DISTRIBUTED SYSTEMS LAB PROGRAMS

1. FTP implementation

FTPSERVER.java

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

public class FTPSERVER {
    public static void main(String args[]) throws Exception {
        ServerSocket serverSocket = new ServerSocket(6789);
        System.out.println("FTP Server Started on Port 6789");
        while (true) {
            System.out.println("Waiting for Connection ...");
            Socket socket = serverSocket.accept();
            new transferfile(socket);
        }
    }
}

class transferfile extends Thread {
    Socket clientSocket;
    DataInputStream din;
    DataOutputStream dout;

    transferfile(Socket socket) {
        try {
            clientSocket = socket;
            din = new DataInputStream(clientSocket.getInputStream());
    dout = new DataOutputStream(clientSocket.getOutputStream());
            System.out.println("FTP Client Connected ...");
            start();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    void SendFile() throws Exception {
        String filename = din.readUTF();
        File file = new File(filename);
        if (!file.exists()) {
            dout.writeUTF("File Not Found");
            return;
```

```
    } else {
        dout.writeUTF("READY");
        FileInputStream fin = new FileInputStream(file);
        int ch;
        do {
            ch = fin.read();
            dout.writeUTF(String.valueOf(ch));
        } while (ch != -1);
        fin.close();
        dout.writeUTF("File Sent Successfully");
    }
}

void ReceiveFile() throws Exception {
    String filename = din.readUTF();
    File file = new File(filename);
    if (file.exists()) {
        dout.writeUTF("File Already Exists");
    } else {
        dout.writeUTF("READY");
        FileOutputStream fout = new FileOutputStream(file);
        int ch;
        String temp;
        do {
            temp = din.readUTF();
            ch = Integer.parseInt(temp);
            if (ch != -1) {
                fout.write(ch);
            }
        } while (ch != -1);
        fout.close();
        dout.writeUTF("File Received Successfully");
    }
}

public void run() {
    while (true) {
        try {
            System.out.println("Waiting for Command ...");
            String command = din.readUTF();
            if (command.equals("GET")) {
                System.out.println("GET Command Received ...");
                SendFile();
            } else if (command.equals("SEND")) {
```

```
                System.out.println("SEND Command Received ...");
                ReceiveFile();
             } else if (command.equals("DISCONNECT")) {
              System.out.println("Disconnect Command Received ...");
                clientSocket.close();
                break;
             }
          } catch (Exception ex) {
             ex.printStackTrace();
             break;
          }
       }
    }
}
```

FTPCLIENT.java

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

class FTPCLIENT {
    public static void main(String args[]) throws Exception {
       Socket socket = new Socket("localhost", 6789);
       transferfileClient t = new transferfileClient(socket);
       t.displayMenu();
    }
}

class transferfileClient {
    Socket clientSocket;
    DataInputStream din;
    DataOutputStream dout;
    BufferedReader br;

    transferfileClient(Socket socket) {
       try {
          clientSocket = socket;
          din = new DataInputStream(clientSocket.getInputStream());
     dout = new DataOutputStream(clientSocket.getOutputStream());
     br = new BufferedReader(new InputStreamReader(System.in));
       } catch (Exception ex) {
          ex.printStackTrace();
       }
    }
```

```
void SendFile() throws Exception {
    System.out.print("Enter File Name: ");
    String filename = br.readLine();
    File file = new File(filename);
    if (!file.exists()) {
        System.out.println("File not Exists...");
        dout.writeUTF("File Not Found");
        return;
    }
    dout.writeUTF(filename);
    String response = din.readUTF();
    if (response.equals("READY")) {
        System.out.println("Sending File ...");
        FileInputStream fin = new FileInputStream(file);
        int ch;
        do {
            ch = fin.read();
            dout.writeUTF(String.valueOf(ch));
        } while (ch != -1);
        fin.close();
        System.out.println(din.readUTF());
    } else {
        System.out.println(response);
    }
}

void ReceiveFile() throws Exception {
    System.out.print("Enter File Name: ");
    String filename = br.readLine();
    dout.writeUTF(filename);
    String response = din.readUTF();
    if (response.equals("READY")) {
        FileOutputStream fout = new FileOutputStream(filename);
        int ch;
        String temp;
        do {
            temp = din.readUTF();
            ch = Integer.parseInt(temp);
            if (ch != -1) {
                fout.write(ch);
            }
        } while (ch != -1);
        fout.close();
        System.out.println(din.readUTF());
```

```
        } else {
            System.out.println(response);
        }
    }

    public void displayMenu() throws Exception {
        while (true) {
            System.out.println("[ MENU ]");
            System.out.println("1. Send File");
            System.out.println("2. Receive File");
            System.out.println("3. Exit");
            System.out.print("Enter Choice: ");
            int choice = Integer.parseInt(br.readLine());
            if (choice == 1) {
                dout.writeUTF("SEND");
                SendFile();
            } else if (choice == 2) {
                dout.writeUTF("GET");
                ReceiveFile();
            } else if (choice == 3) {
                dout.writeUTF("DISCONNECT");
                System.out.println("Disconnecting...");
                break;
            }
        }
    }
}
```

