1. Implement

a. Implement Echo client-server application in JAVA using TCP.

b. Implement a concurrent daytime client-server application in JAVA.

a.

Tcp Client

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

class TcpClient {
    public static void main(String[] args) throws Exception {
        System.out.println("connecting to server");
        Socket cs=new Socket("localhost",8088);

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

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

Tcp Server

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

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

b.

Client_DT

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

public class Client_DT
{
public static void main(String[] args) throws UnknownHostException, IOException
{
Socket cs=new Socket("LocalHost",5000);

System.out.println("Client"+cs.getInetAddress()+"is communicating from port No:"+cs.getPort());

BufferedReader fromserver=new BufferedReader(new InputStreamReader(cs.getInputStream()));
System.out.println(fromserver.readLine());
fromserver.close();
cs.close();
}
}
```

Server_DT

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class Server_DT
{
public static void main(String[] args)throws IOException{

ServerSocket ss=new ServerSocket(5000);
System.out.println("The server has reserved port No:"+ss.getLocalPort()+" for this Service");
Socket cs=ss.accept();
System.out.println("Client with IP Address"+cs.getInetAddress()+"has
communicated via port No:"+cs.getPort());
Date d= new Date();
String s="Current Date & Time on Server is:"+d;
PrintWriter toclient =new PrintWriter(cs.getOutputStream(),true);
toclient.print(s);
toclient.close();
cs.close();
ss.close();
}
}
```

2. Implement

a. Implement Echo client-server application in JAVA using UDP.

b. Implement a concurrent daytime client-server application in JAVA.

a.

UDPClient

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

class UDPClient{
    public static void main(String[] args) throws Exception {
        byte[] buff=new byte[1024];
        DatagramSocket ds = new DatagramSocket(8089);
        DatagramPacket p=new DatagramPacket(buff,buff.length);

        BufferedReader br=new BufferedReader(new InputStreamReader(
            System.in));
        System.out.print("Enter your name:");
        String msg = br.readLine();
        buff = msg.getBytes();
        ds.send(new DatagramPacket(buff,buff.length,
InetAddress.getLocalHost(),8088));
        ds.receive(p);
        msg = new String( p.getData(),0,p.getLength()).trim();
        System.out.println("Msg received "+msg);

    }
}
```

UDPServer

```
import java.net.*;
class UDPServer{
    public static void  main(String[] args) throws Exception{
        byte buff[]=new byte[1024];
        DatagramSocket ds =new DatagramSocket(8088);
        DatagramPacket p=new DatagramPacket(buff,buff.length);
        System.out.println("Server ready :");

        ds.receive(p);
        String msg = new String( p.getData(),0,p.getLength()).trim();
        String str = "Hello "+new String(buff);
        buff = str.getBytes();
        ds.send(new
DatagramPacket(buff,buff.length,InetAddress.getLocalHost(),8089));
        System.out.println("Msg received "+msg);
    }
}
```

b.

Client_DT

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

public class Client_DT
{
public static void main(String[] args) throws UnknownHostException, IOException
{
Socket cs=new Socket("LocalHost",5000);

System.out.println("Client"+cs.getInetAddress()+"is communicating from port
No:"+cs.getPort());

BufferedReader fromserver=new BufferedReader(new
InputStreamReader(cs.getInputStream()));
System.out.println(fromserver.readLine());
fromserver.close();
cs.close();
}
}
```

Server_DT

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class Server_DT
{
public static void main(String[] args)throws IOException{

ServerSocket ss=new ServerSocket(5000);
System.out.println("The server has reserved port No:"+ss.getLocalPort()+" for
this Service");
Socket cs=ss.accept();
System.out.println("Client with IP Address"+cs.getInetAddress()+"has
communicated via port No:"+cs.getPort());
Date d= new Date();
String s="Current Date & Time on Server is:"+d;
PrintWriter toclient =new PrintWriter(cs.getOutputStream(),true);
toclient.print(s);
toclient.close();
cs.close();
ss.close();
}
}
```

3. Write a program to demonstrate Rikart-Agrawal Mutex (RAM) Mutual Exclusion in a distributed environment.

```
import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;


enum MessageType {

    REQUEST,

    REPLY

}


class Message {

    private final MessageType type;

    private final int senderId;


    public Message(MessageType type, int senderId) {

        this.type = type;

        this.senderId = senderId;

    }


    public MessageType getType() {

        return type;

    }


    public int getSenderId() {

        return senderId;
    }
}

class Node {
    private final int nodeId;
    private boolean requestingCriticalSection;
    private int repliesReceived;
    private final List<Message> deferredQueue;
    private final Lock lock;

    public Node(int nodeId) {
        this.nodeId = nodeId;
        this.requestingCriticalSection = false;
        this.repliesReceived = 0;
        this.deferredQueue = new ArrayList<>();
        this.lock = new ReentrantLock();
    }

    public void requestCriticalSection() {
        lock.lock();
        try {
            requestingCriticalSection = true;
            repliesReceived = 0;

            for (int i = 0; i < Main.NUM_NODES; i++) {
                if (i != nodeId) {
                    Message message = new Message(MessageType.REQUEST, nodeId);
                    Main.sendMessage(nodeId, i, message);
                }
            }
        } finally {
            lock.unlock();
        }
```

```
}

public void receiveRequest(int senderId) {
   lock.lock();
   try {
      if (!requestingCriticalSection || (repliesReceived > 0)) {
         Message replyMessage = new Message(MessageType.REPLY, nodeId);
         Main.sendMessage(nodeId, senderId, replyMessage);
      } else {
         deferredQueue.add(new Message(MessageType.REQUEST, senderId));
      }
   } finally {
      lock.unlock();
   }
}

public void receiveReply() {
   lock.lock();
   try {
      repliesReceived++;
      if (repliesReceived == Main.NUM_NODES - 1) {
         enterCriticalSection();
      }
   } finally {
      lock.unlock();
   }
}

public void releaseCriticalSection() {
   lock.lock();
   try {
      requestingCriticalSection = false;
```

