
# COLLEGE OF COMPUTING AND INFORMATICS

## DEPARTMENT OF SOFTWARE ENGINEERING

## FUNDAMENTALS OF DISTRIBUTED SYSTEMS(SEGN3032)

## GROUP ASSIGNMENT ON CLIENT-SERVER INTERACTION

**Group Members** **ID**

1. Yishak Alemu…….3661/14
2. Yohannes Ayenew…….3676/14
3. Yosef Dagne ……3720/14
4. Yusuf Kedir …….3737/14
5. Zekarias Tamiru …….3747/14
6. Hawo Mohammed ……. S/5067/14

Submission date: November 10, 2024

Submitted to: Mr. Hussen A. (MSc)

Haramaya, Ethiopia

## ➢ Client-Server Connection in Java

A client-server connection in Java is a common architecture that enables communication between two programs, a client and a server. This model is widely used in networked applications, where the client requests resources or services, and the server provides them.

### Key Concepts

### 1. Server

- The server is a program that listens for incoming client connections on a specific port.

- It handles requests from clients, processes them, and sends back responses.

- Servers can be multi-threaded to handle multiple clients simultaneously**.**

### 2. Client

- The client initiates a connection to the server by specifying the server's hostname and port number.

- It sends requests to the server and waits for responses.

- Clients can be applications, web browsers, or any software that communicates over a network.

### 3. Sockets

- Java provides the `java.net` package, which includes `Socket` for client-side communication and `ServerSocket` for server-side listening.

- A Socket represents a connection between the client and the server, while a `ServerSocket` listens for incoming connections.

**4. Communication**

- Communication typically involves sending and receiving streams of data. This can be done using input and output streams (`InputStream`, `OutputStream`).

- Data can be transmitted in various formats (text, binary) depending on the application.

**5. Protocol**

- The client and server must agree on a protocol for communication. This includes how messages are formatted, what commands are valid, and how responses are structured.

- **Steps of the workflow**

**1. Server Setup**

The server creates a `ServerSocket` and listens for connections on a specified port.

**2. Client Connection**

- The client creates a `Socket` to connect to the server.

**3. Data Exchange**

- The client sends a request to the server.

- The server processes the request, generates a response, and sends it back to the client.

**4. Closing Connections**

- After the communication is complete, both the client and server close their respective sockets to free up resources.

**Note: the implementation is attached on the zip file**