# Distributed Computing EXP-2

## Name:Atharva Pawar

**Roll NO:9427**

**Batch:D**

# Implement an RPC client and server

## Code:

//client.java

import java.io.\*; import java.net.\*;

public class client{

public static void main(String[] args) { if (args.length != 3) {

System.out.println("Usage: java Example <hostname> <num1> <num2>"); return;

}

String hostname = args[0];

int num1 = Integer.parseInt(args[1]); int num2 = Integer.parseInt(args[2]);

try {

// Create a socket connection to the server Socket socket = new Socket(hostname, 8888);

// Create ObjectOutputStream and ObjectInputStream for communication ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

// Send the numbers to the server out.writeInt(num1); out.writeInt(num2);

out.flush();

// Receive the result from the server int result = in.readInt();

System.out.println("Result of addition: " + result);

// Close the connection socket.close();

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

}

}

}

//server.java

import java.io.\*; import java.net.\*;

public class server {

public static void main(String[] args) { try {

// Create a ServerSocket

ServerSocket serverSocket = new ServerSocket(8888); System.out.println("Server started");

while (true) {

// Accept client connections

Socket socket = serverSocket.accept(); System.out.println("Client connected");

// Create ObjectInputStream and ObjectOutputStream for communication

ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

// Read numbers from client int num1 = in.readInt();

int num2 = in.readInt();

// Calculate the result

int result = num1 + num2;

// Send the result back to the client out.writeInt(result);

out.flush();

// Close the connection socket.close();

System.out.println("Client disconnected");

}

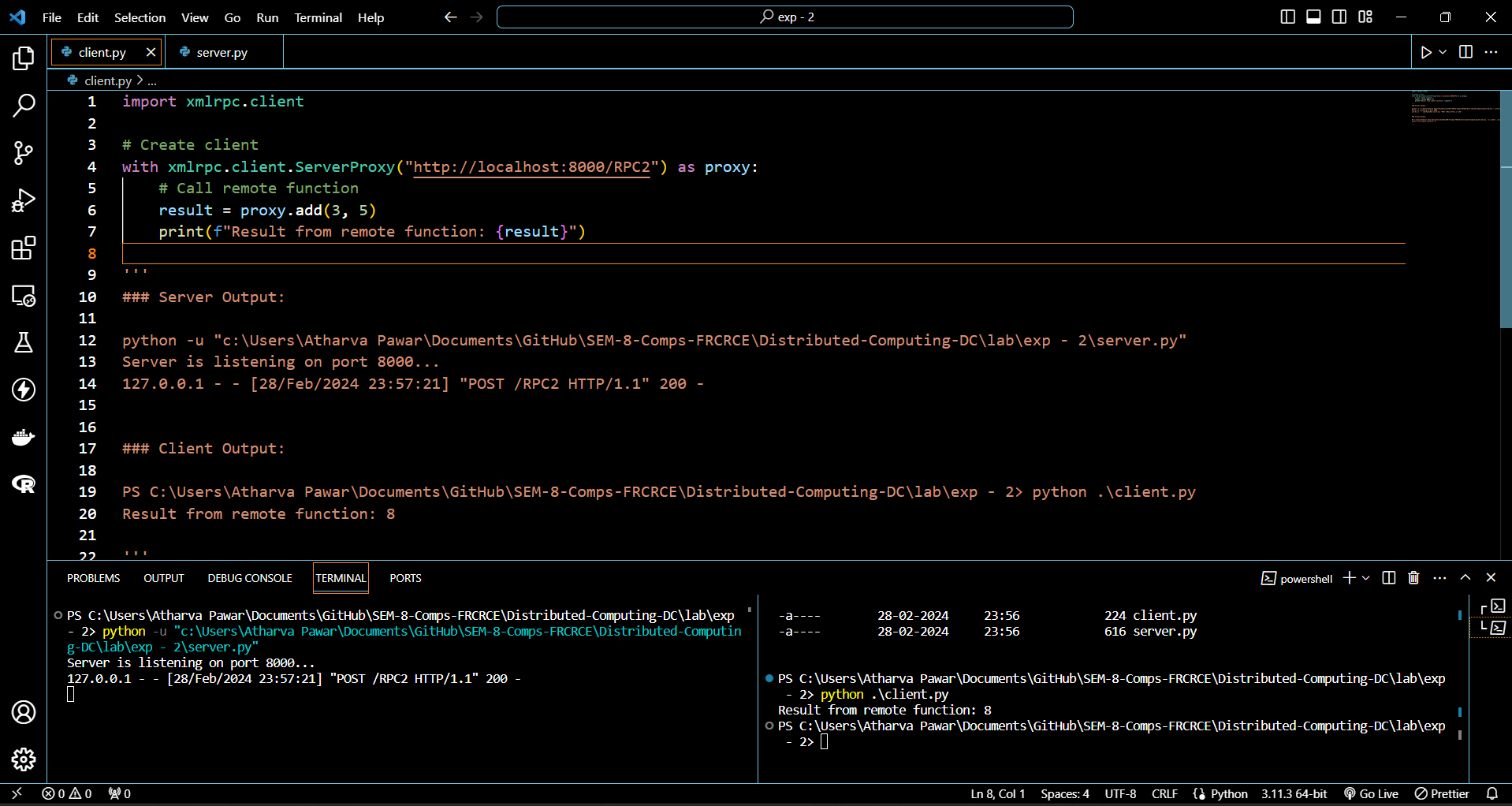
} catch (IOException e) { e.printStackTrace();

