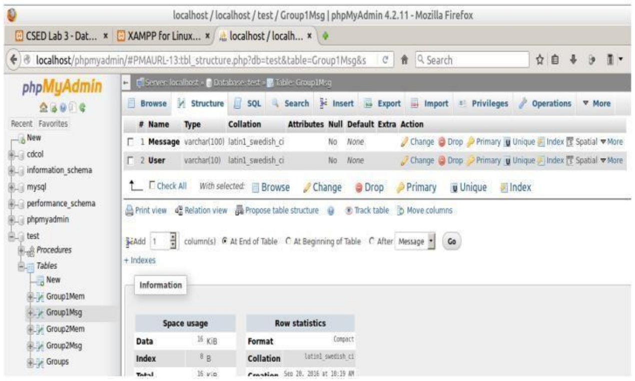
localhost/phpmyadmin/#PMAURL-10:sql.php?db=test&table=Group1Mem&server=1&

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

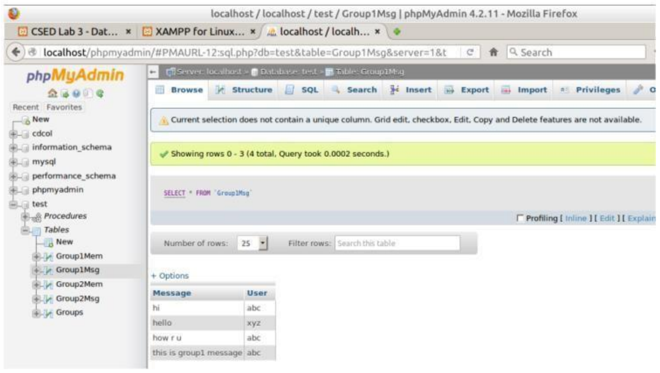
```
SELECT * FROM `Group1Mem`
```

| User | Password |
|---|---|
| abc | abc |
| xyz | xyz |


localhost / localhost / test / Group1Msg | phpMyAdmin 4.2.11 - Mozilla Firefox

localhost/phpmyadmin/#PMAURL-13:tbl_structure.php?db=test&table=Group1Msg&s

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | Message | varchar(100) | latin1_swedish_ci | | No | None | |
| 2 | User | varchar(10) | latin1_swedish_ci | | No | None | |


localhost / localhost / test / Group1Msg | phpMyAdmin 4.2.11 - Mozilla Firefox

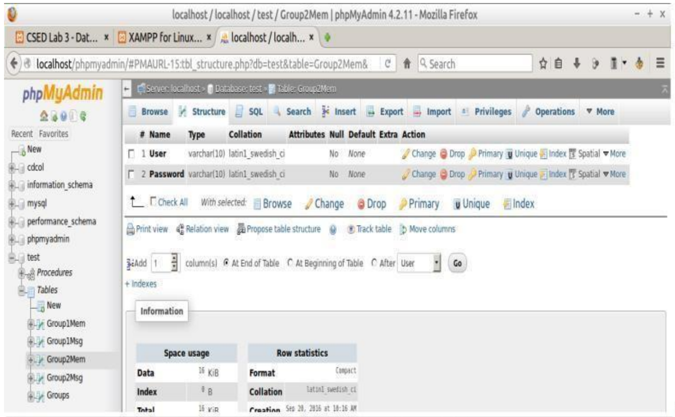
localhost/phpmyadmin/#PMAURL-12:sql.php?db=test&table=Group1Msg&server=1&t

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 3 (4 total, Query took 0.0002 seconds.)