Server Output:

```
FTP Server Started on Port 6789
Waiting for Connection ...
FTP Client Connected ...
Waiting for Command ...
    SEND Command Received ...
File Received Successfully
Waiting for Command ...
    GET Command Received ...
File Sent Successfully
Waiting for Command ...
    Disconnect Command Received ...
```

Client Output:

```
[ MENU ]
1. Send File
2. Receive File
3. Exit
Enter Choice:

Enter Choice: 1
Enter File Name: sample.txt
Sending File ...
File Received Successfully

Enter Choice: 2
Enter File Name: received.txt
Receiving File...
File Sent Successfully

Enter Choice: 3
Disconnecting...
```

2. Iterative Name Server Implementation Using Java

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("sun.jdbc.odbc.JdbcOdbcDriver");
            con = DriverManager.getConnection("jdbc:odbc:myserver",
"system", "manager");
          smt = con.createStatement();
          while (status) {
             s = "\0";
             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();
                  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 {
                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 + "')");
             }
             ip = ip.substring(0, ip.length() - 1).trim();
          }
          System.out.println("IP address is: " + ip);
          con.close();
       } catch (Exception e) {
          System.err.println(e);
       }
```

```
   }
}
```

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("sun.jdbc.odbc.JdbcOdbcDriver");
           con = DriverManager.getConnection("jdbc:odbc:myserver",
"system", "manager");
         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();
            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(rs.getString(2));
               ps.println(s);
            } else {
```

```
                ps.println("Illegal address, please 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("sun.jdbc.odbc.JdbcOdbcDriver");
           con = DriverManager.getConnection("jdbc:odbc:myserver",
"system", "manager");
         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();
            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, please 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("sun.jdbc.odbc.JdbcOdbcDriver");
           con = DriverManager.getConnection("jdbc:odbc:myserver",
"system", "manager");
         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();
            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, please check the spelling
again");
               con.close();
            }
         }
      } catch (Exception e) {
         System.err.println(e);
      }
   }
}
```

Output:

Client.java

Enter name to resolve: www.google.com
IP address is: 192.168.1.10

Server1.java

Connection established with client...
Querying root server for domain google...
Forwarding request to Server3 (Google TLD server)...
Sending response to client: Port 5125, IP 192.168.1.10

Server2.java

Connection established with client...
Querying root server for domain yahoo...
Sending response to client: Port 5124, IP 192.168.1.20

Server3.java

Connection established with client...
Querying Google TLD database...

