Conversation Design

Overview

In this IM system, a **Conversation** represents a series of messages exchanged between two or more users. The Java classes will encapsulate the logic for managing conversations, handling messages, and managing the lifecycle of connections via WebSockets.

Java Classes

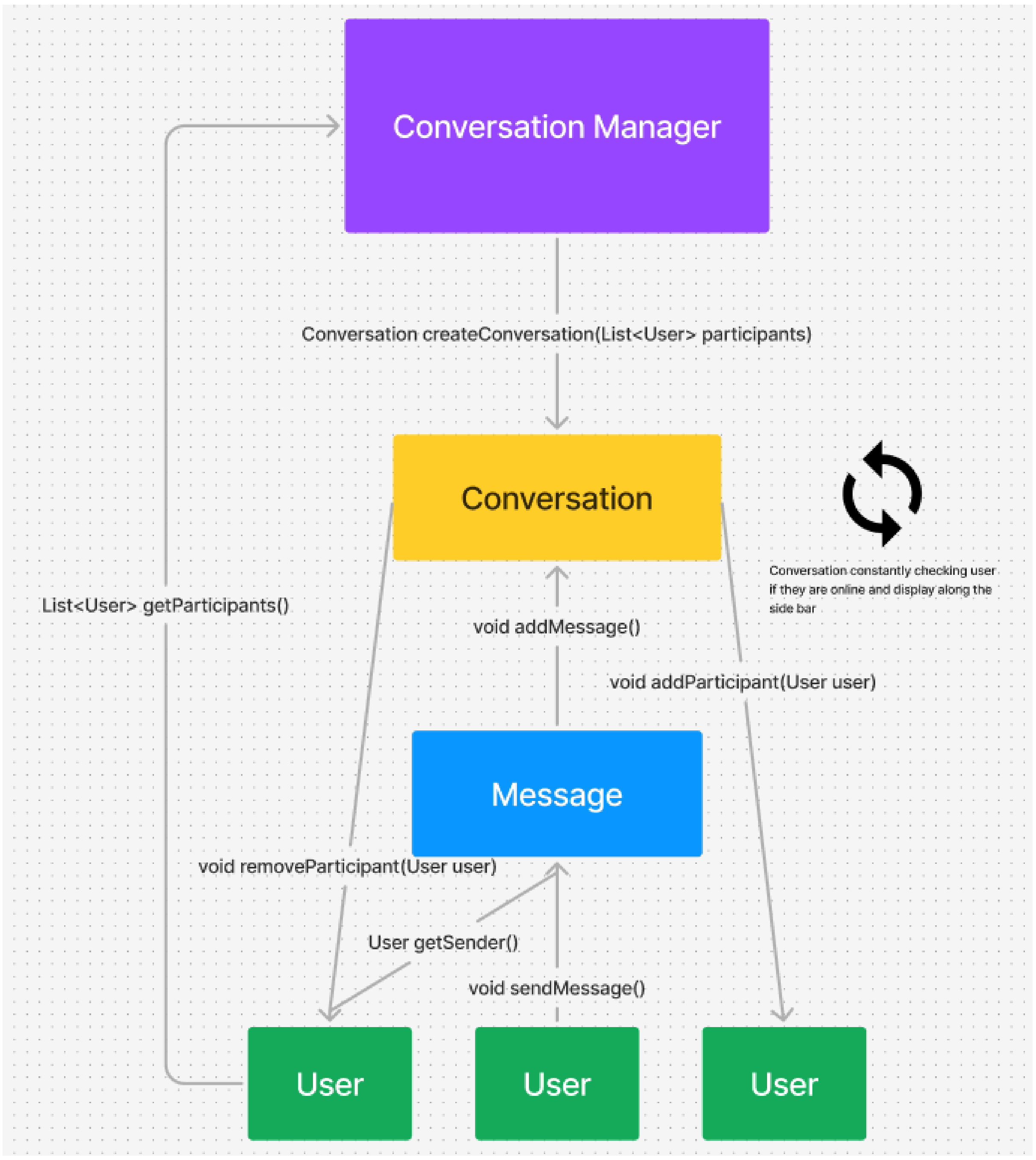
1. **Conversation**
   - **Description**: Represents a conversation between two or more users. It contains a list of participants and messages exchanged within the conversation.
   - **Public Methods**:
     - `void addParticipant(User user)`: Adds a user to the conversation.
     - `void removeParticipant(User user)`: Removes a user from the conversation.
     - `void addMessage(Message message)`: Adds a message to the conversation.
     - `List<Message> getMessages()`: Retrieves the list of all messages in the conversation.
     - `List<User> getParticipants()`: Retrieves the list of all participants.
     - `int getConversationId()`: Returns the unique identifier of the conversation.
   - **Interactions**: This class interacts with `User` and `Message` classes to manage the participants and messages within a conversation.
2. **Message**
   - **Description**: Represents a message within a conversation. Contains details like sender, timestamp, and content.
   - **Public Methods**:
     - `String getContent()`: Retrieves the content of the message.
     - `User getSender()`: Retrieves the sender of the message.
     - `LocalDateTime getTimestamp()`: Retrieves the timestamp of the message.
     - `int getMessageId()`: Returns the unique identifier of the message.
   - **Interactions**: This class interacts with the `User` class to associate a message with its sender and the `Conversation` class to be added to a conversation.
3. **User**
   - **Description**: Represents a user in the system. Contains user details like username, status, and connected WebSocket session.
   - **Public Methods**:

        - `String getUsername()`: Retrieves the username of the user.
        - `boolean isOnline()`: Checks if the user is currently online.
        - `WebSocketSession getSession()`: Retrieves the current WebSocket session of the user.
        - `void sendMessage(Message message)`: Sends a message to the user.
    - **Interactions**: This class interacts with the `Message` class when sending/receiving messages and with the `Conversation` class to participate in conversations.
4. **ConversationManager**
    - **Description**: Manages multiple conversations, creating new ones and managing existing ones.
    - **Public Methods**:
        - `Conversation createConversation(List<User> participants)`: Creates a new conversation with the specified participants.
        - `Conversation getConversation(int conversationId)`: Retrieves an existing conversation by its ID.
        - `void endConversation(int conversationId)`: Ends a conversation and removes it from the active list.
    - **Interactions**: This class manages instances of the `Conversation` class, creating and retrieving them as needed. It interacts with the `User` class to add participants to conversations.

Class Interactions

- **Message Sending**: When a user sends a message, the `User` class uses the `WebSocketSession` to send the message. The message is then added to the corresponding `Conversation` through the `addMessage()` method. The `Conversation` manages the list of messages and participants, ensuring that the message is broadcast to all participants.
- **Conversation Management**: The `ConversationManager` class handles the creation and retrieval of conversations. When a new conversation is initiated, it creates an instance of the `Conversation` class and adds the relevant users as participants. It is also responsible for ending conversations and ensuring they are removed from the active list.

Snapshot Diagram (Rough Draft):


Conversation Manager
Conversation createConversation(List<User> participants)
Conversation
Conversation constantly checking user if they are online and display along the side bar
List<User> getParticipants()
void addMessage()
void addParticipant(User user)
Message
void removeParticipant(User user)
User getSender()
void sendMessage()
User
User
User