```
            for (Message message : deferredQueue) {
                Main.sendMessage(nodeId, message.getSenderId(), new Message(MessageType.REPLY, nodeId));
            }

            deferredQueue.clear();
        } finally {
            lock.unlock();
        }
    }

    private void enterCriticalSection() {
        System.out.println("Node " + nodeId + " is entering the critical section.");
        // Perform operations in the critical section
        System.out.println("Node " + nodeId + " is leaving the critical section.");

        for (int i = 0; i < Main.NUM_NODES; i++) {
            if (i != nodeId) {
                Main.sendMessage(nodeId, i, new Message(MessageType.REPLY, nodeId));
            }
        }
    }
}

public class Main {
    public static final int NUM_NODES = 3;
    private static final List<Node> nodes = new ArrayList<>();

    public static void main(String[] args) {
        for (int i = 0; i < NUM_NODES; i++) {
            nodes.add(new Node(i));
        }

        // Simulate a scenario where a node requests access to the critical section
        nodes.get(0).requestCriticalSection();
    }

    public static void sendMessage(int senderId, int receiverId, Message message) {
        // Simulate sending a message from one node to another
        System.out.println("Node " + senderId + " sends a " + message.getType() +
                " message to Node " + receiverId);
        if (message.getType() == MessageType.REQUEST) {
            nodes.get(receiverId).receiveRequest(senderId);
        } else if (message.getType() == MessageType.REPLY) {
            nodes.get(receiverId).receiveReply();
        }
    }
}
```

4. Develop a distributed chat server using TCP sockets in JAVA for a Single server- Single client environment.

ServerApp

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


public class ServerApp implements Runnable{

    /**
     * @param args
     */
    public static Socket s=null;
    public static int i=1;
    public static String clientName = "";
    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub
        ServerSocket ss = new ServerSocket(8089);
        ServerApp sa = new ServerApp();
        Thread t;
        try{
            while(true){
                System.out.println("Waiting for client "+i);
                s = ss.accept();
                i++;
                t = new Thread(sa);
                t.start();

            }
        }catch (Exception e) {
            // TODO: handle exception
        }
        finally{
            ss.close();
        }
    }
    @Override
    public void run() {
        // TODO Auto-generated method stub

        try
        {
            InputStream is = s.getInputStream();
            byte[] b = new byte[1024];
            is.read(b);
            clientName="";
            clientName = new String(b).trim();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        new ChatGUI(s,clientName);
    }
}
```

ClientApp

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


public class ClientApp {

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

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

}
```

ChatGUI

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

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

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

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

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

    ChatGUI(Socket x, String str) {
        s = x;
        title = str;
        button = new JButton("SEND");
        ta1 = new JTextArea(5, 20);
        ta2 = new JTextArea(5, 20);
        ta1.setEditable(false);
        scrollPane1 = new JScrollPane(ta1);
        scrollPane2 = new JScrollPane(ta2);
        setLayout(new FlowLayout());
        add(scrollPane1);
        add(scrollPane2);
        add(button);
```

```java
        button.addActionListener(this);
        setSize(300, 300);
        setVisible(true);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setTitle("Messenger " + title);
        try {
            is = s.getInputStream();
            os = s.getOutputStream();
        } catch (IOException ioe) {
        }

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

    @SuppressWarnings("deprecation")
    public void chat() throws Exception {
        while (true) {
            try {
                byte data[] = new byte[50];
                is.read(data);
                msg = new String(data).trim();
                ta1.append(title+": " + msg + "\n");
            } catch (SocketException se) {
                JOptionPane.showMessageDialog(this, "Disconnected from 
"+title);
                this.dispose();
                Thread.currentThread().stop();
            }
        }
    }

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

5. Implement a distributed chat server using TCP sockets in JAVA for a Single server-

Multiple client environment.

ChatClient

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ChatClient {
    public static void main(String[] args) {
        new ChatClient().startClient();
    }

    public void startClient() {
        try {
            Socket socket = new Socket("localhost", 12345);
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            BufferedReader consoleInput = new BufferedReader(new
InputStreamReader(System.in));

            // Read and print messages from the server
            new Thread(() -> {
                try {
                    String serverMessage;
                    while ((serverMessage = in.readLine()) != null) {
                        System.out.println(serverMessage);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }).start();

            // Send messages to the server
            String userInput;
            while ((userInput = consoleInput.readLine()) != null) {
                out.println(userInput);
            }

            out.close();
            in.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

ChatServer

```java
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class ChatServer {
    private static final int PORT = 12345;
    private List<ClientHandler> clients = new ArrayList<>();

    public static void main(String[] args) {
        new ChatServer().startServer();
    }

    public void startServer() {
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Server is running on port " + PORT);

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected: " + clientSocket);

