CPCS-371 Computer Networks (I)

Client-Server Group Project

Made By:

Group Alpha

Abdullah Abed Alharbi – 2135999

Abdullah Emad Almashharawi - 2136141

Mahmued Alardawi – 2135209

Table of Contents

1. Introduction

Our group project focuses on developing a client-server application using Java. A client-server architecture enables communication between multiple devices over a network, allowing clients to request services or resources from a server. This project aims to demonstrate the implementation of a client-server model and showcase its practical applications.

1.1 Overview

In our project, we have developed two components: the client and the server. The client is responsible for sending requests to the server, while the server handles these requests and provides the requested services or resources. The communication between the client and server occurs via sockets, which establish a network connection.

1.2 Key Features

Here are the key features of our client-server application:

1. **Client Functionality:** The client component allows users to interact with the server by sending requests and receiving responses. We have implemented user-friendly prompts and input handling to ensure smooth communication with the server.

2. **Server Functionality:** The server component receives client requests, processes them, and sends back the appropriate responses. It handles multiple client connections concurrently, ensuring efficient and scalable performance.

3. **Exception Handling:** Our project includes robust exception handling to address potential errors or exceptions that may occur during the execution of the client-server application. This ensures the application's stability and provides informative error messages to users.

4. **Use of Java Networking:** We use Java's networking capabilities to establish socket connections between the client and server. This allows for reliable and secure communication over a network, enabling the exchange of data and services.

2. Implementation

To implement the client-server idea into code, we can use Java as the programming language. Java provides built-in support for networking, making it a suitable choice for developing client-server applications.

In Java, we can utilize the standard library, which includes classes like Socket, ServerSocket, PrintWriter, and BufferedReader, to handle the networking aspects. These classes allow us to establish connections, send and receive data between the client and server.

To establish communication between the client and server, we create a Socket object on the client-side, specifying the server's IP address and port number. On the server-side, we create a ServerSocket object that listens for incoming client connections. When a client connects, the server accepts the connection and creates a Socket object to communicate with the client.

For data exchange, we use input and output streams. The PrintWriter class allows us to send data to the other side. And the BufferedReader class reads data from the input stream, enabling us to receive data from the other side.

In addition to input and output streams for data exchange between the client and server, we can also incorporate the Scanner class to read input from the console. On the client-side, we create a Scanner object to read user input from the console. This allows the client to accept input from the user, which can then be sent to the server for processing or as part of a request.

3. Testing

3.1 Testing on a regular run

- When we run the programs, first they will check for connection:

```
Checking For Connection...
```

*Note: The server must be launched in order for the client to connect. Otherwise, an error will occur. More on that in section 3.2.

- When connection is successful, the following messages are displayed:

Server-side:

```
Checking For Connection...
Client Connected Successfully!
```

Client-Side:

```
Checking For Connection...
Connected to Server Successfully!
Enter a Character to be searched:
```

*In addition to the connection update status, the server will send the first prompt asking the client to enter their character to be searched.

- After client-side enters the character of their choice, it gets sent to the server-side:
  Client-Side:

```
Checking For Connection...
Connected to Server Successfully!
Enter a Character to be searched: a
```

  Server-Side:

```
Checking For Connection...
Client Connected Successfully!
Client chose 'A' to be searched.
```

- Afterwards, the server prompts the client for the string to be searched:
  Client-Side:

```
Checking For Connection...
Connected to Server Successfully!
Enter a Character to be searched: a
Enter a String: La Ilaha Ila Allah :]
```

- When the client gives the string to the server. The server notifies us in the console. And sends the number of occurrences of the chosen character in the given string to the client. Then prompts the client whether they want to repeat the process or not. Here's what's being displayed on both sides:
  Server-Side:

```
Checking For Connection...
Client Connected Successfully!
Client chose 'A' to be searched.
Client's String is "La Ilaha Ila Allah :]".
```

  Client-Side:

```
Checking For Connection...
Connected to Server Successfully!
Enter a Character to be searched: a
Enter a String: La Ilaha Ila Allah :]
The number of Occurrences are: 6
Want to repeat (Y/N):
```

- Now the client has the option to repeat it by inputting Y, or to quit the program by inputting N to the server. Here's an example of the client quitting:
  Client-Side:

```
Checking For Connection...
Connected to Server Successfully!
Enter a Character to be searched: a
Enter a String: La Ilaha Ila Allah :]
The number of Occurrences are: 6
Want to repeat (Y/N): n
Thank You!

Process finished with exit code 0
```

  Server-Side:

```
Checking For Connection...
Client Connected Successfully!
Client chose 'A' to be searched.
Client's String is "La Ilaha Ila Allah :]".
Client chose to stop.

Process finished with exit code 0
```

3.2 Possible Errors

- As mentioned previously, when the Client program is started while the server is not running, the following error is displayed on the client console:

```
Checking For Connection...
Could not find server. Make sure the server program is running before turning the client program on.
Shutting Down...

Process finished with exit code 1
```

- When both sides connect successfully, but one of the sides loses connection or terminates, the following error will be displayed on the current running side:

```
Checking For Connection...
Connected to Server Successfully!
Enter a Character to be searched: a
Lost connection to server.
Shutting Down...

Process finished with exit code 2
```

4. Conclusion

Our project focuses on creating a client-server application to showcase the benefits of distributed computing. It highlights how different devices can work together to provide services, exchange data, and communicate effectively.

Our goal is to provide a practical example of a client-server architecture and inspire further exploration and development in this area.