Documentation of the Communication Protocol between Client and Server

1. Introduction

This documentation describes the communication protocol between the client and server using sockets and RMI (Remote Method Invocation). The project includes a series of Java classes that handle different parts of the communication and data flow between the client and the server.

2. Architecture

The system architecture is primarily divided into three components:

- Client: Responsible for interacting with the user and sending requests to the server.
- Server: Handles requests from clients and sends appropriate responses.
- Messages: Define the structure of data exchanged between the client and server.

3. Key Components

3.1 Client

- Client.java: The main client class that initiates the connection and manages communication.
- ClientController.java: Controls client operations, including game logic and handling server responses.

3.2 Server

- Server.java: The main server class that accepts connections from clients and coordinates server operations.
- ServerController.java: Manages the server game logic and communications with clients.

3.3 Messages

The message classes define the format of data exchanged between client and server. Some examples include:

- Message.java: Base class for all messages.
- LoginRequestMessage.java: Message for login requests.
- LoginResponseMessage.java: Message for login request responses.
- ErrorMessage.java: Message for communicating errors.
- GameRoomRequest.java: Message for game room requests.

3.4 RMI

- RmiServer.java: RMI server interface.
- RmiClient.java: RMI client interface.
- RmiServerImpl.java: RMI server implementation.
- RmiClientImpl.java: RMI client implementation.
- RmiServerController.java: Controls RMI server operations.

3.5 Socket

- SktClient.java: Socket client implementation.
- SktServer.java: Socket server implementation.
- SktServerController.java: Controls socket server operations.

4. Communication Flow

1. Connection:
   - The client initiates a connection to the server via socket or RMI.
   - The server accepts the connection and creates a thread to handle communication with the client.

2. Message Exchange:

   - The client sends a request (e.g., LoginRequestMessage) to the server.

   - The server processes the request and sends a response (e.g., LoginResponseMessage).

3. Request Handling:

   - Client requests are managed by the ServerController, which invokes appropriate operations and responds to the client.

4. Connection Closure:

   - At the end of communication, either the client or server can close the connection.

5. Conclusion

This documentation provides an overview of the communication protocol between the client and server using sockets and RMI. The described classes handle various aspects of communication, from initial connection to message exchange and request handling.