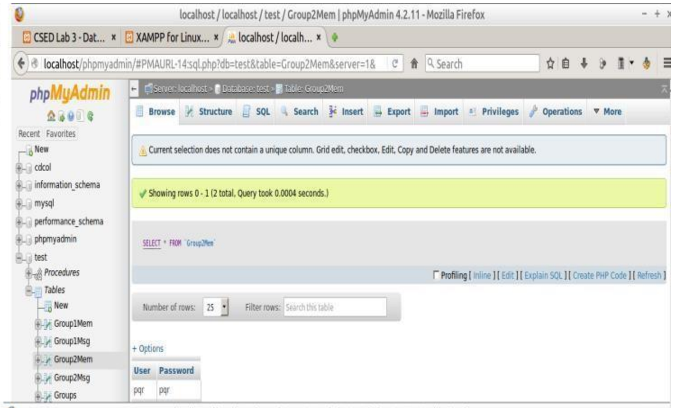
```
SELECT * FROM `Group1Msg`
```

| Message | User |
|---|---|
| hi | abc |
| hello | xyz |
| how r u | abc |
| this is group1 message | abc |


localhost / localhost / test / Group2Mem | phpMyAdmin 4.2.11 - Mozilla Firefox

localhost/phpmyadmin/#PMAURL-15:tbl_structure.php?db=test&table=Group2Mem&

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | User | varchar(10) | latin1_swedish_ci | | No | None | |
| 2 | Password | varchar(10) | latin1_swedish_ci | | No | None | |


localhost / localhost / test / Group2Mem | phpMyAdmin 4.2.11 - Mozilla Firefox

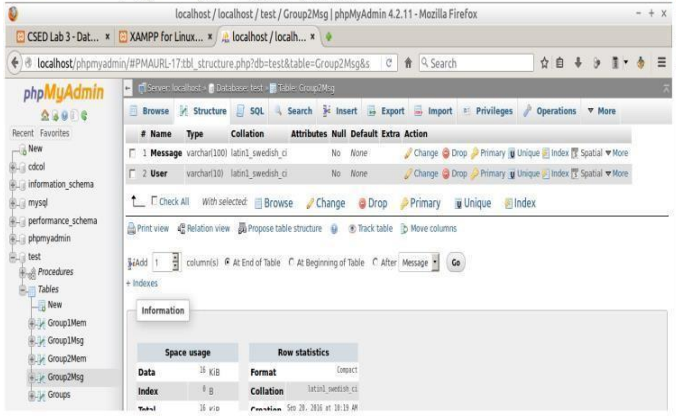
localhost/phpmyadmin/#PMAURL-14:sql.php?db=test&table=Group2Mem&server=1&

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

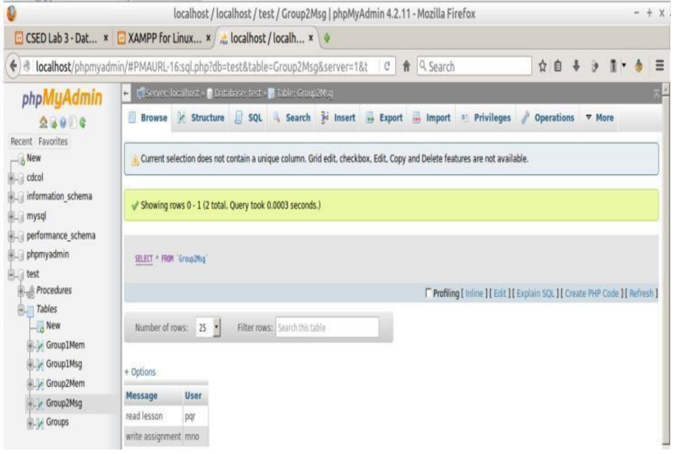
```
SELECT * FROM `Group2Mem`
```

| User | Password |
|---|---|
| pqr | pqr |


localhost / localhost / test / Group2Msg | phpMyAdmin 4.2.11 - Mozilla Firefox

localhost/phpmyadmin/#PMAURL-17:tbl_structure.php?db=test&table=Group2Msg&s

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | Message | varchar(100) | latin1_swedish_ci | | No | None | |
| 2 | User | varchar(10) | latin1_swedish_ci | | No | None | |


