Communications App

By: Ayoub Mekkaoui, Ben Levy, Akbar Hashimi, Van Nguyen, Ibraheem Fawal, Nico Pallma

Design

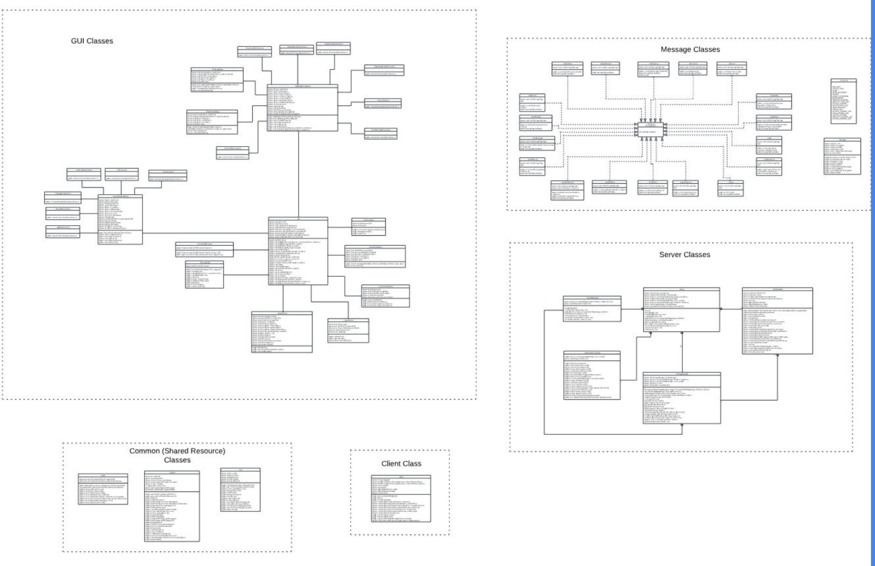
Client-Server

Client Responsibilities

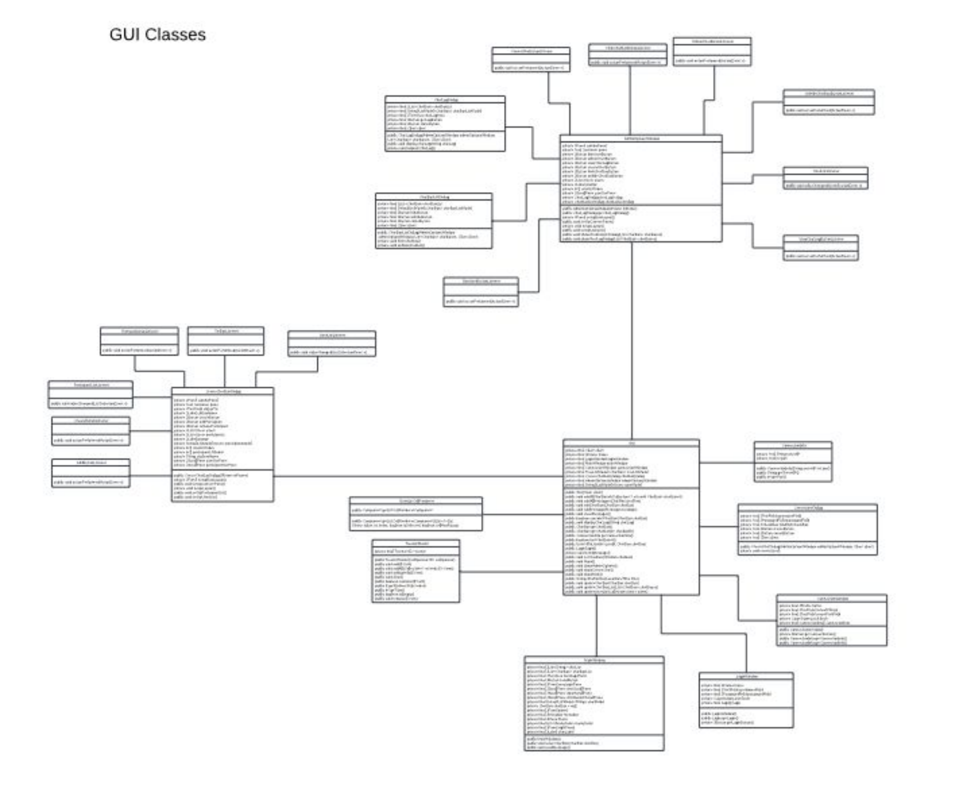
- GUI component
- Sending/receiving messages
- Local state updates