                ClientHandler clientHandler = new ClientHandler(clientSocket,
this);
                clients.add(clientHandler);
                new Thread(clientHandler).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void broadcastMessage(String message, ClientHandler sender) {
        for (ClientHandler client : clients) {
            if (client != sender) {
                client.sendMessage(sender.getClientName() + ": " + message);
            }
        }
    }

    public void removeClient(ClientHandler client) {
        clients.remove(client);
        System.out.println("Client disconnected: " + client.getClientSocket());
    }
}
```

ClientHandler

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientHandler implements Runnable {
    private Socket clientSocket;
    private ChatServer server;
    private PrintWriter out;
    private BufferedReader in;
    private String clientName;

    public ClientHandler(Socket socket, ChatServer server) {
        this.clientSocket = socket;
        this.server = server;
    }

    public String getClientName() {
        return clientName;
    }

    public Socket getClientSocket() {
        return clientSocket;
    }

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

    @Override
    public void run() {
        try {
            out = new PrintWriter(clientSocket.getOutputStream(), true);
            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

            out.println("Enter your name:");
            clientName = in.readLine();
            out.println("Welcome to the chat, " + clientName + "!");

            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                if (inputLine.equalsIgnoreCase("exit")) {
                    break;
                }
                server.broadcastMessage(inputLine, this);
            }

            server.removeClient(this);
            in.close();
            out.close();
            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

6. Write a program for Remote Method Invocation (RMI) mechanism for accessing remote methods (ADD, SUB, MUL & DIV).

Calculator.java

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

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

CalculatorServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class CalculatorServer extends UnicastRemoteObject implements Calculator {

    protected CalculatorServer() throws RemoteException {
        super();
    }

    @Override
    public int add(int a, int b) throws RemoteException {
        return a + b;
    }

    @Override
    public int subtract(int a, int b) throws RemoteException {
        return a - b;
    }

    @Override
    public int multiply(int a, int b) throws RemoteException {
        return a * b;
    }

    @Override
    public int divide(int a, int b) throws RemoteException {
        if (b == 0) {
            throw new RemoteException("Cannot divide by zero");
        }
        return a / b;
    }

    public static void main(String[] args) {
        try {
            CalculatorServer calculatorServer = new CalculatorServer();
            Naming.rebind("CalculatorService", calculatorServer);
            System.out.println("Calculator Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

CalculatorClient.java

```
import java.rmi.Naming;


public class CalculatorClient {
    public static void main(String[] args) {
        try {
            Calculator calculator = (Calculator) Naming.lookup("rmi://localhost/CalculatorService");


            int a = 10;
            int b = 5;


            System.out.println("Addition: " + calculator.add(a, b));
            System.out.println("Subtraction: " + calculator.subtract(a, b));
            System.out.println("Multiplication: " + calculator.multiply(a, b));
            System.out.println("Division: " + calculator.divide(a, b));


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Compilation

```
javac Calculator.java
javac CalculatorServer.java
javac CalculatorClient.java


start rmiregistry
java CalculatorServer
 java CalculatorClient
```

7.Write a program to calculate Factorial of the given number using the Remote Method Invocation (RMI) mechanism.

FactorialCalculator.java

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


public interface FactorialCalculator extends Remote {
    long calculateFactorial(int n) throws RemoteException;
}
```

FactorialServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;


public class FactorialServer extends UnicastRemoteObject implements FactorialCalculator {


    protected FactorialServer() throws RemoteException {
        super();
    }


    @Override
    public long calculateFactorial(int n) throws RemoteException {
        if (n < 0) {
            throw new RemoteException("Factorial is not defined for negative numbers.");
        }
        if (n == 0 || n == 1) {
            return 1;
        }
        return n * calculateFactorial(n - 1);
    }


    public static void main(String[] args) {
        try {
            FactorialServer factorialServer = new FactorialServer();
            Naming.rebind("FactorialService", factorialServer);
            System.out.println("Factorial Server is running...");
        } catch (Exception e) {
            e.printStackTrace();

        }

    }

}
```

FactorialClient.java

```
import java.rmi.Naming;


public class FactorialClient {

    public static void main(String[] args) {

        try {

            FactorialCalculator factorialCalculator = (FactorialCalculator)
Naming.lookup("rmi://localhost/FactorialService");


            int number = 5; // Change this to the desired number


            long result = factorialCalculator.calculateFactorial(number);

            System.out.println("Factorial of " + number + " is: " + result);


        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```

8. Write a program to perform Matrix NxN Multiplication using the Remote Method

Invocation (RMI) mechanism.

MatrixMultiplier.java

```
import java.rmi.Remote;

import java.rmi.RemoteException;


public interface MatrixMultiplier extends Remote {

    int[][] multiply(int[][] matrixA, int[][] matrixB) throws RemoteException;

}
```

MatrixMultiplierServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MatrixMultiplierServer extends UnicastRemoteObject implements MatrixMultiplier {

  protected MatrixMultiplierServer() throws RemoteException {
    super();
  }

  @Override
  public int[][] multiply(int[][] matrixA, int[][] matrixB) throws RemoteException {
    int rowsA = matrixA.length;
    int colsA = matrixA[0].length;
    int colsB = matrixB[0].length;

    int[][] result = new int[rowsA][colsB];

    for (int i = 0; i < rowsA; i++) {
      for (int j = 0; j < colsB; j++) {
        for (int k = 0; k < colsA; k++) {
          result[i][j] += matrixA[i][k] * matrixB[k][j];
        }
      }
    }

    return result;
  }

  public static void main(String[] args) {
    try {
      MatrixMultiplierServer matrixMultiplierServer = new MatrixMultiplierServer();
      Naming.rebind("MatrixMultiplierService", matrixMultiplierServer);
```

```
            System.out.println("Matrix Multiplier Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

MatrixMultiplierClient.java

```
import java.rmi.Naming;

public class MatrixMultiplierClient {
    public static void main(String[] args) {
        try {
            MatrixMultiplier matrixMultiplier = (MatrixMultiplier)
Naming.lookup("rmi://localhost/MatrixMultiplierService");

            int[][] matrixA = {
                {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9}
            };

            int[][] matrixB = {
                {9, 8, 7},
                {6, 5, 4},
                {3, 2, 1}
            };

            int[][] result = matrixMultiplier.multiply(matrixA, matrixB);

            System.out.println("Resultant Matrix:");
            for (int[] row : result) {
                for (int value : row) {
```

```
                System.out.print(value + " ");
            }
            System.out.println();
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