Sending response: Port 5125, IP 192.168.1.10

3. Chat Server Implementation

ChatServer.java

```
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("Error starting server: " + e);
      }
   }

   public void run() {
      while (thread != null) {
         try {
            System.out.println("Waiting for a client ...");
            addThread(server.accept());
         } catch (IOException e) {
            System.out.println("Error accepting client connection: " +
e);
         }
      }
   }

   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);
        }

        // Broadcast message to all clients
        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]));
    }
}

class ChatServerThread extends Thread {
    private ChatServer server = null;
    private Socket socket = null;
    private PrintWriter out = null;
    private BufferedReader in = null;
    private int ID = -1;

    public ChatServerThread(ChatServer server, Socket socket) {
        this.server = server;
        this.socket = socket;
        this.ID = socket.getPort();
    }

    public int getID() {
        return ID;
    }

    public void open() throws IOException {
        out = new PrintWriter(socket.getOutputStream(), true);
                                        in      =      new      BufferedReader(new
InputStreamReader(socket.getInputStream()));
    }
```

```
    public void run() {
        String input;
        try {
            while ((input = in.readLine()) != null) {
                server.handle(ID, input);
            }
        } catch (IOException e) {
            System.out.println("Error reading from client: " + e);
        }
    }

    public void send(String message) {
        out.println(message);
    }

    public void close() throws IOException {
        if (socket != null)
            socket.close();
    }

    public void stop() {
        try {
            close();
        } catch (IOException e) {
            System.out.println("Error closing socket: " + e);
        }
    }
}
```

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 PrintStream Out = null;
    private int ID = -1;

    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 PrintStream Out = null;
private Thread thread = 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);
      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 (IOException e) {
      System.out.println("Error: " + e.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("Goodbye. 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: " + e);
        }
        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 getting 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:

ChatServer

```
java ChatServer 5000
Binding to port 5000, please wait...
Server started:
ServerSocket[addr=0.0.0.0/0.0.0.0,port=5000,localport=5000]
Waiting for a client...
```

FirstClient

```
java ChatClient localhost 5000
Establishing connection, please wait...
Connected:
Socket[addr=localhost/127.0.0.1,port=5000,localport=<some_port
>]
```

SecondClient

```
java ChatClient localhost 5000
Establishing connection, please wait...
Connected:
Socket[addr=localhost/127.0.0.1,port=5000,localport=<some_othe
r_port>]
```

Client1
Hello, this is Client 1.
Hello, this is Client 2.

Client2
Hello, this is Client 1.
Hello, this is Client 2.

Server
Waiting for a client...
Client accepted:
Socket[addr=localhost/127.0.0.1,port=<port>,localport=5000]
Waiting for a client...
Client accepted:
Socket[addr=localhost/127.0.0.1,port=<other_port>,localport=5000]
1: Hello, this is Client 1.
2: Hello, this is Client 2.

4.Understanding Working of NFS

Servers:

File Sharing Servers:

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 & Windows heterogeneous networks.

Note:

1. NFS is used in LAN because the bandwidth for NFS is higher compared to FTP and Samba.
2. FTP is used in WAN because it 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`
   Files: aa, bb, cc
2. Create new files inside the share directory:
   Files: aa, bb, cc
3. **Packages:**
   - i) **nfs-utils**: to install or uninstall the nfs-utils package.
   - ii) **portmap**: (please do not uninstall portmap).
4. **Services:**
   - nfs
   - portmap
5. **Daemons:**
   - nfsd
   - quotad
   - mountd
6. **Main Configuration File:**
   `/etc/exports`

Experiment:

1. Make a directory named nfsshare with files aa, bb, cc:

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

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

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

3. 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)
```

4. If the package is not installed, there are two ways to install it:
   - **Install from FTP:**

```
# ping the server
# rpm -ivh
ftp://192.168.0.250:/pub.RedHat/RPMS/nfs* --
force --aid
```

- **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
```

5. After installing the nfs-utils package, create the 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 IP address and number of systems connected in the network).

6. After installing the services, enter the command to restart:

```
# services nfs restart
```

   Note: Execute this command twice because the first time it will show "failed" and the second time it will show "ok".

7. Access to NFS share from the client:

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

   Note: In the client machine, enter the server IP address.

Notes:

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 a terminal connection in Windows.
3. In NFS, all files and directories are by default in read-only mode.