Server Responsibilities

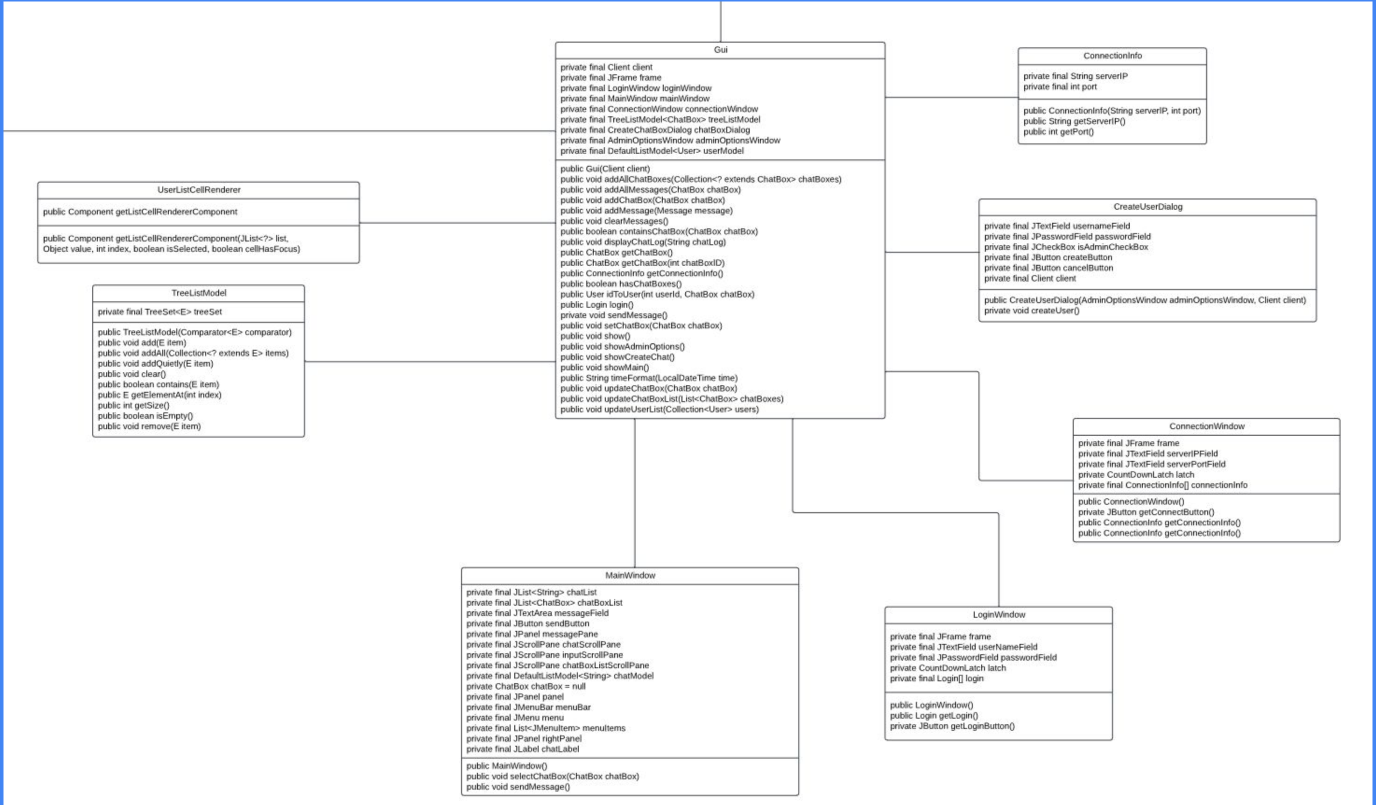
- User Authentication
- Handle multiple clients
- Data storage