localhost / localhost / test / Group2Msg | phpMyAdmin 4.2.11 - Mozilla Firefox

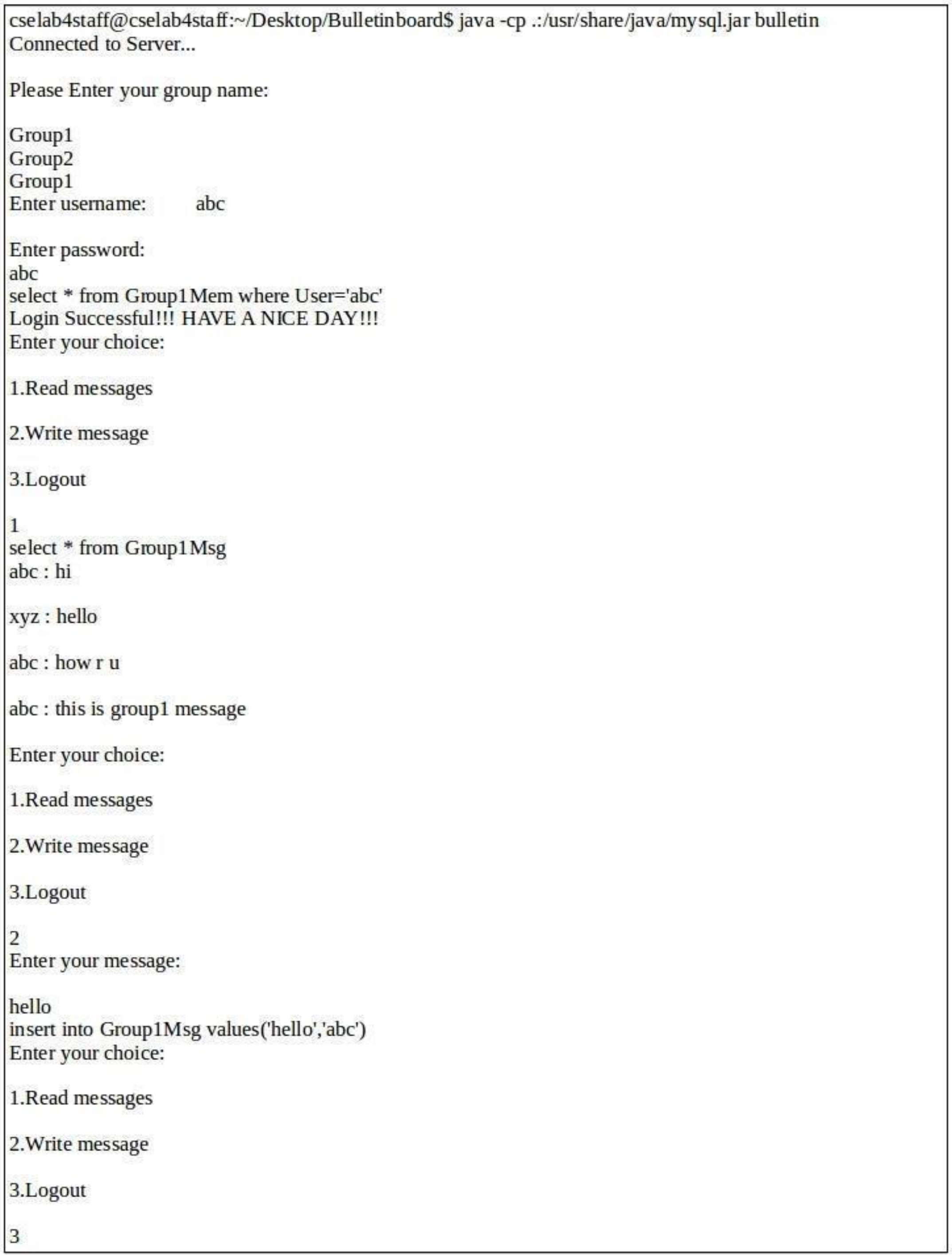
localhost/phpmyadmin/#PMAURL-16:sql.php?db=test&table=Group2Msg&server=1&t