9. Write a program for displaying Fibonacci Series using the Remote Method Invocation (RMI) mechanism.

FibonacciCalculator.java

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

public interface FibonacciCalculator extends Remote {
    int calculateFibonacci(int n) throws RemoteException;
}
```

FibonacciCalculatorServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class FibonacciCalculatorServer extends UnicastRemoteObject implements FibonacciCalculator
{

    protected FibonacciCalculatorServer() throws RemoteException {
        super();
    }

    @Override
    public int calculateFibonacci(int n) throws RemoteException {
        if (n <= 1) {
            return n;
        }
        return calculateFibonacci(n - 1) + calculateFibonacci(n - 2);
    }

    public static void main(String[] args) {
        try {
            FibonacciCalculatorServer fibonacciCalculatorServer = new FibonacciCalculatorServer();
            Naming.rebind("FibonacciCalculatorService", fibonacciCalculatorServer);
            System.out.println("Fibonacci Calculator Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

FibonacciCalculatorClient.java

```
import java.rmi.Naming;

public class FibonacciCalculatorClient {
    public static void main(String[] args) {
        try {
            FibonacciCalculator fibonacciCalculator = (FibonacciCalculator)
Naming.lookup("rmi://localhost/FibonacciCalculatorService");

            int n = 10; // Change this to the desired number

            System.out.println("Fibonacci Series:");
            for (int i = 0; i < n; i++) {
                System.out.print(fibonacciCalculator.calculateFibonacci(i) + " ");
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

10. Implement the Matrix Transportation program using the Remote Method Invocation (RMI) mechanism.

MatrixTransposer.java

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

public interface MatrixTransposer extends Remote {
    int[][] transpose(int[][] matrix) throws RemoteException;
}
```

MatrixTransposerServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MatrixTransposerServer extends UnicastRemoteObject implements MatrixTransposer {

    protected MatrixTransposerServer() throws RemoteException {
        super();
    }

    @Override
    public int[][] transpose(int[][] matrix) throws RemoteException {
        int rows = matrix.length;
        int cols = matrix[0].length;

        int[][] result = new int[cols][rows];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[j][i] = matrix[i][j];
            }
        }

        return result;
    }

    public static void main(String[] args) {
        try {
            MatrixTransposerServer matrixTransposerServer = new MatrixTransposerServer();
            Naming.rebind("MatrixTransposerService", matrixTransposerServer);
            System.out.println("Matrix Transposer Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

MatrixTransposerClient.java

```
import java.rmi.Naming;

public class MatrixTransposerClient {
    public static void main(String[] args) {
        try {
            MatrixTransposer matrixTransposer = (MatrixTransposer)
Naming.lookup("rmi://localhost/MatrixTransposerService");

            int[][] matrix = {
                {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9}
            };

            System.out.println("Original Matrix:");
            printMatrix(matrix);

            int[][] transposedMatrix = matrixTransposer.transpose(matrix);

            System.out.println("\nTransposed Matrix:");
            printMatrix(transposedMatrix);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int value : row) {
                System.out.print(value + " ");
            }
            System.out.println();
        }
    }
}
```

11. Write a program to perform Inverse of a Matrix using the Remote Method Invocation (RMI) mechanism.

```
// MatrixInverseServerMain.java

import java.rmi.Naming;

public class MatrixInverseServerMain {
    public static void main(String[] args) {
        try {
            MatrixInverseServer server = new MatrixInverseServerImpl();
            Naming.rebind("MatrixInverseServer", server);

            System.out.println("Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
// MatrixInverseClient.java

import java.rmi.Naming;
import java.util.Scanner;

public class MatrixInverseClient {
    public static void main(String[] args) {
        try {
            MatrixInverseServer server = (MatrixInverseServer)
Naming.lookup("rmi://localhost/MatrixInverseServer");

            // Get matrix size from the user
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter the size of the square matrix: ");
            int size = scanner.nextInt();

            // Get matrix elements from the user
            System.out.println("Enter the elements of the matrix:");
            double[][] matrix = new double[size][size];
            for (int i = 0; i < size; i++) {
                for (int j = 0; j < size; j++) {
                    System.out.print("Enter element at position (" + (i + 1) +
", " + (j + 1) + "): ");
                    matrix[i][j] = scanner.nextDouble();
                }
            }

            // Call the remote function
            double[][] result = server.inverseMatrix(matrix);

            // Display the result
            System.out.println("Original Matrix:");
            printMatrix(matrix);
            System.out.println("\nInverse Matrix:");
            printMatrix(result);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void printMatrix(double[][] matrix) {
        for (double[] row : matrix) {
            for (double element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```

```java
// MatrixInverseServer.java

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

public interface MatrixInverseServer extends Remote {
    double[][] inverseMatrix(double[][] matrix) throws RemoteException;
}
```

```java
// MatrixInverseServerImpl.java

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MatrixInverseServerImpl extends UnicastRemoteObject implements
MatrixInverseServer {
    protected MatrixInverseServerImpl() throws RemoteException {
        super();
    }

    @Override
    public double[][] inverseMatrix(double[][] matrix) throws RemoteException {
        try {
            // Assuming the matrix is square for simplicity
            int size = matrix.length;

            // Augment the matrix with the identity matrix
            double[][] augmentedMatrix = new double[size][2 * size];
            for (int i = 0; i < size; i++) {
                for (int j = 0; j < size; j++) {
                    augmentedMatrix[i][j] = matrix[i][j];
                    augmentedMatrix[i][j + size] = (i == j) ? 1 : 0;
                }
            }

            // Apply elementary row operations to transform the left side into
the identity matrix
            for (int i = 0; i < size; i++) {
                double pivot = augmentedMatrix[i][i];
                for (int j = 0; j < 2 * size; j++) {
                    augmentedMatrix[i][j] /= pivot;
                }
                for (int k = 0; k < size; k++) {
                    if (k != i) {
                        double factor = augmentedMatrix[k][i];
                        for (int j = 0; j < 2 * size; j++) {
                            augmentedMatrix[k][j] -= factor *
augmentedMatrix[i][j];
                        }
                    }
                }
            }

            // Extract the right side (inverse) from the augmented matrix
            double[][] invMatrix = new double[size][size];
            for (int i = 0; i < size; i++) {
                for (int j = 0; j < size; j++) {
                    invMatrix[i][j] = augmentedMatrix[i][j + size];
                }
            }

            return invMatrix;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

12. Write program to search (any method) the number from a given list using the Remote Method Invocation (RMI) mechanism.

NumberSearcher.java

