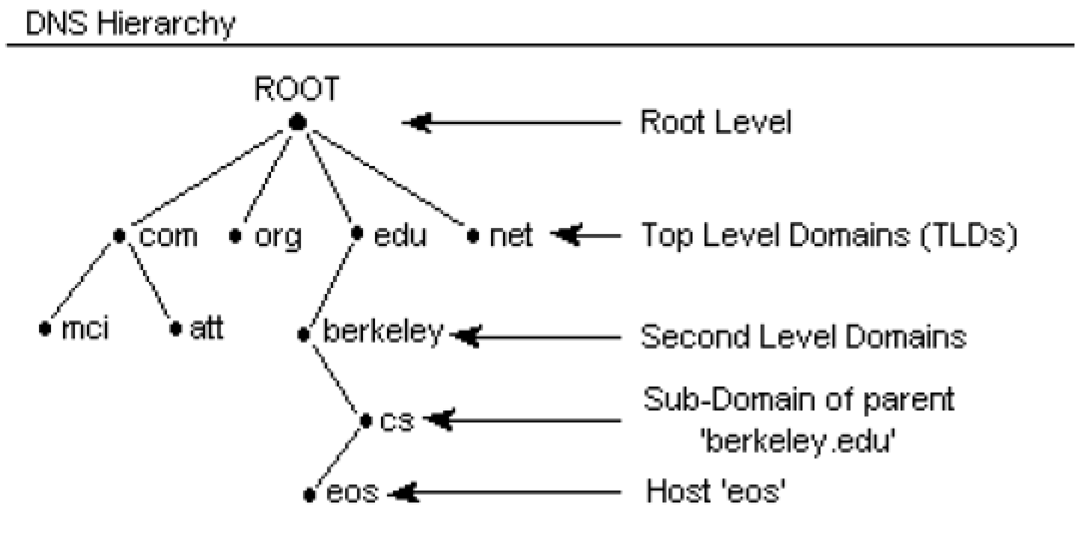
1. Domain Name Server Implementation

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.

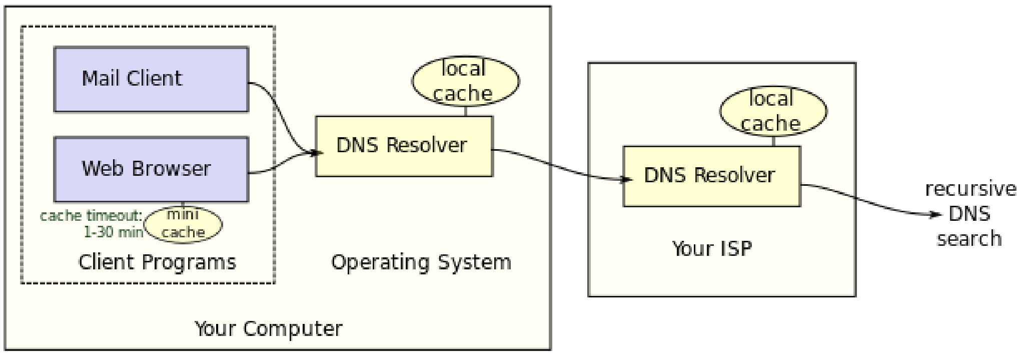
The Domain Name System also specifies the technical functionality of the database service which is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS, as part of the Internet Protocol Suite. Historically, other directory services preceding DNS were not scalable to large or global directories as they were originally based on text files, prominently the HOSTS.TXT resolver. DNS has been in wide use since the 1980s. The most common types of records stored in the DNS database are for DNS zone authority (SOA), IP addresses (A and AAAA), SMTP mail exchangers (MX), name servers (NS), pointers for reverse DNS lookups (PTR), and domain 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

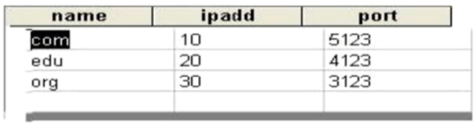
The client side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.



Implementation:

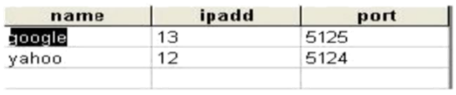
- Design the table with in the database as specified in the next section.
- Implement individual server for each domain with in DNS.
- Each server program is now in waiting state.
- Implement client program which accepts URL from user, divides the URL into different domain and fetch the IPs of the domains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

Table Name : **Client**



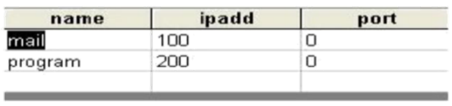
name	ipadd	port
com	10	5123
edu	20	4123
org	30	3123

Table Name: Root



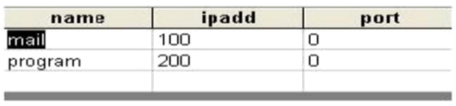
name	ipadd	port
google	13	5125
yahoo	12	5124

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.util.*;
class Server1 {
    public static void main(String a[]) {
        ServerSocket sock;
        Socket client;
        DataInputStream input;
        PrintStream ps;
        String url, u, s;
        Connection con;
        Statement smt;
        ResultSet rs;
        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);
                StringTokenizer st = 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.util.*;
class Server2 {
    public static void main(String a[]) {
        ServerSocket sock;
        Socket client;
        DataInputStream input;
        PrintStream ps;
        String url, u, s;
        Connection con;
        Statement smt;
        ResultSet rs;
        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);
                StringTokenizer st = 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);
        }
    }
}
```

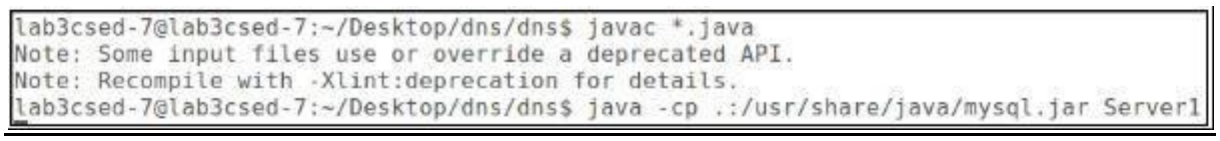
Server3.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
class Server3 {
    public static void main(String a[]) {
        ServerSocket sock;
        Socket client;
        DataInputStream input;
        PrintStream ps;
        String url, u, s;
        Connection con;
        Statement smt;
        ResultSet rs;
        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);
                StringTokenizer st = 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);
        }
    }
}
```

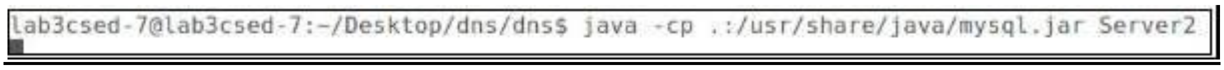
Client.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
class Client {
    public static void main(String a[]) {
        Socket clisock;
        DataInputStream input;
        PrintStream ps;
        String url, ip, s, u, p, str;
        int pno = 5123;
        Connection con;
        Statement smt;
        ResultSet rs;
        boolean status = true;
        try {
            ip = s = p = u = "\0";
            System.out.println("enter name to resolve");
            BufferedReader br = 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);
                StringTokenizer st = 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);
        }
    }
}
```

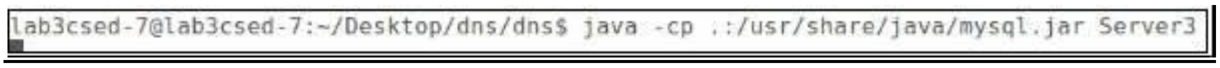
OUTPUT:



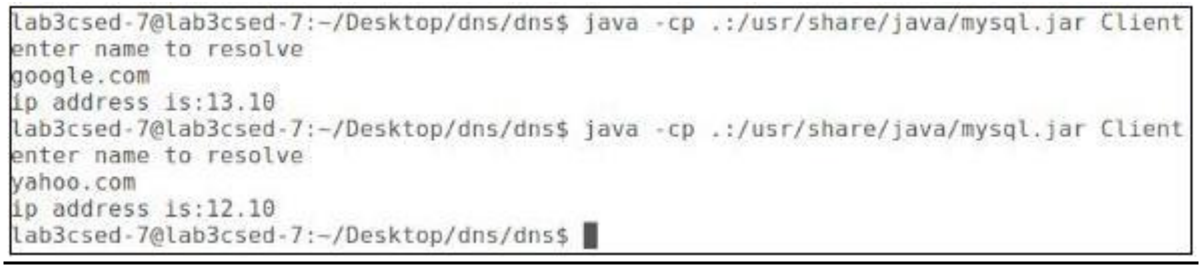
```
lab3csed-7@lab3csed-7:~/Desktop/dns/dns$ javac *.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
lab3csed-7@lab3csed-7:~/Desktop/dns/dns$ java -cp .:/usr/share/java/mysql.jar Server1
```



