API - Serializable Message Class

Introduction

The serializable `Message` class is used to send information between the client and the server. A `MessageType` enumerated type attribute is used to determine the type of message being sent or received. The constructor for the `Message` has a single mandatory parameter of type `MessageType`, which is defined as follows:

```
enum MessageType {
  Login,
  Logout,
  NewChat,
  NewRoom,
  CreateRoom,
  LeaveRoom,
  AddToRoom,
  ChangeStatus,
  UpdateUserStatus,
  GetLog,
}
```

Messages of type `Login`, `Logout`, `ChangeStatus`, and `UpdateUserStatus` reflect changes in the status of a user. The possible statuses are defined in the `UserStatus` enumerated type, which is defined as follows:

```
enum UserStatus {
  Online,
  Offline,
  Away,
  Busy,
}
```

Usage

Login

To establish communcation with the server, a `Message` with type `Login` must be sent. The required attributes to be set are:

- The id of the user to login
- The password of the user to login

In direct response to the `Message` sent, the server will send a `Message` with type `Login`. **If the login was successful,** the following attributes will be set:

- The `User` object of the user who logged in
- A list of `User` objects of all users
- A list of `Room` objects of all rooms

```
// Creating and setting Message attributes
Message msg = new Message(MessageType.Login);
msg.setUserId("userId");
msg.setPassword("password");

// Handling the Message response
Message res = (Message) ObjectInputStream.readObject();
User user = res.getUser();
List<User> user_list = res.getUserList();
List<Room> rooms = res.getRooms();
```

The client should expect to receive messages of type `UpdateUserStatus` that are sent by the server when another user logs in. The following attributes will be set:

- The id of the user who logged in
- The status of the user

```
// Handling the received Message
Message res = (Message) ObjectInputStream.readObject();

if (res.getType() == MessageType.UpdateUserStatus) {
  String userId = res.getUserId();
  UserStatus status = res.getUserStatus();
}
```

Logout

To logout from the server, a `Message` with type `Logout` must be sent. The required attributes to be set are:

- The id of the user to logout
- The status of the user to logout (should be `UserStatus.Offline`)

```
// Creating and setting Message attributes
Message msg = new Message(MessageType.Logout);
msg.setUserId("userId");
msg.setUserStatus(UserStatus.Offline);
```

No response is expected for the `Message` sender.

The client should expect to receive messages of type `UpdateUserStatus` that are sent by the server when another user logs out. Handling of this message is the same as described in the `Login` section.

NewChat

To send a chat message to another a room, a `Message` with type `NewChat` must be sent. The required attributes to be set are:

- The id of the user sending the chat message
- The id of the room to send the chat message to
- The contents of the chat message

```
// Creating and setting Message attributes
Message msg = new Message(MessageType.NewChat);
msg.setUserId("senderUserId");
msg.setRoomId("roomId")
msg.setContents("message");
```

No response is expected for the `Message` sender.

The client should expect to receive messages of type `NewChat` that are sent by the server when another user sends a chat message to the room that the client is in. The following attributes will be set:

- The id of the user who sent the chat message
- The id of the room the chat message was sent in
- The timestamp of the chat message
- The contents of the chat message

```
// Handling the received Message
Message res = (Message) ObjectInputStream.readObject();

if (res.getType() == MessageType.NewChat) {
  String sender = res.getUserId();
  String roomId = res.getRoomId();
  String timestamp = res.getTimestamp();
  String contents = res.getContents();
}
```

CreateRoom

To create a new room, a `Message` with type `CreateRoom` must be sent. The required attributes to be set are:

- The id of the user creating the room
- The name of the room to create
- The list of users to add to the room

```
// Message creation and attribute setting
Message msg = new Message(MessageType.CreateRoom);
msg.setUserId("creatorUserId");
msg.setContents("roomName");
// This setter expects a List<String> of user ids
msg.setUsers(users);
```

No response is expected for the `Message` sender.

The client should expect to receive messages of type `NewRoom` that are sent by the server when another user creates a new room that the client will be a part of. The following attributes will be set:

- The id of the user who created the room
- A list of rooms which contains the newly created room

```
// Handling the received Message
Message res = (Message) ObjectInputStream.readObject();

if (res.getType() == MessageType.NewRoom) {
  String creator = res.getUserId();
  // Expect only one room in the list
  List<Room> rooms = res.getRooms();
}
```

LeaveRoom

To leave a room, a `Message` with type `LeaveRoom` must be sent. The required attributes to be set are:

- The id of the user leaving the room
- The id of the room to leave

```
// Creating and setting Message attributes
Message msg = new Message(MessageType.LeaveRoom);
msg.setUserId("userId");
msg.setRoomId("roomId");
```

No response is expected for the `Message` sender.

The client should expect to receive messages of type `LeaveRoom` that are sent by the server when another user leaves a room that the client is in. The following attributes will be set:

- The id of the user who left the room
- The id of the room that the user left
- The timestamp of when the user left the room

```
// Handling the received Message
Message res = (Message) ObjectInputStream.readObject();

if (res.getType() == MessageType.LeaveRoom) {
  String userId = res.getUserId();
  String roomId = res.getRoomId();
  String timestamp = res.getTimestamp();
}
```

AddToRoom

To add a user to a room, a `Message` with type `AddToRoom` must be sent. The required attributes to be set are:

- The id of the user adding another user to the room
- The id of the user to be added
- The id of the room to add the user to

```
// Creating and setting Message attributes
Message msg = new Message(MessageType.AddToRoom);
msg.setUserId("userId");
msg.setContents("userIdToAdd");
msg.setRoomId("roomId");
```

No response is expected for the `Message` sender.

There are two types of messages that the client should expect to receive from the server when a user is added to a room.

The user that was added to the room will receive a message of type `NewRoom`; handling of this message is the same as described in the `CreateRoom` section.

All other users in the room will receive a message of type `AddToRoom`. The following attributes will be set:

- The id of the user who added to the room
- The id of the room that the user was added to
- The timestamp of when the user was added to the room

```
// Handling the received Message
Message res = (Message) ObjectInputStream.readObject();

if (res.getType() == MessageType.AddToRoom) {
  // the id of the user to be added is stored in the contents
  String addedUserId = res.getContents();
  String roomId = res.getRoomId();
  String timestamp = res.getTimestamp();
}
```

ChangeStatus

To change the status of a user, a `Message` with type `ChangeStatus` must be sent. The required attributes to be set are:

- The id of the user changing their status
- The new status of the user

```
// Creating and setting Message attributes
Message msg = new Message(MessageType.ChangeStatus);
msg.setUserId("userId");
msg.setUserStatus(UserStatus.Away);
```

No response is expected for the `Message` sender.

The client should expect to receive messages of type `UpdateUserStatus` that are sent by the server when another user changes their status. Handling of this message is the same as described in the `Login` section.