Conversation Design

Course Name: Basic Computer Science Practice

Professor: Li Yugang

Class Day/ Time: Monday - Friday / 08.00 - 11.30, 14.10-16.40

Leader: 1820222041 – 温富勝 Alexander Darryl Kristiawan

Member: 1820222021 – 郑国強 Darren Tejaatmaja

1820222030 – 林哲豪 Wilbert Jaya Sucipto

1820222040 – 冯明想 Jesslyn Clarissa Hermanto

Project Topic: Instant Messaging

To design a robust instant messaging (IM) system with the specified properties, we need to focus on creating a well-structured conversation mechanism that can handle real-time, text-based communication among multiple users. Here's a precise definition of how conversations will be managed in this IM system, including the necessary Java classes and their interactions.

Concept of a Conversation

A conversation in the IM system is an interactive session where multiple users exchange messages in real-time. Each conversation has a unique identifier and maintains a list of participants and messages. The conversation allows users to join, leave, and view the message history while ensuring that messages are delivered instantly to all participants.

Java Classes for Conversation Management

1. Conversation Class: Manages the state of a conversation, including participants and message history.

   Public Methods:

- addParticipant(User user): Adds a user to the conversation.
- removeParticipant(User user): Removes a user from the conversation.
- sendMessage(User sender, String message): Sends a message from a participant.
- getMessages(): Returns a list of messages in the conversation.
- getParticipants(): Returns a list of participants in the conversation.

2. User Class : Represents a user who participates in conversations.
   Public Methods:
   - sendMessage(Conversation conversation, String message): Sends a message to a specified conversation.
   - receiveMessage(Message message): Receives a message from a conversation.
   - getName(): Returns the user's name.
3. Message Class:
   a. **Purpose:** Represents a message in a conversation.
   b. Public Methods:
      - "getSender()": Returns the sender of the message.
      - "getContent()": Returns the content of the message.

3. Client-Server Interaction

a. Client Actions:

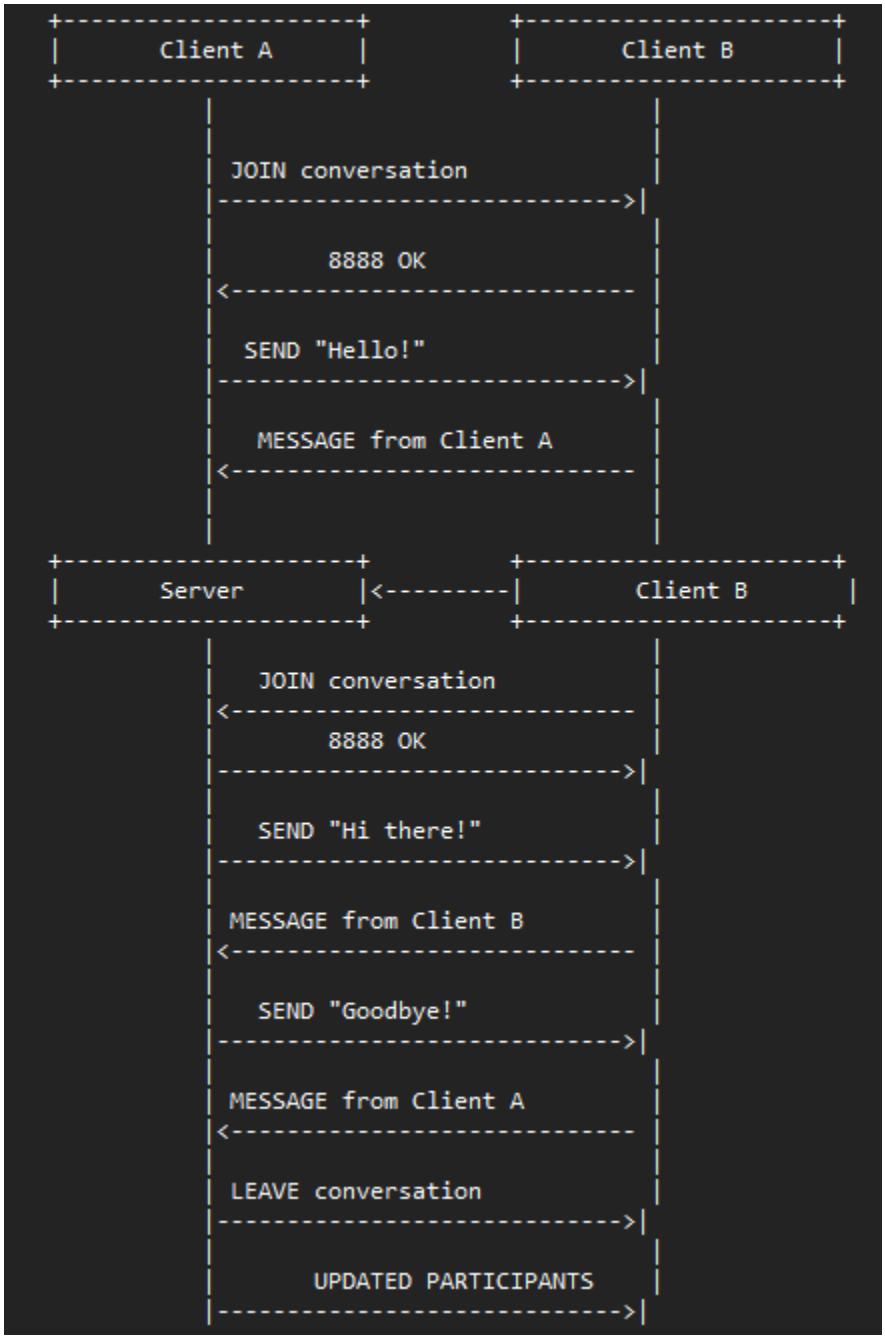
- **Join Conversation:** Client sends a request to join a conversation. The server adds the user to the conversation and sends back the current message history.
- **Send Message:** Client sends a message to the conversation. The server broadcasts this message to all participants.
- **Leave Conversation:** Client sends a request to leave a conversation. The server removes the user and updates the remaining participants.

b. Server Actions:

- **Manage Participants:** The server keeps track of all conversations and participants.
- **Broadcast Messages:** The server ensures that messages are distributed to all participants in a conversation.
- **Handle Connections:** The server manages client connections, including adding and removing users from conversations.

4. Snapshot Diagram of a Conversation in Action

Here’s a visual representation of how a conversation flows:



Summary

    - **“Conversation” Class:** Manages conversation state, participants, and messages.
    - **“User” Class:** Represents users and handles message sending/receiving.
- **“Message” Class:** Encapsulates message details.
- **Client-Server Interaction:** Includes joining, sending, and leaving conversations with proper message handling.

This design ensures that the IM system supports real-time communication, allows multiple concurrent conversations, and provides a clear mechanism for managing users and messages.