```
lab3csed-7@lab3csed-7:~/Desktop/dns/dns$ java -cp .:/usr/share/java/mysql.jar Server2
```



```
lab3csed-7@lab3csed-7:~/Desktop/dns/dns$ java -cp .:/usr/share/java/mysql.jar Server3
```



```
lab3csed-7@lab3csed-7:~/Desktop/dns/dns$ java -cp .:/usr/share/java/mysql.jar Client
enter name to resolve
google.com
ip address is:13.10
lab3csed-7@lab3csed-7:~/Desktop/dns/dns$ java -cp .:/usr/share/java/mysql.jar Client
enter name to resolve
yahoo.com
ip address is:12.10
lab3csed-7@lab3csed-7:~/Desktop/dns/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...");
            }
}
```

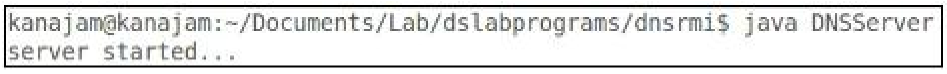
DNSClient.java

```
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));
                              }
}
```

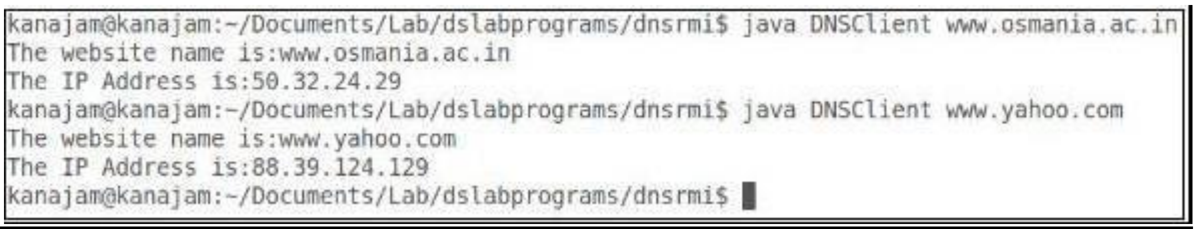
DNSServerIntf.java

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

OUTPUT:



```
kanajam@kanajam:~/Documents/Lab/dslabprograms/dnsrmi$ java DNSServer
server started...
```



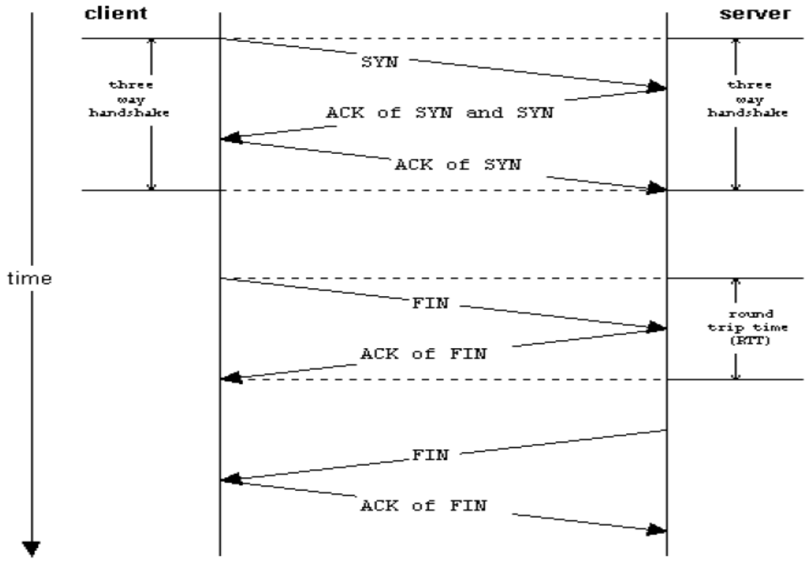
```
kanajam@kanajam:~/Documents/Lab/dslabprograms/dnsrmi$ java DNSClient www.osmania.ac.in
The website name is:www.osmania.ac.in
The IP Address is:50.32.24.29
kanajam@kanajam:~/Documents/Lab/dslabprograms/dnsrmi$ java DNSClient www.yahoo.com
The website name is:www.yahoo.com
The IP Address is:88.39.124.129
kanajam@kanajam:~/Documents/Lab/dslabprograms/dnsrmi$ 
```

4. 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, e-mail, 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 clients with several services simultaneously, as long as a client takes care of initiating any simultaneous connections to one destination port from different source ports.

TCPServer.java

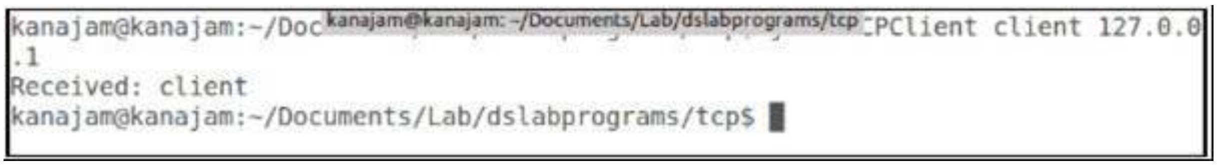
```
import java.net.*;
import java.io.*;
public class TCPServer {
   public static void main(String a[]) {
      Socket s = null;
      try {
         int serverPort = 8117;
         ServerSocket listenSocket = new ServerSocket(serverPort);
         while (true) {
            Socket clientSocket = listenSocket.accept();
            Connection c = new Connection(clientSocket);
         }
      } catch (IOException e) {
         System.out.println("Listen:" + e.getMessage());
      }
   }
}
class Connection extends Thread {
   DataInputStream in ;
   DataOutputStream out;
   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);
      }
      TCPServTCPServercatch(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());
         }
      }
```

```
    }
}
```

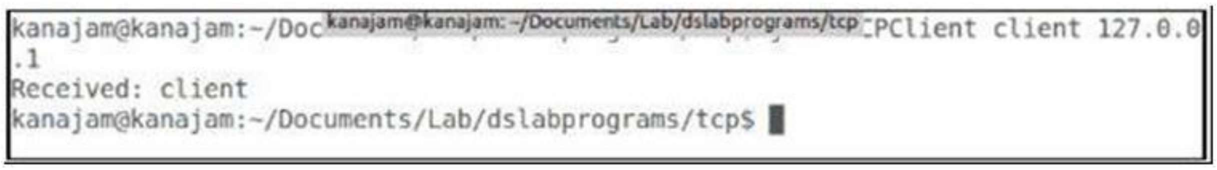
TCPClient.java

```
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());
            }
        }
    }
}
```

OUTPUT:



```
kanajam@kanajam:~/Doc kanajam@kanajam: ~/Documents/Lab/dslabprograms/tcp CPClient client 127.0.0
.1
Received: client
kanajam@kanajam:~/Documents/Lab/dslabprograms/tcp$
```