```
import java.rmi.Remote;

import java.rmi.RemoteException;


public interface NumberSearcher extends Remote {

   int searchNumber(int[] numbers, int target) throws RemoteException;

}
```

NumberSearcherServer.java

```
import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.Arrays;


public class NumberSearcherServer extends UnicastRemoteObject implements NumberSearcher {


   protected NumberSearcherServer() throws RemoteException {
    super();
}

@Override
public int searchNumber(int[] numbers, int target) throws RemoteException {
    Arrays.sort(numbers);

    int left = 0;
    int right = numbers.length - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (numbers[mid] == target) {
            return mid; // Return the index where the target is found
        }

        if (numbers[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return -1; // Return -1 if the target is not found
}

public static void main(String[] args) {
    try {
        NumberSearcherServer numberSearcherServer = new NumberSearcherServer();
        Naming.rebind("NumberSearcherService", numberSearcherServer);
        System.out.println("Number Searcher Server is running...");
    } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

NumberSearcherClient.java

```
import java.rmi.Naming;


public class NumberSearcherClient {
    public static void main(String[] args) {
        try {
            NumberSearcher numberSearcher = (NumberSearcher)
Naming.lookup("rmi://localhost/NumberSearcherService");


            int[] numbers = {2, 4, 7, 10, 13, 18, 21, 25, 29, 32};
            int target = 18; // Change this to the desired target number


            int result = numberSearcher.searchNumber(numbers, target);


            if (result != -1) {
                System.out.println("Number " + target + " found at index " + result);
            } else {
                System.out.println("Number " + target + " not found in the list.");
            }


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

13. Write a program to sort the given list using the Remote Method Invocation (RMI)

mechanism (any sorting technique)

ListSorter.java

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

public interface ListSorter extends Remote {
    List<Integer> sortList(List<Integer> list) throws RemoteException;
}
```

ListSorterServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Collections;
import java.util.List;

public class ListSorterServer extends UnicastRemoteObject implements ListSorter {

    protected ListSorterServer() throws RemoteException {
        super();
    }

    @Override
    public List<Integer> sortList(List<Integer> list) throws RemoteException {
        Collections.sort(list);
        return list;
    }

    public static void main(String[] args) {
        try {
            ListSorterServer listSorterServer = new ListSorterServer();
            Naming.rebind("ListSorterService", listSorterServer);
            System.out.println("List Sorter Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

ListSorterClient.java

```
import java.rmi.Naming;
import java.util.ArrayList;
import java.util.List;

public class ListSorterClient {
    public static void main(String[] args) {
        try {
            ListSorter listSorter = (ListSorter) Naming.lookup("rmi://localhost/ListSorterService");

            List<Integer> list = new ArrayList<>();
            list.add(5);
            list.add(2);
            list.add(8);
            list.add(1);
            list.add(6);

            System.out.println("Original List: " + list);

            List<Integer> sortedList = listSorter.sortList(list);

            System.out.println("Sorted List: " + sortedList);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

14. Implement the String concatenation program using the Remote Method Invocation (RMI) mechanism.

StringConcatenator.java

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

public interface StringConcatenator extends Remote {
   String concatenateStrings(String str1, String str2) throws RemoteException;
}
```

StringConcatenatorServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class StringConcatenatorServer extends UnicastRemoteObject implements StringConcatenator {

   protected StringConcatenatorServer() throws RemoteException {
      super();
   }

   @Override
   public String concatenateStrings(String str1, String str2) throws RemoteException {
      return str1 + str2;
   }

   public static void main(String[] args) {
      try {
         StringConcatenatorServer stringConcatenatorServer = new StringConcatenatorServer();
         Naming.rebind("StringConcatenatorService", stringConcatenatorServer);
         System.out.println("String Concatenator Server is running...");
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

StringConcatenatorClient.java

```
import java.rmi.Naming;


public class StringConcatenatorClient {

    public static void main(String[] args) {

        try {

            StringConcatenator stringConcatenator = (StringConcatenator)
Naming.lookup("rmi://localhost/StringConcatenatorService");


            String str1 = "Hello, ";

            String str2 = "World!";


            String result = stringConcatenator.concatenateStrings(str1, str2);


            System.out.println("Concatenated String: " + result);


        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```

15. Implement the program to reverse the given string using the Remote Method Invocation (RMI) mechanism.