}

}

}

## Output:

****

POSTLAB

1. In which category of communication, RPC be included?

ANS:-

Remote Procedure Call (RPC) is a communication protocol used for communication between different processes or systems. It can be categorized under inter-process communication (IPC) because it allows one process to execute procedures or functions in another process, typically across a network. RPC enables distributed computing by allowing a program to cause a procedure (subroutine) to execute in another address space (commonly on another machine), without the programmer explicitly coding the details for this remote interaction.

1. What are stubs? What are the different ways of stub generation? Ans:-

Stubs are components used in software development and testing, particularly in the context of remote procedure calls (RPC) and distributed systems. They serve as placeholders or substitutes for actual modules or components during development and testing phases.

Manual Stub Creation: Developers manually create stub code, writing functions or methods that mimic the behavior of the real components. This approach requires a good understanding of the interfaces and expected behavior of the components being stubbed.

Automated Stub Generation from Interface Definitions: Many programming languages and RPC frameworks support automated stub generation based on interface definitions. Developers define the interfaces and methods that remote components will expose, and then tools generate the stub code automatically based on these interface definitions

Code Generation Tools: Some development environments provide code generation tools or plugins that facilitate stub generation. These tools may generate stub code based on high-level descriptions or configurations provided by developers, abstracting away the low-level details of stub creation.

1. What is binding?

Ans:- "binding" typically refers to the process of associating a specific network address (such as an IP address and port number) with a particular service or application.

Socket Binding: In network programming, when a server application listens for incoming connections on a specific port, it binds the server's socket to that port. This ensures that incoming network traffic addressed to that port is directed to the appropriate application for processing.

Service Binding: In distributed systems or microservices architecture, binding refers to associating a service instance with a specific network address or endpoint. Clients can then use this binding information to communicate with the service.

1. Name the transparencies achieved through stubs

Ans:- Stubs are used primarily in distributed computing and software testing to achieve various forms of transparency, which help to abstract and simplify.

Location Transparency: Stubs can hide the actual location of a remote service or component from the client. The client interacts with the stub as if it were the actual service, and the stub handles the details of communication with the remote component, which may be located on a different machine or network.

Access Transparency: Stubs can provide access to remote components in a uniform manner,

regardless of the underlying communication mechanisms or protocols used. This shields the client from the complexities of network communication and allows for easier integration of remote services.

Failure Transparency: Stubs can handle failures and errors transparently, shielding the client from the details of network failures, service unavailability, or other issues. Depending on the implementation, stubs may provide error handling and retry mechanisms to improve the robustness of the distributed system.