```
kanajam@kanajam:~/Doc kanajam@kanajam: ~/Documents/Lab/dslabprograms/tcp CPClient client 127.0.0
.1
Received: client
kanajam@kanajam:~/Documents/Lab/dslabprograms/tcp$
```

5. 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] Each packet sent or received on a Datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may arrive in any order and might not arrive at the receiving computer.UDP broadcasts sends are always enabled on a Datagram Socket. In order to receive broadcast packets a Datagram Socket must be bound to a more specific address.

An application binds a socket to its endpoint of data transmission, which is a combination of an IP address and a service port. A port is a software structure that is identified by the port number, a 16 bit integer value, 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

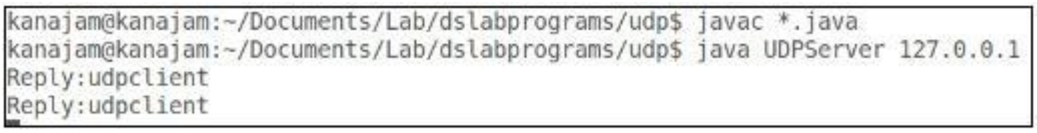
```
import java.net.*;
import java.io.*;
public class UDPServer {
   public static void main(String args[]) {
      DatagramSocket aSocket = null;
      try {
         aSocket = new DatagramSocket(8117);
         byte[] buffer = new byte[1000];
         while (true) {
            DatagramPacket request = 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();
        }
    }
}
```

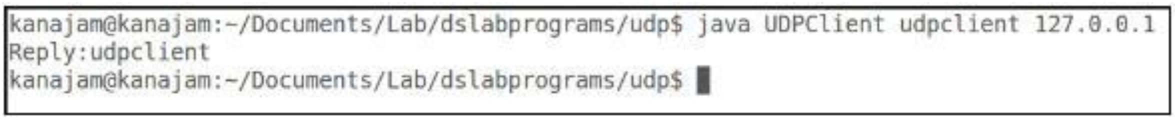
UDPClient.java

```
import java.net.*;
import java.io.*;
public class UDPClient {
    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        try {
            aSocket = new DatagramSocket();
            byte[] m = args[0].getBytes();
            InetAddress aHost = 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();
        }
    }
}
```

OUTPUT:



```
kanajam@kanajam:~/Documents/Lab/dslabprograms/udp$ javac *.java
kanajam@kanajam:~/Documents/Lab/dslabprograms/udp$ java UDPServer 127.0.0.1
Reply:udpclient
Reply:udpclient
```



```
kanajam@kanajam:~/Documents/Lab/dslabprograms/udp$ java UDPClient udpclient 127.0.0.1
Reply:udpclient
kanajam@kanajam:~/Documents/Lab/dslabprograms/udp$ 
```

6. 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.

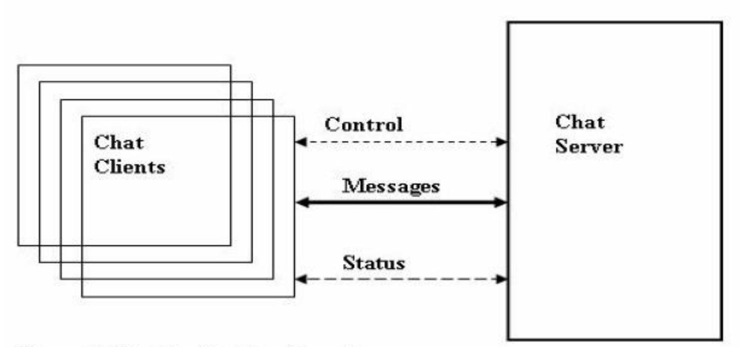


Figure 1: Chat Application Overview

<u>ChatServer.java</u>

```
import java.net.*;
import java.io.*;
public class ChatServer implements Runnable {
    private ChatServerThread clients[] = new ChatServerThread[50];
    private ServerSocket server = 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:java ChatServer Port");
        else
            server = new ChatServer(Integer.parseInt(a[0]));
    }
}
```

ChatServerThread.java

```
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(ChatServer serv, 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();
    }
}
```

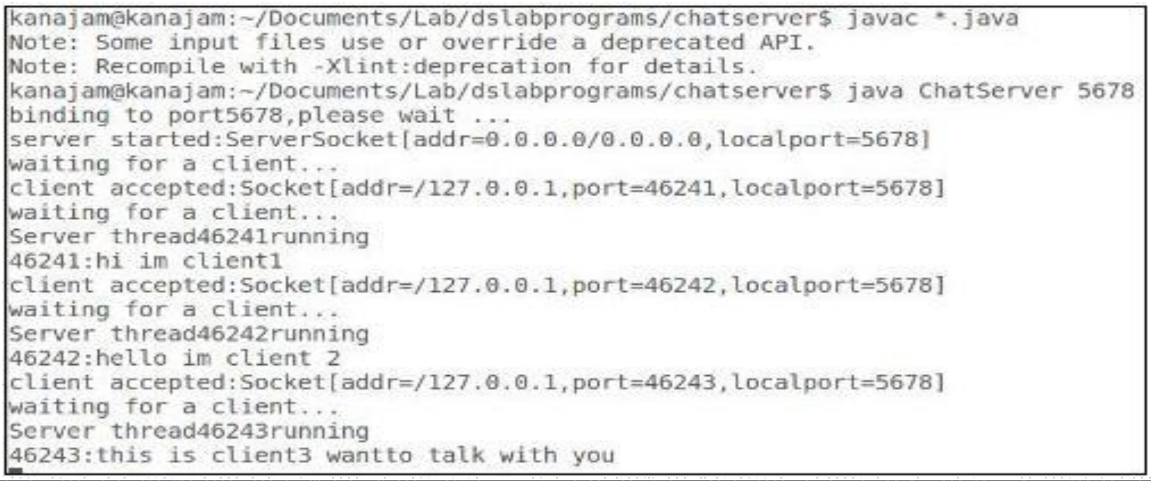
ChatClient.java

```
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 (IOException ioe) {
            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:java ChatClient host port");
        else
            client = new ChatClient(args[0], Integer.parseInt(args[1]));
    }
}
```

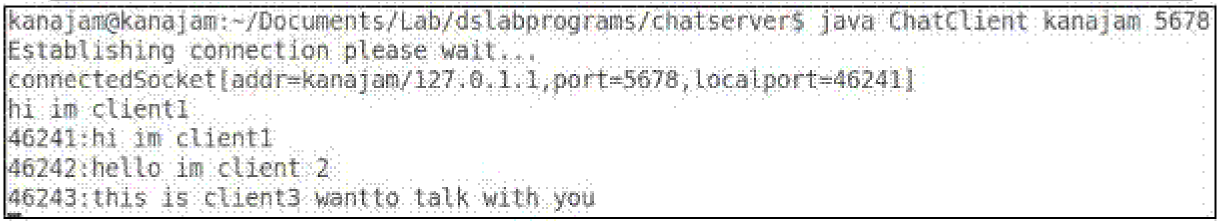
ChatClientThread.java