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

5.Write a program to implement date service

DateServer.java

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

class DateServer {
   public static void main(String args[]) throws Exception {
      ServerSocket s = new ServerSocket(5217);
      while (true) {
         System.out.println("Waiting For Connection ...");
         Socket soc = s.accept();
         // Missing code to print the date
            PrintWriter out = new PrintWriter(soc.getOutputStream(),
true);
         Date date = new Date();
         out.println(date.toString()); // Sends the date to the client
         soc.close();
      }
   }
}
```

DateClient.java

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

class DateClient {
   public static void main(String args[]) throws Exception {
      Socket soc = new Socket(InetAddress.getLocalHost(), 5217);
                  BufferedReader in = new BufferedReader(new
InputStreamReader(soc.getInputStream()));
         System.out.println(in.readLine()); // Prints the date received
from the server
   }
}
```

Output:

Server

```
Waiting For Connection ...
```

Client

```
Fri Jan 08 17:05:42 IST 2025
```

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

WC_Mapper.java

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();

            public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
         word.set(tokenizer.nextToken());
         output.collect(word, one);
      }
   }
}
```

WC_Reducer.java

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WC_Reducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
```

```
        public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter)
throws IOException {
        int sum = 0;
        while (values.hasNext()) {
           sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
```

WC_Runner.java

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;

public class WC_Runner {
    public static void main(String[] args) throws IOException {

        JobConf conf = new JobConf(WC_Runner.class);
        conf.setJobName("WordCount");

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
```

```
}
```

Output:

```
hadoop fs -put wordcountFile wordCountFile

hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1

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-00000

hadoop fs -cat MRDir1/part-00000

bus 7
car 4
train 6
```

7. Develop an application using 3-tier architectures.

Presentation Tier (User Interface) using JavaFX

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class PresentationTier extends Application {
    private ApplicationLogic appLogic;

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        appLogic = new ApplicationLogic();
        primaryStage.setTitle("3-Tier App for BookSelf");
```

```
        VBox layout = new VBox(10);
        Scene scene = new Scene(layout, 300, 200);

        Button addButton = new Button("Add Book");
        Button listButton = new Button("List Books");
        ListView<String> itemList = new ListView<>();

        addButton.setOnAction(e -> {
            TextInputDialog dialog = new TextInputDialog();
            dialog.setTitle("Add Book");
            dialog.setHeaderText("Enter Book name:");
            dialog.setContentText("Book Name:");
            dialog.showAndWait().ifPresent(item -> {
                appLogic.addItem(item);
            });
        });

        listButton.setOnAction(e -> {
            itemList.getItems().setAll(appLogic.getItems());
        });

        layout.getChildren().addAll(addButton, listButton, itemList);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

Application Logic Tier (Business Logic)

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ApplicationLogic {
    private Connection conn;

    public ApplicationLogic() {
        try {
    conn = DriverManager.getConnection("jdbc:sqlite:app_db.sqlite");
            createTable();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    private void createTable() {
        try (Statement stmt = conn.createStatement()) {
                stmt.execute("CREATE TABLE IF NOT EXISTS items (id
INTEGER PRIMARY KEY, name TEXT)");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void addItem(String item) {
        String sql = "INSERT INTO items (name) VALUES (?)";
         try (PreparedStatement pstmt = conn.prepareStatement(sql))
{
            pstmt.setString(1, item);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public List<String> getItems() {
        List<String> items = new ArrayList<>();
        String sql = "SELECT name FROM items";
         try (Statement stmt = conn.createStatement(); ResultSet rs =
stmt.executeQuery(sql)) {
            while (rs.next()) {
                items.add(rs.getString("name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return items;
    }
}
```

Data Storage Tier (Data Access)

For the data storage tier, SQLite is used, and the JDBC driver needs to be added to the classpath.

Output:
Add the SQLite JDBC driver (sqlite-jdbc-3.36.0.3.jar) to your project's classpath.

Compile the Java classes and ensure all necessary JAR files (e.g., SQLite JDBC driver) are included.
Run the PresentationTier class to start the GUI. The window will display: "Add Book" button for adding a new book to the database.
"List Books" button to retrieve and display the list of books stored in the database.

Window Title: "3-Tier App for BookSelf"
Buttons:

- "Add Book" will open a dialog box for entering the book name
- "List Books" will display the list of books from the SQLite database.