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)

```
SELECT * FROM `Group2Msg`
```

| Message | User |
|---|---|
| read lesson | pqr |
| write assignment | mno |



```
cselab4staff@cselab4staff:~/Desktop/Bulletinboard$ java -cp .:/usr/share/java/mysql.jar bulletin
Connected to Server...

Please Enter your group name:

Group1
Group2
Group1
Enter username:        abc

Enter password:
abc
select * from Group1Mem where User='abc'
Login Successful!!! HAVE A NICE DAY!!!
Enter your choice:

1.Read messages

2.Write message

3.Logout

1
select * from Group1Msg
abc : hi

xyz : hello

abc : how r u

abc : this is group1 message

Enter your choice:

1.Read messages

2.Write message

3.Logout

2
Enter your message:

hello
insert into Group1Msg values('hello','abc')
Enter your choice:

1.Read messages

2.Write message

3.Logout

3
```

## 9.Implement a word count application which counts the number of occurrences of each word a large collection of documents Using Map Reduce model

**Workflow of MapReduce consists of 5 steps:**

1. Splitting – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. Mapping – as explained above.
3. Intermediate splitting – the entire process in parallel on different clusters. In order to group them in "Reduce

   Phase" the similar KEY data should be on the same cluster.
4. Reduce – it is nothing but mostly group by phase.
5. Combining – The last phase where all the data (individual result set from each cluster) is combined together to form a result.

Hadoop must be installed on your system with the Java SDK.
Steps

1. Open Eclipse> File > New > Java Project >( Name it –Samplewordcount) > Finish.
2. Right Click > New > Package ( Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - WordCount).
4. Add Following Reference Libraries:

   1.Right Click on Project > Build Path> Add External

   1. /usr/lib/hadoop-0.20/hadoop-core.jar
   2. Usr/lib/hadoop-0.20/lib/Commons-cli-1.2.jar

The program consists of three classes:

- Driver class (Public, void, static, or main; this is the entry point).
- The Map class which extends the public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Map function.
- The Reduce class which extends the public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Reduce function.

6. Make a jar file

Right Click on Project> Export> Select export destination as Jar File > next> Finish.

7. Take a text file and move it into HDFS format:

To move this into Hadoop directly, open the terminal and enter the following commands:

```
[training@localhost~]$hadoop fs -put wordcountFilewordCountFile
```

8. Run the jar file:

(Hadoop jar jarfilename.jar packageName.ClassNamePathToInputTextFilePathToOutputDirectry)

```
[training@localhost~]$hadoop jar MRProgramsDemo.jar PackageDemo.WordCountwordCountFile MRDir1
```

9. Open the result:

```
[training@localhost~]$hadoop fs -ls MRDir1
Found 3 items
-rw-r--r--   1 training supergroup          0 2016-02-23 03:36
/user/training/MRDir1/_SUCCESS drwxr-xr-x   - training supergroup          0 2016-
02-23 03:36 /user/training/MRDir1/_logs -rw-r--r--   1 training supergroup         20
2016-02-23 03:36 /user/training/MRDir1/part-r-00000
[training@localhost~]$hadoop fs -cat MRDir1/part-r-00000
BUS     7
CAR     4
TRAIN   6
```

**WordCount.java**

```
package PackageDemo;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount
{
public static void main(String [] args) throws Exception
{
		Configuration c=new Configuration();
		String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
		Path input=new Path(files[0]);
```

```
            Path output=new Path(files[1]);
            Job j=new Job(c,"wordcount");
            j.setJarByClass(WordCount.class);
            j.setMapperClass(MapForWordCount.class);
            j.setReducerClass(ReduceForWordCount.class);
            j.setOutputKeyClass(Text.class);
          j.setOutputValueClass(IntWritable.class);
          FileInputFormat.addInputPath(j, input);
            FileOutputFormat.setOutputPath(j, output);
            System.exit(j.waitForCompletion(true)?0:1);
}

public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>
{
public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException  {
String line = value.toString();
String[] words=line.split(",");
for(String word: words )
{
Text outputKey = new Text(word.toUpperCase().trim());
IntWritableoutputValue = new IntWritable(1);
con.write(outputKey, outputValue);
}
}
}
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException,
InterruptedException
{ int sum = 0;
for(IntWritable value : values)
{
sum += value.get();
}
con.write(word, new IntWritable(sum));
}
}
}
```

**10.Develop an application using Android SDK.**