```
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();
            }
        }
    }
}
```

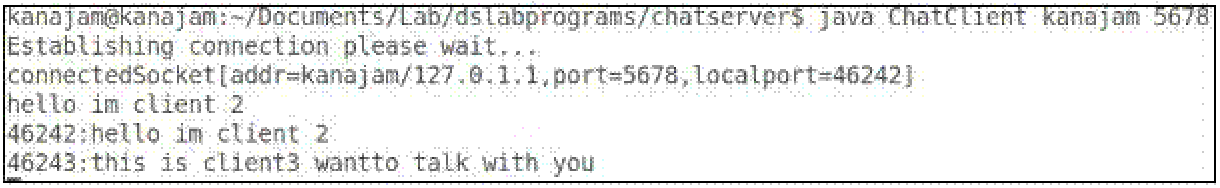
OUTPUT:



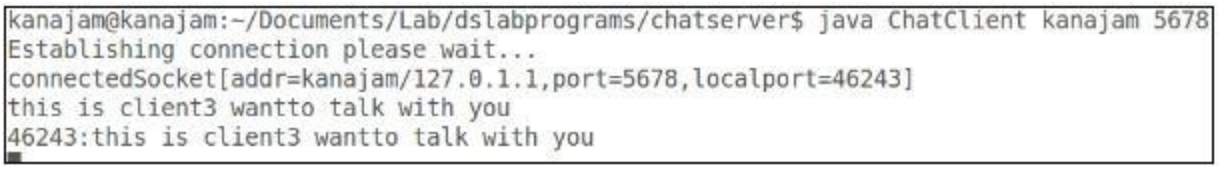
```
kanajam@kanajam:~/Documents/Lab/dslabprograms/chatserver$ javac *.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
kanajam@kanajam:~/Documents/Lab/dslabprograms/chatserver$ java ChatServer 5678
binding to port5678,please wait ...
server started:ServerSocket[addr=0.0.0.0/0.0.0.0,localport=5678]
waiting for a client...
client accepted:Socket[addr=/127.0.0.1,port=46241,localport=5678]
waiting for a client...
Server thread46241running
46241:hi im client1
client accepted:Socket[addr=/127.0.0.1,port=46242,localport=5678]
waiting for a client...
Server thread46242running
46242:hello im client 2
client accepted:Socket[addr=/127.0.0.1,port=46243,localport=5678]
waiting for a client...
Server thread46243running
46243:this is client3 wantto talk with you
```



```
kanajam@kanajam:~/Documents/Lab/dslabprograms/chatserver$ java ChatClient kanajam 5678
Establishing connection please wait...
connectedSocket[addr=kanajam/127.0.1.1,port=5678,localport=46241]
hi im client1
46241:hi im client1
46242:hello im client 2
46243:this is client3 wantto talk with you
```



```
kanajam@kanajam:~/Documents/Lab/dslabprograms/chatserver$ java ChatClient kanajam 5678
Establishing connection please wait...
connectedSocket[addr=kanajam/127.0.1.1,port=5678,localport=46242]
hello im client 2
46242:hello im client 2
46243:this is client3 wantto talk with you
```



```
kanajam@kanajam:~/Documents/Lab/dslabprograms/chatserver$ java ChatClient kanajam 5678
Establishing connection please wait...
connectedSocket[addr=kanajam/127.0.1.1,port=5678,localport=46243]
this is client3 wantto talk with you
46243:this is client3 wantto talk with you
```

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

/nfsshare

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 nfs

portmap

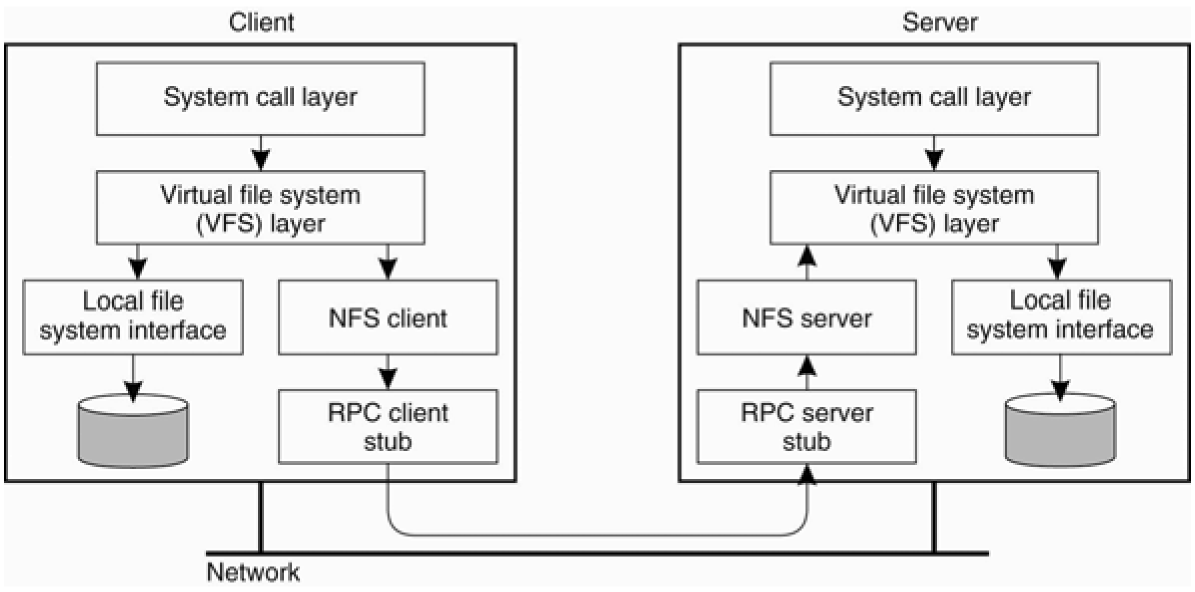
5) Daemons:

nfsd

quotad

mountd

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:

#services nfs 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 –ivh ftp://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 –ivh nfs-utils* --force –aid

#rpm –ivh portmap* --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

#services nfs 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.

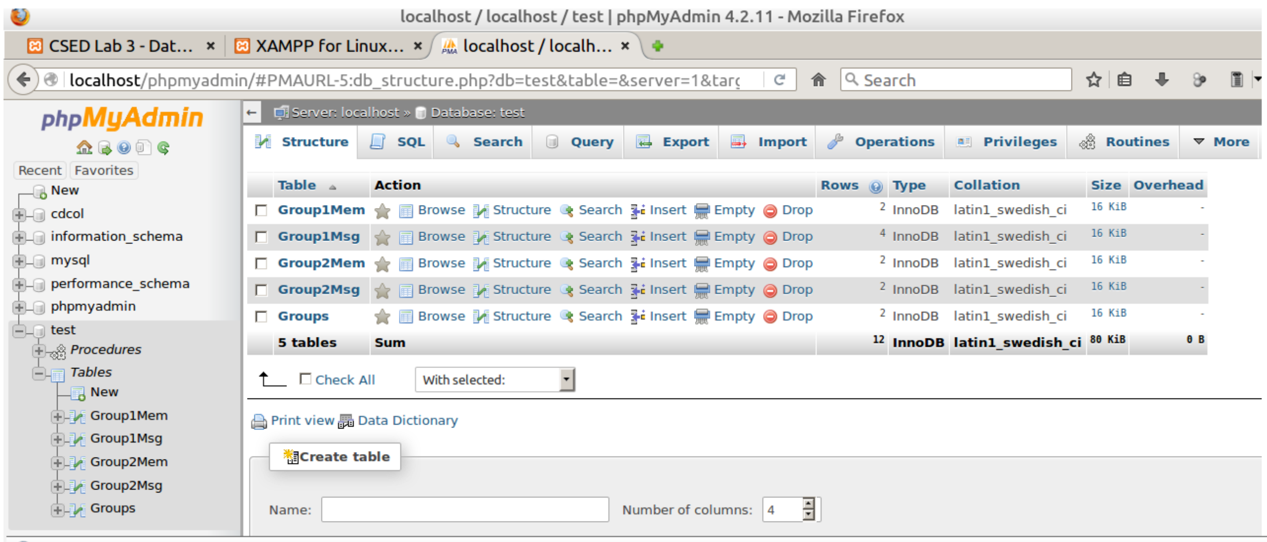
3) In NFS all files & Directory are by default in read only mode.

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.*;
public class bulletin {
    static Connection con;
    static Statement st;
    static ResultSet rs;
    public static void main(String[] args) throws IOException {
            try {
                Class.ForName("com.mysql.jdbc.Driver");
                con = DriverManager.getConnection("jdbc.mysql://172.0.2.9/database", "username",: password ");
                    st = con.createStatement();
                }
                catch (ClassNotFoundException ex) {
                    ex.printStackTrace();
                } catch (SQLException ex) {
                    ex.printStack Trace();
                }
                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;
                    BufferedReader br = 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 (rts.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) {
                                        case1: 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;
                                        case2: System.out.println("enter your message :\n");
                                        String msg = br.readLine();
                                        query = "insert into" + grpname + "Msg values('" + msg + "','" + usr + "')";
                                        System.out.println(query);
                                        st.executeUpdate(query);
                                        break;
                                        case3: 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 (SQL Exception ex) {
                        ex.printStackTrace();
                    }
                }
              }
```