```
StringReverserServer
```

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class StringReverserServer extends UnicastRemoteObject implements
StringReverser {

    protected StringReverserServer() throws RemoteException {
        super();
    }

    public String reverse(String input) throws RemoteException {
        return new StringBuilder(input).reverse().toString();
    }
    public static void main(String[] args) {
        try {
            StringReverserServer server = new StringReverserServer();
            Naming.rebind("StringReverser", server);
            System.out.println("StringReverserServer is ready.");
        } catch (Exception e) {
            System.err.println("StringReverserServer exception: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

```
StringReverserClient
```

```
import java.rmi.Naming;
import java.util.Scanner;

public class StringReverserClient {

    public static void main(String[] args) {
        try {
            StringReverser reverser = (StringReverser)
Naming.lookup("rmi://localhost/StringReverser");

            // Take user input
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a string to reverse: ");
            String input = scanner.nextLine();

            // Invoke the remote method
            String reversed = reverser.reverse(input);

            // Display the result
            System.out.println("Original string: " + input);
            System.out.println("Reversed string: " + reversed);

        } catch (Exception e) {
            System.err.println("StringReverserClient exception: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

```
StringReverser
```

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

public interface StringReverser extends Remote {
    String reverse(String input) throws RemoteException;
}
```

16.Write a program illustrating Palindrome using the Remote Method Invocation (RMI) mechanism.

Code
PalindromeService

```java
import java.rmi.Remote;

import java.rmi.RemoteException;


public interface PalindromeService extends Remote {

   boolean isPalindrome(String str) throws RemoteException;

}
```

PalindromeServiceImpl Class

```java
import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;


public class PalindromeServiceImpl extends UnicastRemoteObject implements PalindromeService {

   protected PalindromeServiceImpl() throws RemoteException {

      super();

   }


   @Override

   public boolean isPalindrome(String str) throws RemoteException {

      str = str.toLowerCase().replaceAll("[^a-zA-Z0-9]", "");

      int length = str.length();

      for (int i = 0; i < length / 2; i++) {

         if (str.charAt(i) != str.charAt(length - i - 1)) {

            return false;

         }
```

```
        }
        return true;
    }
}
```

PalindromeServer Class

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class PalindromeServer {
    public static void main(String[] args) {
        try {
            PalindromeService palindromeService = new PalindromeServiceImpl();

            LocateRegistry.createRegistry(1099);

            // Bind the remote object's stub in the registry
            Naming.rebind("PalindromeService", palindromeService);

            System.out.println("PalindromeService is ready to check palindromes.");
        } catch (Exception e) {
            System.err.println("PalindromeService exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

PalindromeClient Class

```
import java.rmi.Naming;

public class PalindromeClient {
    public static void main(String[] args) {
        try {
            PalindromeService palindromeService = (PalindromeService)
Naming.lookup("rmi://localhost/PalindromeService");


            String testString1 = "level";

            String testString2 = "hello";


            System.out.println("Is '" + testString1 + "' a palindrome? " +
palindromeService.isPalindrome(testString1));

            System.out.println("Is '" + testString2 + "' a palindrome? " +
palindromeService.isPalindrome(testString2));

        } catch (Exception e) {

            System.err.println("PalindromeClient exception: " + e.getMessage());

            e.printStackTrace();

        }

    }

}
```

compile and run

Ex 17

Aim : Develop multiple clients- single server application that uses File Transfer Protocol (FTP) using JAVA.

Code :

Server : FTPS.JAVA

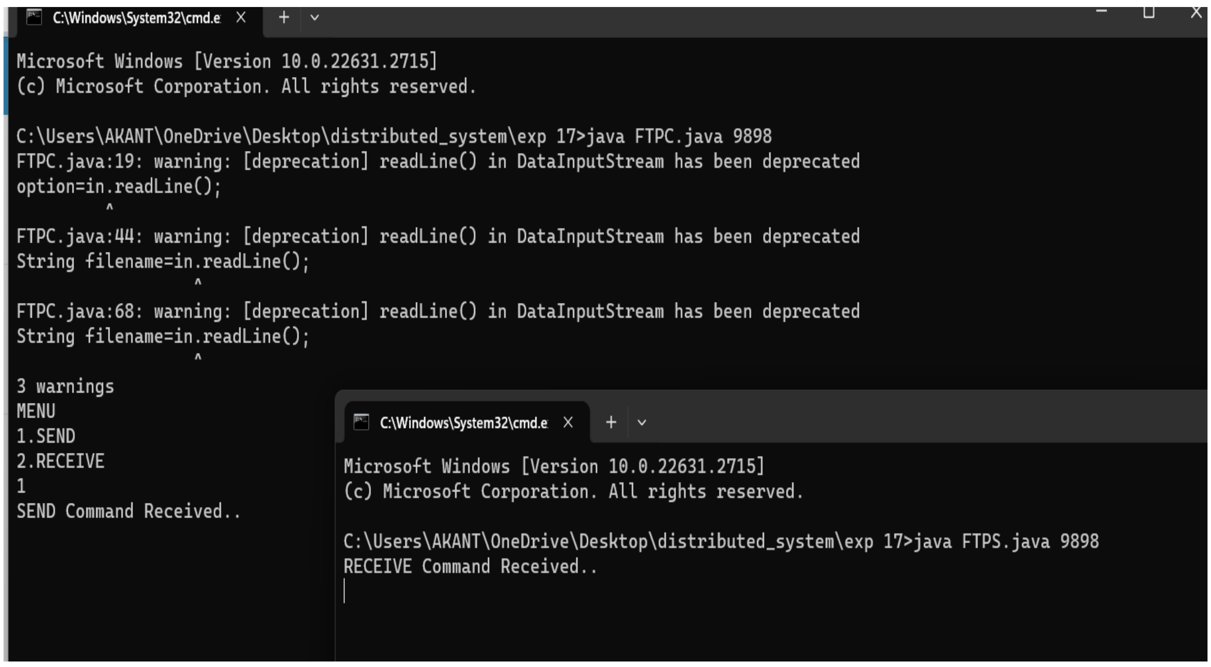
```
import java.io.*;
import java.net.*;

class FTPS {
      public static void main(String[] args) throws Exception {
            ServerSocket Sock = new ServerSocket(Integer.parseInt(args[0]));
            Socket s = Sock.accept();
            DataInputStream cin = new DataInputStream(s.getInputStream());
            DataOutputStream cout = new
DataOutputStream(s.getOutputStream());

