Task 5: Java Networking and Serialization

Develop a basic TCP client and server application where the client sends a serialized object with 2 numbers and operation to be performed on them to the server, and the server computes the result and sends it back to the client. for eg, we could send 2, 2, ""+"" which would mean 2 + 2

ANS:

```
package com.Day25;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.Serializable;
// Main class containing server, client, and operation class
public class NetworkingSerialization {
  public static void main(String[] args) {
    // Start the server in a separate thread
    new Thread(() -> {
      try {
        Server.startServer();
      } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
      }
    }).start();
    // Give the server a moment to start
    try {
      Thread.sleep(1000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
    // Start the client
    Client.startClient();
  }
  // Operation class
  static class Operation implements Serializable {
    private static final long serialVersionUID = 1L;
    private double number1;
    private double number2;
    private String operation;
    public Operation(double number1, double number2, String operation) {
      this.number1 = number1;
      this.number2 = number2;
      this.operation = operation;
    }
    public double getNumber1() {
      return number1;
    }
    public double getNumber2() {
      return number2;
    }
    public String getOperation() {
```

```
        return operation;
    }
}
// Server class
static class Server {
    public static void startServer() throws IOException, ClassNotFoundException {
        try (ServerSocket serverSocket = new ServerSocket(12345)) {
            System.out.println("Server is listening on port 12345");
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New client connected");
                // Deserialize the operation object
                ObjectInputStream inputStream = new ObjectInputStream(socket.getInputStream());
                Operation operation = (Operation) inputStream.readObject();
                // Perform the operation
                double result = performOperation(operation);
                // Send the result back to the client
                ObjectOutputStream outputStream = new ObjectOutputStream(socket.getOutputStream());
                outputStream.writeObject(result);
                socket.close();
            }
        }
    }
    private static double performOperation(Operation operation) {
        double number1 = operation.getNumber1();
        double number2 = operation.getNumber2();
        String op = operation.getOperation();
        switch (op) {
            case "+":
                return number1 + number2;
            case "-":
                return number1 - number2;
            case "*":
                return number1 * number2;
            case "/":
                if (number2 != 0) {
                    return number1 / number2;
                } else {
                    throw new IllegalArgumentException("Cannot divide by zero");
                }
            default:
                throw new UnsupportedOperationException("Unsupported operation: " + op);
        }
    }
}
// Client class
static class Client {
    public static void startClient() {
        String hostname = "localhost";
        int port = 12345;
```

```
        Operation operation = new Operation(2, 2, "+");
        try (Socket socket = new Socket(hostname, port)) {
            // Serialize the operation object
            ObjectOutputStream outputStream = new ObjectOutputStream(socket.getOutputStream());
            outputStream.writeObject(operation);
            // Receive the result from the server
            ObjectInputStream inputStream = new ObjectInputStream(socket.getInputStream());
            double result = (double) inputStream.readObject();
            System.out.println("Result: " + result);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
  }
}
```