OUTPUT:



localhost / localhost / test | phpMyAdmin 4.2.11 - Mozilla Firefox

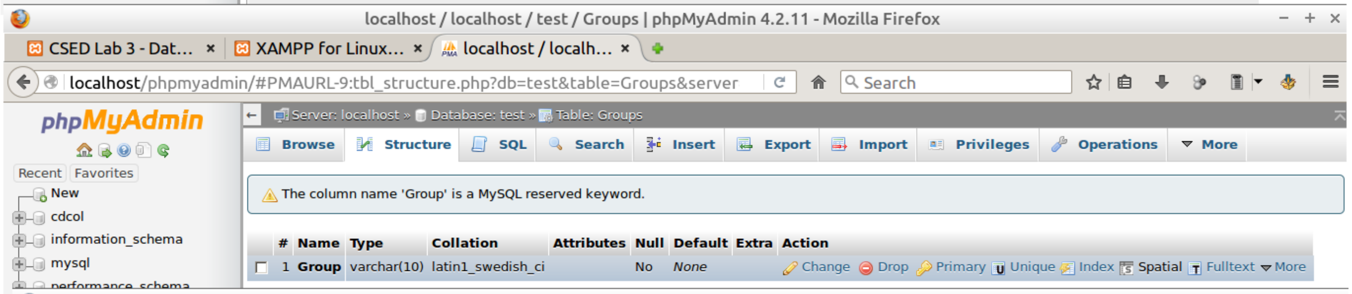
localhost/phpmyadmin/#PMAURL-5:db_structure.php?db=test&table=&server=1&targ

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

Create table

Name: Number of columns: 4



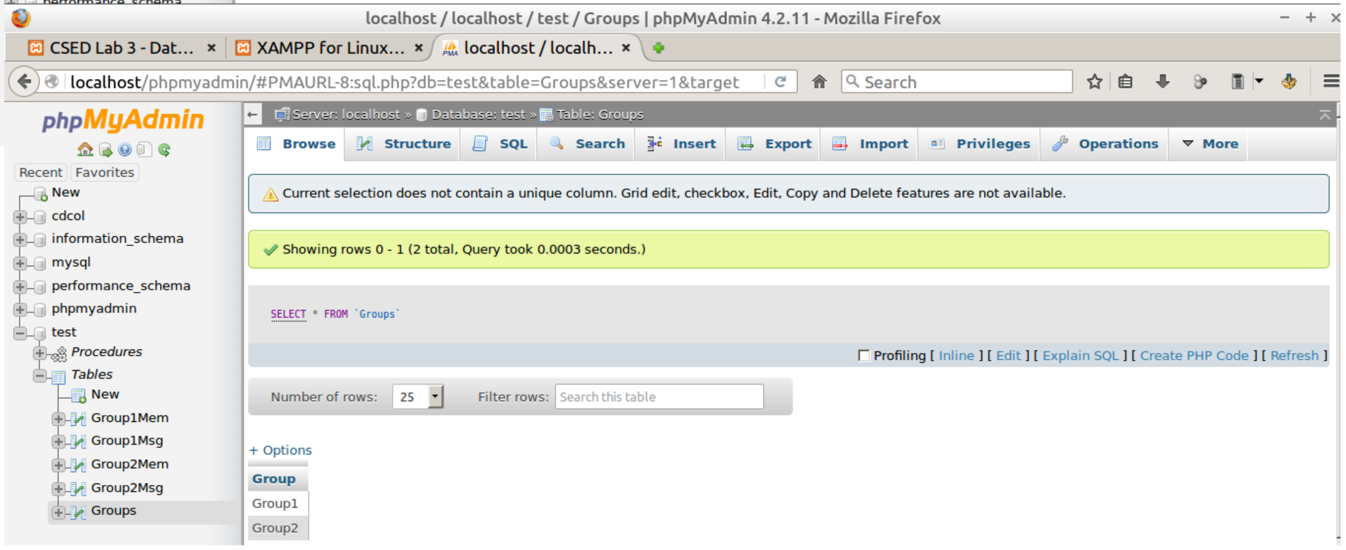
localhost / localhost / test / Groups | phpMyAdmin 4.2.11 - Mozilla Firefox

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

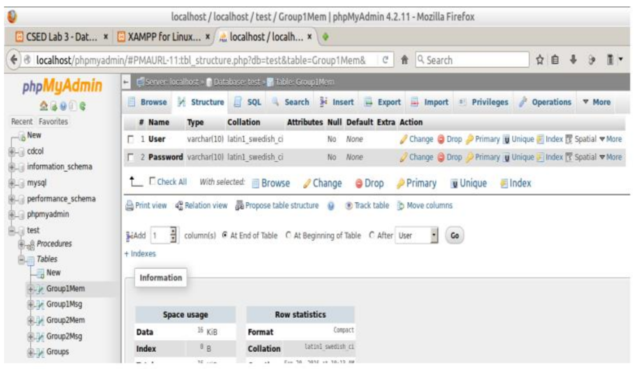
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

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

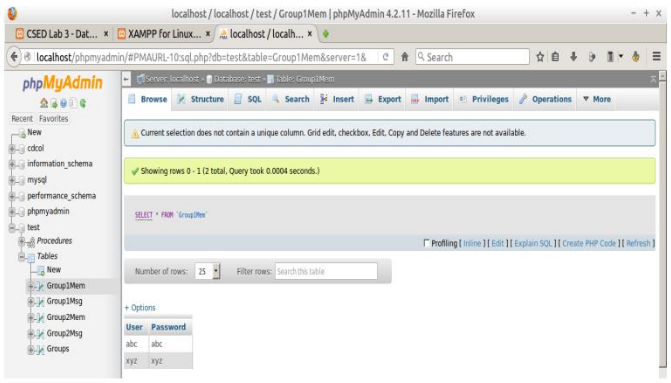

localhost / localhost / test / Group1Mem | phpMyAdmin 4.2.11 - Mozilla Firefox

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

Server: localhost » Database: 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	

Space usage		Row statistics	
Data	16 KiB	Format	Compact
Index	0 B	Collation	latin1_swedish_ci


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

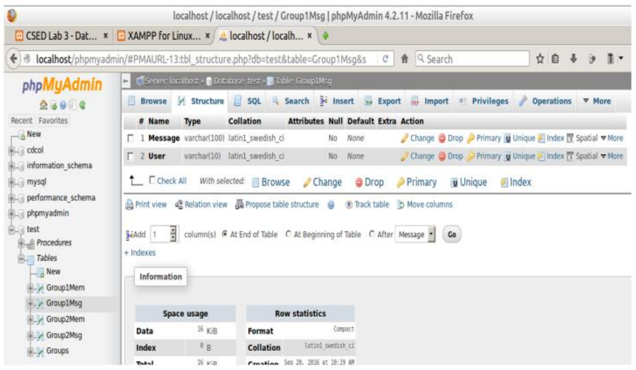
localhost/phpmyadmin/#PMAURL-10:sql.php?db=test&table=Group1Mem&server=1&