GUI (Class Diagram)



GUI (Class Diagram) - cont



```
Gui
private final Client client
private final JFrame frame
private final LoginWindow loginWindow
private final MainWindow mainWindow
private final ConnectionWindow connectionWindow
private final TreeListModel<ChatBox> treeListModel
private final CreateChatBoxDialog chatBoxDialog
private final AdminOptionsWindow adminOptionsWindow
private final DefaultListModel<User> userModel

public Gui(Client client)
public void addAllChatBoxes(Collection<? extends ChatBox> chatBoxes)
public void addAllMessages(ChatBox chatBox)
public void addChatBox(ChatBox chatBox)
public void addMessage(Message message)
public void clearMessages()
public boolean containsChatBox(ChatBox chatBox)
public void displayChatLog(String chatLog)
public ChatBox getChatBox()
public ChatBox getChatBox(int chatBoxID)
public ConnectionInfo getConnectionInfo()
public boolean hasChatBoxes()
public User idToUser(int userId, ChatBox chatBox)
public Login login()
private void sendMessage()
public void setChatBox(ChatBox chatBox)
public void show()
public void showAdminOptions()
public void showCreateChat()
public void showMain()
public String timeFormat(LocalDateTime time)
public void updateChatBox(ChatBox chatBox)
public void updateChatBoxList(List<ChatBox> chatBoxes)
public void updateUserList(Collection<User> users)
```

```
ConnectionInfo
private final String serverIP
private final int port

public ConnectionInfo(String serverIP, int port)
public String getServerIP()
public int getPort()
```

```
UserListCellRenderer
public Component getListCellRendererComponent

public Component getListCellRendererComponent(JList<?> list,
Object value, int index, boolean isSelected, boolean cellHasFocus)
```

```
CreateUserDialog
private final JTextField usernameField
private final JPasswordField passwordField
private final JCheckBox isAdminCheckBox
private final JButton createButton
private final JButton cancelButton
private final Client client

public CreateUserDialog(AdminOptionsWindow adminOptionsWindow, Client client)
private void createUser()
```