            FTPS ftp = new FTPS();
            while (true) {

                  String option = cin.readUTF();
                  if (option.equals("SEND")) {
                        System.out.println("SEND Command Received..");
                        ftp.sendfile(s);
                  }
                  else if (option.equals("RECEIVE")) {
                        System.out.println("RECEIVE Command Received..");
                        ftp.receivefile(s);
                  }
            }
      }

      public void sendfile(Socket s) throws Exception {
            Socket ssock = s;

            DataInputStream cin = new
DataInputStream(ssock.getInputStream());
            DataOutputStream cout = new
DataOutputStream(ssock.getOutputStream());
            String filename = cin.readUTF();
            System.out.println("Reading File " + filename);
            File f = new File(filename);
            FileInputStream fin = new FileInputStream(f);
            int ch;
            do {
                  ch = fin.read();
                  cout.writeUTF(Integer.toString(ch));
            } while (ch != -1);
            fin.close();
            System.out.println("File Sent");
      }

      public void receivefile(Socket s) throws Exception {
            Socket ssock = s;

            DataInputStream cin = new
DataInputStream(ssock.getInputStream());
            DataOutputStream cout = new
DataOutputStream(ssock.getOutputStream());

            String filename = cin.readUTF();
            System.out.println("Receiving File " + filename);
            File f = new File(filename);
            FileOutputStream fout = new FileOutputStream(f);
            int ch;
            while ((ch = Integer.parseInt(cin.readUTF())) != -1) {
                  fout.write(ch);
            }
            System.out.println("Received File...");
            fout.close();
      }
}
```

Client **:** FTPC.JAVA

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

class FTPC {

      public static void main(String[] args) throws Exception {
            String option;
            DataInputStream in = new DataInputStream(System.in);
            Socket s = new Socket("localhost", Integer.parseInt(args[0]));
            System.out.println("MENU");
```

```
        System.out.println("1.SEND");
        System.out.println("2.RECEIVE");
        FTPC ftp = new FTPC();
        while (true) {
            option = in.~~readLine~~();
            if (option.equals("1")) {
                System.out.println("SEND Command Received..");
                ftp.sendfile(s);
            }

            else if (option.equals("2")) {
                System.out.println("RECEIVE Command Received..");
                ftp.receivefile(s);
            }

        }
    }

    public void sendfile(Socket s) throws Exception {
        Socket ssock = s;

        DataInputStream in = new DataInputStream(System.in);

        DataInputStream cin = new
DataInputStream(ssock.getInputStream());
        DataOutputStream cout = new
DataOutputStream(ssock.getOutputStream());

        cout.writeUTF("RECEIVE");

        String filename = in.~~readLine~~();
        System.out.println("Reading File " + filename);
        cout.writeUTF(filename);
        File f = new File(filename);
        FileInputStream fin = new FileInputStream(f);
        int ch;
        do {
            ch = fin.read();
            cout.writeUTF(String.valueOf(ch));
        } while (ch != -1);
        fin.close();
        System.out.println("File Sent");
    }

    public void receivefile(Socket s) throws Exception {
        Socket ssock = s;
        DataInputStream in = new DataInputStream(System.in);
        DataInputStream cin = new
DataInputStream(ssock.getInputStream());
        DataOutputStream cout = new
DataOutputStream(ssock.getOutputStream());

        cout.writeUTF("SEND");

        String filename = in.~~readLine~~();
        cout.writeUTF(filename);
        System.out.println("Receiving File " + filename);
        File f = new File(filename);
        FileOutputStream fout = new FileOutputStream(f);
        int ch;
        do {
            ch = Integer.parseInt(cin.readUTF());
            if (ch != -1)
                fout.write(ch);
        } while (ch != -1);
        System.out.println("Received File...");
```

```
                fout.close();
        }
}
```



```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AKANT\OneDrive\Desktop\distributed_system\exp 17>java FTPC.java 9898
FTPC.java:19: warning: [deprecation] readLine() in DataInputStream has been deprecated
option=in.readLine();
         ^
FTPC.java:44: warning: [deprecation] readLine() in DataInputStream has been deprecated
String filename=in.readLine();
                  ^
FTPC.java:68: warning: [deprecation] readLine() in DataInputStream has been deprecated
String filename=in.readLine();
                  ^
3 warnings
MENU
1.SEND
2.RECEIVE
1
SEND Command Received..
```

```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AKANT\OneDrive\Desktop\distributed_system\exp 17>java FTPS.java 9898
RECEIVE Command Received..
```

Exp 21

Aim : Write a program to increment Counter in Shared memory using JAVA.

Code :

```
class SharedMemory extends Thread {
            static int i=0;
            void increment()
            {
               i=i+1;
               System.out.println("shared memory after
increment "+i);
            }
            @Override
            public void run()
            {
               increment();
            }
}
class IncCounter {
            public static void main(String[] args) throws
InterruptedException
            {  Thread t1 = new SharedMemory();
               Thread t2 = new SharedMemory();
               Thread t3 = new SharedMemory();
               t1.start();
               t1.join();
               t2.start();
               t2.join();
               t3.start();
            }
}
```

Exp 22

Aim : Write a program to simulate Distribute Mutual Exclusion.