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

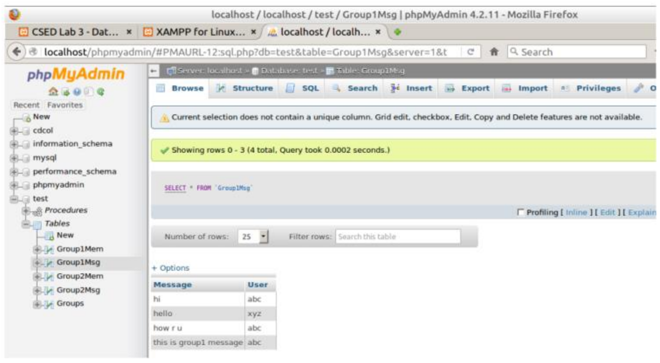

localhost / localhost / test / Group1Msg | phpMyAdmin 4.2.11 - Mozilla Firefox

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

Server: localhost » Database: test » Table: Group1Msg

#	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	

Space usage		Row statistics	
Data	16 KiB	Format	Compact
Index	0 B	Collation	latin1_swedish_ci
Total	16 KiB	Creation	Sep 28, 2016 at 10:19 AM


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

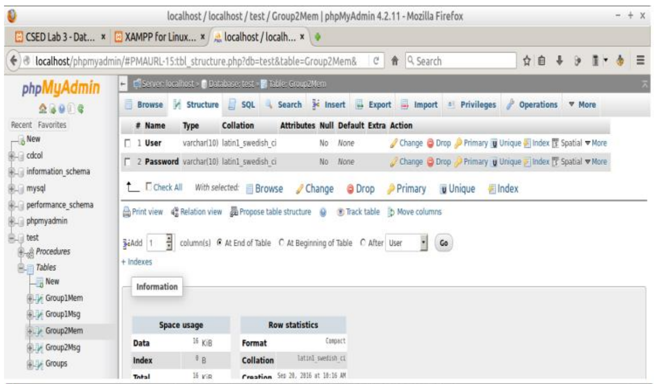
localhost/phpmyadmin/#PMAURL-12:sql.php?db=test&table=Group1Msg&server=1&t

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

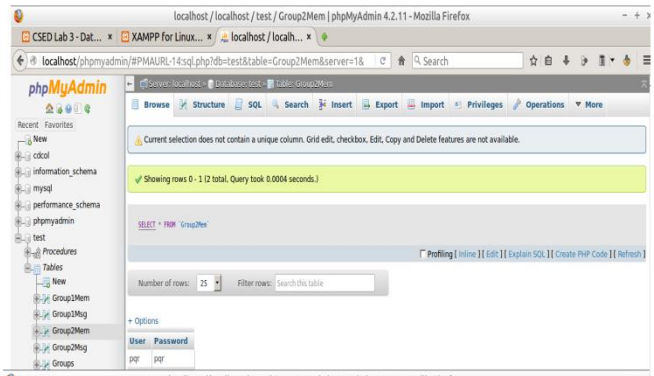

localhost / localhost / test / Group2Mem | phpMyAdmin 4.2.11 - Mozilla Firefox

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

Server: localhost » Database: 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	

Space usage		Row statistics	
Data	16 KiB	Format	Compact
Index	0 B	Collation	latin1_swedish_ci
Total	16 KiB	Creation	Sep 28, 2016 at 10:16 AM


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

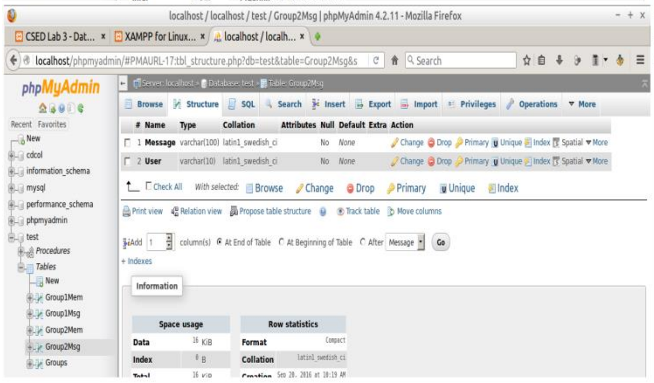
localhost/phpmyadmin/#PMAURL-14:sql.php?db=test&table=Group2Mem&server=1&

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

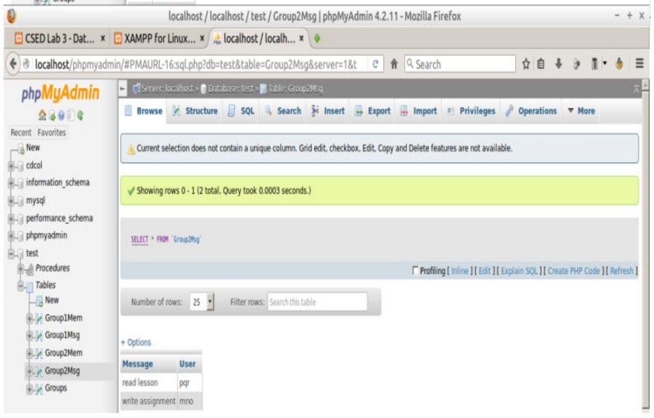

localhost / localhost / test / Group2Msg | phpMyAdmin 4.2.11 - Mozilla Firefox

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

Server: localhost » Database: test » Table: Group2Msg

#	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	

Space usage		Row statistics	
Data	16 KiB	Format	Compact
Index	0 B	Collation	latin1_swedish_ci
Total	16 KiB	Creation	Sep 28, 2016 at 10:19 AM


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

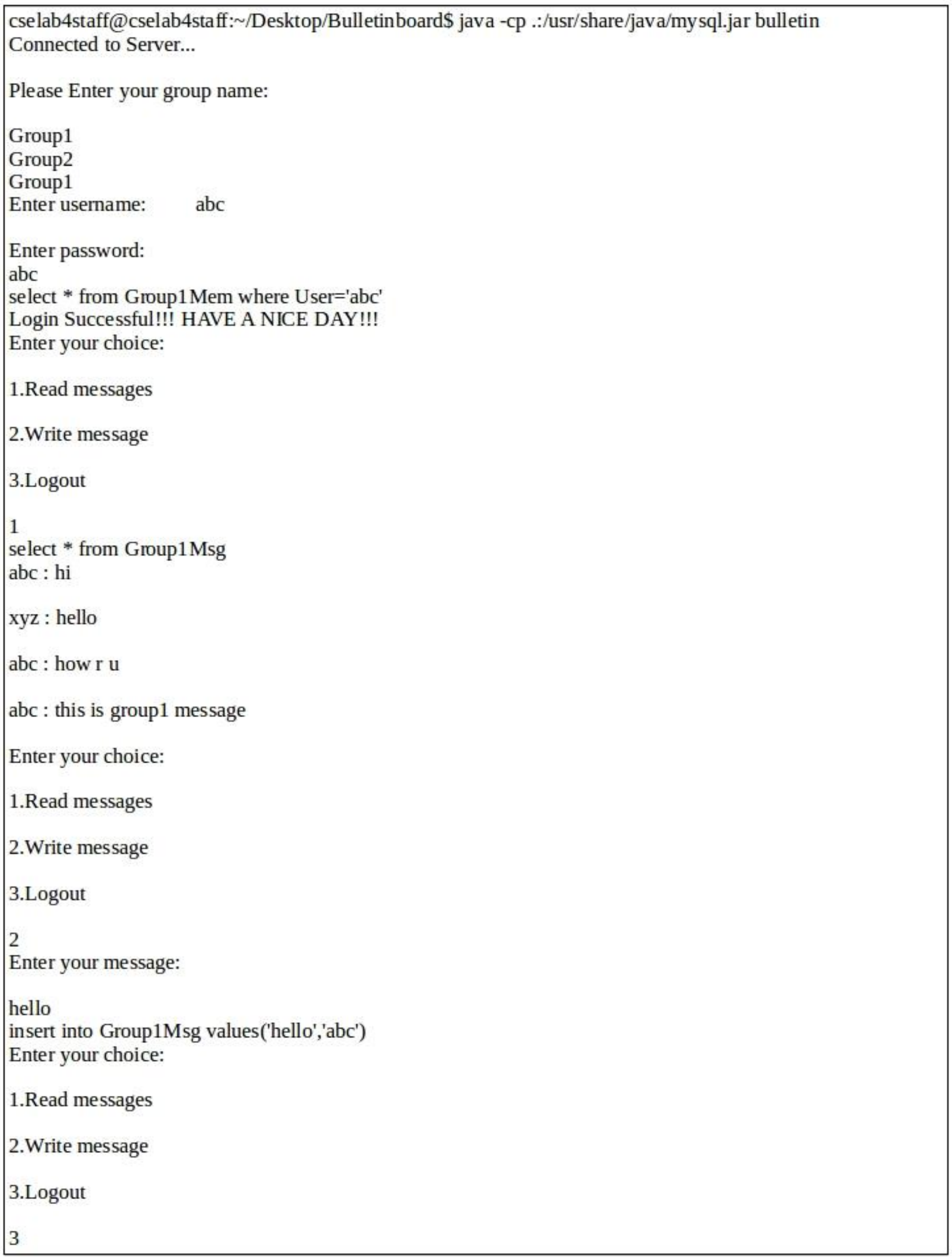
localhost/phpmyadmin/#PMAURL-16:sql.php?db=test&table=Group2Msg&server=1&t

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 wordcountFile wordCountFile
```

8. Run the jar file:

(Hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFile PathToOutputDirectry)

```
[training@localhost ~]$ hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile 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());
  IntWritable outputValue = 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.

MainActivity .java

```
package com.example.snaked;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
    private Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button=findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                openGameScreen();
            }
        });
    }

    public void openGameScreen(){
        Intent intent=new Intent(this, Snake.class);
        startActivity(intent);
    }
}
```

Snake.java

```
package com.example.snaked;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;

import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.drawable.Drawable;
import android.util.AttributeSet;
import android.view.View;

import static com.example.snaked.SnakeView.EAST;
import static com.example.snaked.SnakeView.LOSE;
import static com.example.snaked.SnakeView.NORTH;
import static com.example.snaked.SnakeView.PAUSE;
import static com.example.snaked.SnakeView.READY;
import static com.example.snaked.SnakeView.RUNNING;
import static com.example.snaked.SnakeView.SOUTH;
import static com.example.snaked.SnakeView.WEST;

public class Snake extends Activity {

    private SnakeView mSnakeView;
    private Button up;
    private Button down;
    private Button left;
    private Button right;
    private static String ICICLE_KEY = "snake-view";


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_snake);
        up = (Button) findViewById(R.id.button);

        left = (Button) findViewById(R.id.button2);


        right = (Button) findViewById(R.id.button3);


        down = (Button) findViewById(R.id.button4);

        mSnakeView = (SnakeView) findViewById(R.id.snake);
        mSnakeView.setTextView((TextView) findViewById(R.id.text));

        if (savedInstanceState == null) {
            mSnakeView.setMode(READY);
        } else {
            Bundle map = savedInstanceState.getBundle(ICICLE_KEY);
            if (map != null) {
                mSnakeView.restoreState(map);
            } else {
                mSnakeView.setMode(PAUSE);
            }
        }
    }

    public void buttonClicked(View view) {
        if (view.getId() == R.id.button) {
            if (mSnakeView.mMode == READY | mSnakeView.mMode == LOSE) {
                mSnakeView.initNewGame();
                mSnakeView.setMode(RUNNING);
                mSnakeView.update();
            }
            if (mSnakeView.mMode == PAUSE) {
                mSnakeView.setMode(RUNNING);
                mSnakeView.update();
            }
            if (mSnakeView.mDirection != SOUTH) {
                mSnakeView.mNextDirection = NORTH;
            }
        }
        else if (view.getId() == R.id.button2) {
            if (mSnakeView.mDirection != EAST) {
                mSnakeView.mNextDirection = WEST;
            }
        }
        else if (view.getId() == R.id.button3) {
            if (mSnakeView.mDirection != WEST) {
                mSnakeView.mNextDirection = EAST;
            }
        }
        else if (view.getId() == R.id.button4) {
            if (mSnakeView.mDirection != NORTH) {
                mSnakeView.mNextDirection = SOUTH;
            }
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        mSnakeView.setMode(PAUSE);
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        outState.putBundle(ICICLE_KEY, mSnakeView.saveState());
    }

}


class TileView extends View {

    protected static int mTileSize;

    protected static int mXTileCount;
    protected static int mYTileCount;

    private static int mXOffset;
    private static int mYOffset;

    private Bitmap[] mTileArray;

    private int[][] mTileGrid;

    private final Paint mPaint = new Paint();

    public TileView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);

        TypedArray a = context.obtainStyledAttributes(attrs,
                R.styleable.TileView);

        mTileSize = a.getInt(R.styleable.TileView_tileSize, 12);

        a.recycle();
```

```
}

public TileView(Context context, AttributeSet attrs) {
   super(context, attrs);

   TypedArray a = context.obtainStyledAttributes(attrs,
         R.styleable.TileView);

   mTileSize = a.getInt(R.styleable.TileView_tileSize, 12);

   a.recycle();
}


public void resetTiles(int tilecount) {
   mTileArray = new Bitmap[tilecount];
}

@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
   mXTileCount = (int) Math.floor(w / mTileSize);
   mYTileCount = (int) Math.floor(h / mTileSize);

   mXOffset = ((w - (mTileSize * mXTileCount)) / 2);
   mYOffset = ((h - (mTileSize * mYTileCount)) / 2);

   mTileGrid = new int[mXTileCount][mYTileCount];
   clearTiles();
}


public void loadTile(int key, Drawable tile) {
   Bitmap bitmap = Bitmap.createBitmap(mTileSize, mTileSize,
         Bitmap.Config.ARGB_8888);
   Canvas canvas = new Canvas(bitmap);
   tile.setBounds(0, 0, mTileSize, mTileSize);
   tile.draw(canvas);

   mTileArray[key] = bitmap;
}


public void clearTiles() {
   for (int x = 0; x < mXTileCount; x++) {
      for (int y = 0; y < mYTileCount; y++) {
         setTile(0, x, y);
      }
   }
}


public void setTile(int tileindex, int x, int y) {
   mTileGrid[x][y] = tileindex;
}

@Override
public void onDraw(Canvas canvas) {
   super.onDraw(canvas);
```

```
        for (int x = 0; x < mXTileCount; x += 1) {
            for (int y = 0; y < mYTileCount; y += 1) {
                if (mTileGrid[x][y] > 0) {
                    canvas.drawBitmap(mTileArray[mTileGrid[x][y]], mXOffset + x
                            * mTileSize, mYOffset + y * mTileSize, mPaint);
                }
            }
        }

    }

}
```