Code:

```
import java.util.*;

public class MutualExclusion {

      public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            int opt0, opt1;
            int p1 = 1;
            int p2 = 2;
            int p3 = 3;
            int flag = 0;
            int cs = 0;
            Queue<Integer> q = new LinkedList<>();
            do {
                  System.out.println("....menu...");
                  System.out.println("1.Request the critical section");
                  System.out.println("2.Release the critical section");
                  System.out.println("3.Exit");
                  opt0 = sc.nextInt();
                  switch (opt0) {
                  case 1: {
                        System.out.println("Select the process.");
                        System.out.println("1.p1");
                        System.out.println("2.p2");
                        System.out.println("3.p3");
                        opt1 = sc.nextInt();
                        switch (opt1) {
                        case 1: {
                              if (flag == 0) {
                                    cs = 1;
                                    flag = 1;
                              } else {
                                    System.out.println("process p" + cs +
"is already in critical section.");
                                    q.add(p1);
                              }

                              System.out.println("System Status:");
                              System.out.println("critical section is
occupoied by:" + cs);
                              System.out.println("process waiting is: " +
q);
                              break;
                        }
                        case 2: {
                              if (flag == 0) {
                                    cs = 2;
                                    flag = 1;
                              } else {
                                    System.out.println("process p" + cs +
"is already in critical section.");
                                    q.add(p2);
                              }
                              System.out.println("System Status:");
                              System.out.println("critical section is
occupoied by:" + cs);
```

```
                    System.out.println("process waiting is: " +
q);
                    break;
                }
                case 3: {
                    if (flag == 0) {
                        cs = 3;
                        flag = 1;
                    } else {
                        System.out.println("process p" + cs +
"is already in critical section.");
                        q.add(p3);
                    }
                    System.out.println("System Status:");
                    System.out.println("critical section is
occupoied by:" + cs);
                    System.out.println("process waiting is: " +
q);
                    break;
                }
                }
                break;
            }
            case 2: {
                System.out.println("the process p" + cs + "is
removed from section.");
                if (!q.isEmpty()) {
                    cs = q.peek();
                    q.remove();
                    System.out.println("System status:");
                    System.out.println("Critical Section occupied
by p" + cs);
                } else {
                    System.out.println("No Process is waiting in
the queue");
                    flag = 0;
                }
            }

            case 3: {
                break;
            }
            }
        } while (3 != opt0);
    }

}
```

18. Develop multiple clients- single server application that uses File Transfer Protocol (FTP) using JAVA.

FTP Server

import org.apache.commons.net.ftp.FTPClient;

import org.apache.commons.net.ftp.FTPFile;

import java.io.*;

import java.net.ServerSocket;

```
import java.net.Socket;

public class FTPMultiClientServer {

   public static void main(String[] args) {
      FTPMultiClientServer ftpServer = new FTPMultiClientServer();
      ftpServer.startServer();
   }

   private void startServer() {
      try {
         ServerSocket serverSocket = new ServerSocket(21);
         System.out.println("FTP Server started. Waiting for connections...");

         while (true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("Accepted connection from " + clientSocket.getInetAddress());

            Thread clientThread = new Thread(() -> handleClient(clientSocket));
            clientThread.start();
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
   }

   private void handleClient(Socket clientSocket) {
      try {
         FTPClient ftpClient = new FTPClient();
         ftpClient.connect("localhost", 21);

         BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
         PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);
```

```
writer.println("220 Welcome to the FTP server");

String username = null;
String password = null;

while (true) {
    String request = reader.readLine();
    System.out.println("Received request: " + request);

    if (request.startsWith("USER ")) {
        username = request.substring(5);
        writer.println("331 Password required for " + username);
    } else if (request.startsWith("PASS ")) {
        password = request.substring(5);
        writer.println("230 User logged in");
        break;
    }
}

while (true) {
    String request = reader.readLine();
    System.out.println("Received request: " + request);

    if (request.equals("PWD")) {
        writer.println("257 \"/\" is the current directory");
    } else if (request.equals("QUIT")) {
        writer.println("221 Goodbye");
        break;
    } else if (request.startsWith("LIST")) {
        writer.println("150 Opening ASCII mode data connection");
        handleListCommand(ftpClient, writer);
        writer.println("226 Transfer complete");
```

```
                } else {
                    writer.println("502 Command not implemented");
                }
            }

            ftpClient.logout();
            ftpClient.disconnect();
            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void handleListCommand(FTPClient ftpClient, PrintWriter writer) {
        try {
            FTPFile[] files = ftpClient.listFiles();
            for (FTPFile file : files) {
                writer.println(file.getName());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

FTP Client

```
import org.apache.commons.net.ftp.FTPClient;

import java.io.*;

public class FTPMultiClientExample {
```

```
public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
        Thread clientThread = new Thread(() -> {
            FTPClientExample ftpClient = new FTPClientExample();
            ftpClient.startClient();
        });
        clientThread.start();
    }
}

private void startClient() {
    String server = "localhost";
    int port = 21;
    String user = "anonymous";
    String pass = "anonymous";

    FTPClient ftpClient = new FTPClient();

    try {
        ftpClient.connect(server, port);
        ftpClient.login(user, pass);

        // Print the working directory
        String workingDir = ftpClient.printWorkingDirectory();
        System.out.println("Current working directory: " + workingDir);

        // Upload a file
        File fileToUpload = new File("localFile.txt");
        FileInputStream inputStream = new FileInputStream(fileToUpload);
        boolean uploaded = ftpClient.storeFile("remoteFile.txt", inputStream);
        inputStream.close();
        if (uploaded) {
            System.out.println("File uploaded successfully.");
            } else {
                System.out.println("Failed to upload the file.");
            }

            // Download a file
            String remoteFile = "remoteFile.txt";
            FileOutputStream outputStream = new FileOutputStream("downloadedFile.txt");
            boolean downloaded = ftpClient.retrieveFile(remoteFile, outputStream);
            outputStream.close();
            if (downloaded) {
                System.out.println("File downloaded successfully.");
            } else {
                System.out.println("Failed to download the file.");
            }

            // Logout and disconnect
            ftpClient.logout();
            ftpClient.disconnect();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```