Snakeview.java

```
package com.example.snaked;

import android.content.Context;
import android.content.res.Resources;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.AttributeSet;
import android.util.Log;
import android.view.View;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.Random;


class SnakeView extends TileView {
    private View view;

    public SnakeView(Context context) {
        this(context, null);
    }
    public static final String TAG = "SnakeView";

    public int mMode = READY;
    public static final int PAUSE = 0;
    public static final int READY = 1;
    public static final int RUNNING = 2;
    public static final int LOSE = 3;

    public int mDirection = NORTH;
    public int mNextDirection = NORTH;
    public static final int NORTH = 1;
    public static final int SOUTH = 2;
    public static final int EAST = 3;
    public static final int WEST = 4;

    public static final int RBLOCK = 1;
    public static final int YBLOCK = 2;
    public static final int GBLOCK = 3;
    public static final int BBLOCK = 4;

    public long mScore = 0;
    public long mMoveDelay = 200;

    public long mLastMove;

    public TextView mStatusText;

    public ArrayList<Coordinate> mSnakeTrail = new ArrayList<Coordinate>();
    public ArrayList<Coordinate> mAppleList = new ArrayList<Coordinate>();

    public static final Random RNG = new Random();

    public RefreshHandler mRedrawHandler = new RefreshHandler();

    class RefreshHandler extends Handler {

        @Override
        public void handleMessage(Message msg) {
           SnakeView.this.update();
           SnakeView.this.invalidate();
        }

        public void sleep(long delayMillis) {
           this.removeMessages(0);
           sendMessageDelayed(obtainMessage(0), delayMillis);
        }
    };


    public SnakeView(Context context, AttributeSet attrs) {
        super(context, attrs);
        initSnakeView();
    }

    public SnakeView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        initSnakeView();
    }

    public void initSnakeView() {
        setFocusable(true);

        Resources r = this.getContext().getResources();

        resetTiles(5);
        loadTile(RBLOCK, r.getDrawable(R.drawable.rblock));
        loadTile(YBLOCK, r.getDrawable(R.drawable.yblock));
        loadTile(GBLOCK, r.getDrawable(R.drawable.gblock));
        loadTile(BBLOCK, r.getDrawable(R.drawable.bblock));
    }

    public void initNewGame() {
        mSnakeTrail.clear();
        mAppleList.clear();

        mSnakeTrail.add(new Coordinate(7, 7));
        mSnakeTrail.add(new Coordinate(6, 7));
        mSnakeTrail.add(new Coordinate(5, 7));
        mSnakeTrail.add(new Coordinate(4, 7));
```

```
        mSnakeTrail.add(new Coordinate(3, 7));
        mSnakeTrail.add(new Coordinate(2, 7));
        mNextDirection = NORTH;

        addRandomApple();
        addRandomApple();

        mMoveDelay = 200;
        mScore = 0;
    }


    private int[] coordArrayListToArray(ArrayList<Coordinate> cvec) {
        int count = cvec.size();
        int[] rawArray = new int[count * 2];
        for (int index = 0; index < count; index++) {
            Coordinate c = cvec.get(index);
            rawArray[2 * index] = c.x;
            rawArray[2 * index + 1] = c.y;
        }
        return rawArray;
    }


    public Bundle saveState() {
        Bundle map = new Bundle();

        map.putIntArray("mAppleList", coordArrayListToArray(mAppleList));
        map.putInt("mDirection", Integer.valueOf(mDirection));
        map.putInt("mNextDirection", Integer.valueOf(mNextDirection));
        map.putLong("mMoveDelay", Long.valueOf(mMoveDelay));
        map.putLong("mScore", Long.valueOf(mScore));
        map.putIntArray("mSnakeTrail", coordArrayListToArray(mSnakeTrail));

        return map;
    }


    private ArrayList<Coordinate> coordArrayToArrayList(int[] rawArray) {
        ArrayList<Coordinate> coordArrayList = new ArrayList<Coordinate>();

        int coordCount = rawArray.length;
        for (int index = 0; index < coordCount; index += 2) {
            Coordinate c = new Coordinate(rawArray[index], rawArray[index + 1]);
            coordArrayList.add(c);
        }
        return coordArrayList;
    }


    public void restoreState(Bundle icicle) {
        setMode(PAUSE);

        mAppleList = coordArrayToArrayList(icicle.getIntArray("mAppleList"));
        mDirection = icicle.getInt("mDirection");
        mNextDirection = icicle.getInt("mNextDirection");
        mMoveDelay = icicle.getLong("mMoveDelay");
        mScore = icicle.getLong("mScore");
        mSnakeTrail = coordArrayToArrayList(icicle.getIntArray("mSnakeTrail"));
    }


    public void setTextView(TextView newView) {
        mStatusText = newView;
    }

    public void setMode(int newMode) {
        int oldMode = mMode;
        mMode = newMode;

        if (newMode == RUNNING & oldMode != RUNNING) {
            mStatusText.setVisibility(View.INVISIBLE);
            update();
            return;
        }

        Resources res = getContext().getResources();
        CharSequence str = "";
        if (newMode == PAUSE) {
            str = res.getText(R.string.mode_pause);
        }
        if (newMode == READY) {
            str = res.getText(R.string.mode_ready);
        }
        if (newMode == LOSE) {
            str = res.getString(R.string.mode_lose_prefix) + mScore
                    + res.getString(R.string.mode_lose_suffix);
        }

        mStatusText.setText(str);
        mStatusText.setVisibility(View.VISIBLE);
    }


    private void addRandomApple() {
        Coordinate newCoord = null;
        boolean found = false;
        while (!found) {
            int newX = 1 + RNG.nextInt(mXTileCount - 2);
            int newY = 1 + RNG.nextInt(mYTileCount - 2);
            newCoord = new Coordinate(newX, newY);

            boolean collision = false;
            int snakelength = mSnakeTrail.size();
            for (int index = 0; index < snakelength; index++) {
                if (mSnakeTrail.get(index).equals(newCoord)) {
                    collision = true;
                }
            }
            found = !collision;
        }
        if (newCoord == null) {
            Log.e(TAG, "Somehow ended up with a null newCoord!");
        }
        mAppleList.add(newCoord);
    }
```

```
public void update() {
    if (mMode == RUNNING) {
        long now = System.currentTimeMillis();

        if (now - mLastMove > mMoveDelay) {
            clearTiles();
            updateWalls();
            updateSnake();
            updateApples();
            mLastMove = now;
        }
        mRedrawHandler.sleep(mMoveDelay);
    }

}


private void updateWalls() {
    for (int x = 0; x < mXTileCount; x++) {
        setTile(RBLOCK, x, 0);
        setTile(RBLOCK, x, mYTileCount - 1);
    }
    for (int y = 1; y < mYTileCount - 1; y++) {
        setTile(RBLOCK, 0, y);
        setTile(RBLOCK, mXTileCount - 1, y);
    }
}


private void updateApples() {
    for (Coordinate c : mAppleList) {
        setTile(BBLOCK, c.x, c.y);
    }
}


private void updateSnake() {
    boolean growSnake = false;

    Coordinate head = mSnakeTrail.get(0);
    Coordinate newHead = new Coordinate(1, 1);

    mDirection = mNextDirection;

    switch (mDirection) {
        case EAST: {
            newHead = new Coordinate(head.x + 1, head.y);
            break;
        }
        case WEST: {
            newHead = new Coordinate(head.x - 1, head.y);
            break;
        }
        case NORTH: {
            newHead = new Coordinate(head.x, head.y - 1);
            break;
        }
        case SOUTH: {
            newHead = new Coordinate(head.x, head.y + 1);
            break;
        }
    }


    if ((newHead.x < 1) || (newHead.y < 1) || (newHead.x > mXTileCount - 2)
            || (newHead.y > mYTileCount - 2)) {
        setMode(LOSE);
        return;

    }

    int snakelength = mSnakeTrail.size();
    for (int snakeindex = 0; snakeindex < snakelength; snakeindex++) {
        Coordinate c = mSnakeTrail.get(snakeindex);
        if (c.equals(newHead)) {
            setMode(LOSE);
            return;
        }
    }

    int applecount = mAppleList.size();
    for (int appleindex = 0; appleindex < applecount; appleindex++) {
        Coordinate c = mAppleList.get(appleindex);
        if (c.equals(newHead)) {
            mAppleList.remove(c);
            addRandomApple();

            mScore++;
            mMoveDelay *= 0.9;

            growSnake = true;
        }
    }

    mSnakeTrail.add(0, newHead);
    if (!growSnake) {
        mSnakeTrail.remove(mSnakeTrail.size() - 1);
    }

    int index = 0;
    for (Coordinate c : mSnakeTrail) {
        if (index == 0) {
            setTile(GBLOCK, c.x, c.y);
        } else {
            setTile(YBLOCK, c.x, c.y);
        }
        index++;
    }

}

private class Coordinate {
    public int x;
    public int y;
```

```
        public Coordinate(int newX, int newY) {
            x = newX;
            y = newY;
        }

        public boolean equals(Coordinate other) {
            if (x == other.x && y == other.y) {
                return true;
            }
            return false;
        }

        @Override
        public String toString() {
            return "Coordinate: [" + x + "," + y + "]";
        }
    }

}
```