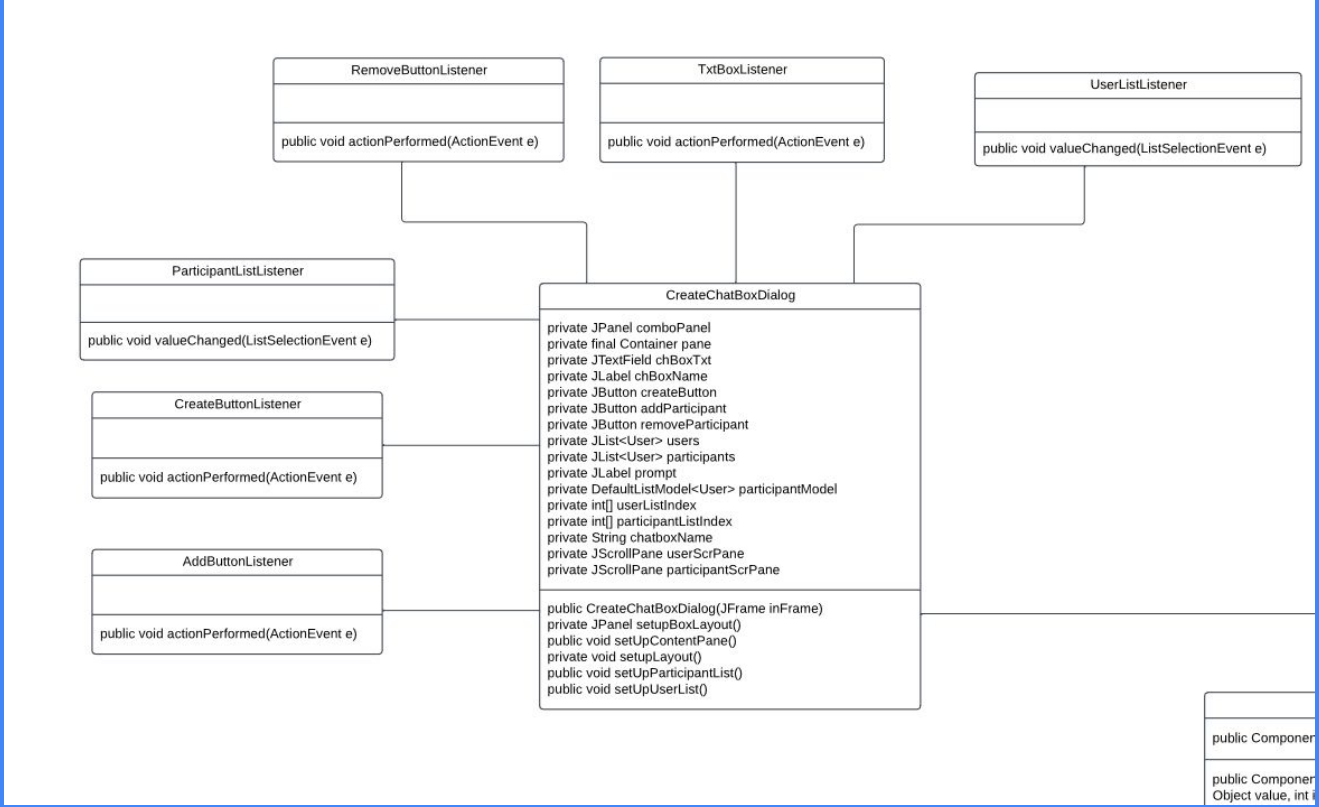
```
TreeListModel
private final TreeSet<E> treeSet

public TreeListModel(Comparator<E> comparator)
public void add(E item)
public void addAll(Collection<? extends E> items)
public void addQuietly(E item)
public void clear()
public boolean contains(E item)
public E getElementAt(int index)
public int getSize()
public boolean isEmpty()
public void remove(E item)
```

```
ConnectionWindow
private final JFrame frame
private final JTextField serverIPField
private final JTextField serverPortField
private CountDownLatch latch
private final ConnectionInfo[] connectionInfo

public ConnectionWindow()
private JButton getConnectButton()
public ConnectionInfo getConnectionInfo()
public ConnectionInfo getConnectionInfo()
```

```
MainWindow
private final JList<String> chatList
private final JList<ChatBox> chatBoxList
private final JTextArea messageField
private final JButton sendButton
private final JPanel messagePane
private final JScrollPane chatScrollPane
private final JScrollPane inputScrollPane
private final JScrollPane chatBoxListScrollPane
private final DefaultListModel<String> chatModel
private ChatBox chatBox = null
private final JPanel panel
private final JMenuBar menuBar
private final JMenu menu
private final List<JMenuItem> menuItems
private final JPanel rightPanel
private final JLabel chatLabel

public MainWindow()
public void selectChatBox(ChatBox chatBox)
public void sendMessage()
```

```
LoginWindow
private final JFrame frame
private final JTextField userNameField
private final JPasswordField passwordField
private CountDownLatch latch
private final Login[] login

public LoginWindow()
public Login getLogin()
private JButton getLoginButton()
```

GUI (Class Diagram) - cont



RemoveButtonListener

public void actionPerformed(ActionEvent e)

TxtBoxListener

public void actionPerformed(ActionEvent e)

UserListListener

public void valueChanged(ListSelectionEvent e)

ParticipantListListener

public void valueChanged(ListSelectionEvent e)

CreateButtonListener

public void actionPerformed(ActionEvent e)

AddButtonListener

public void actionPerformed(ActionEvent e)

CreateChatBoxDialog

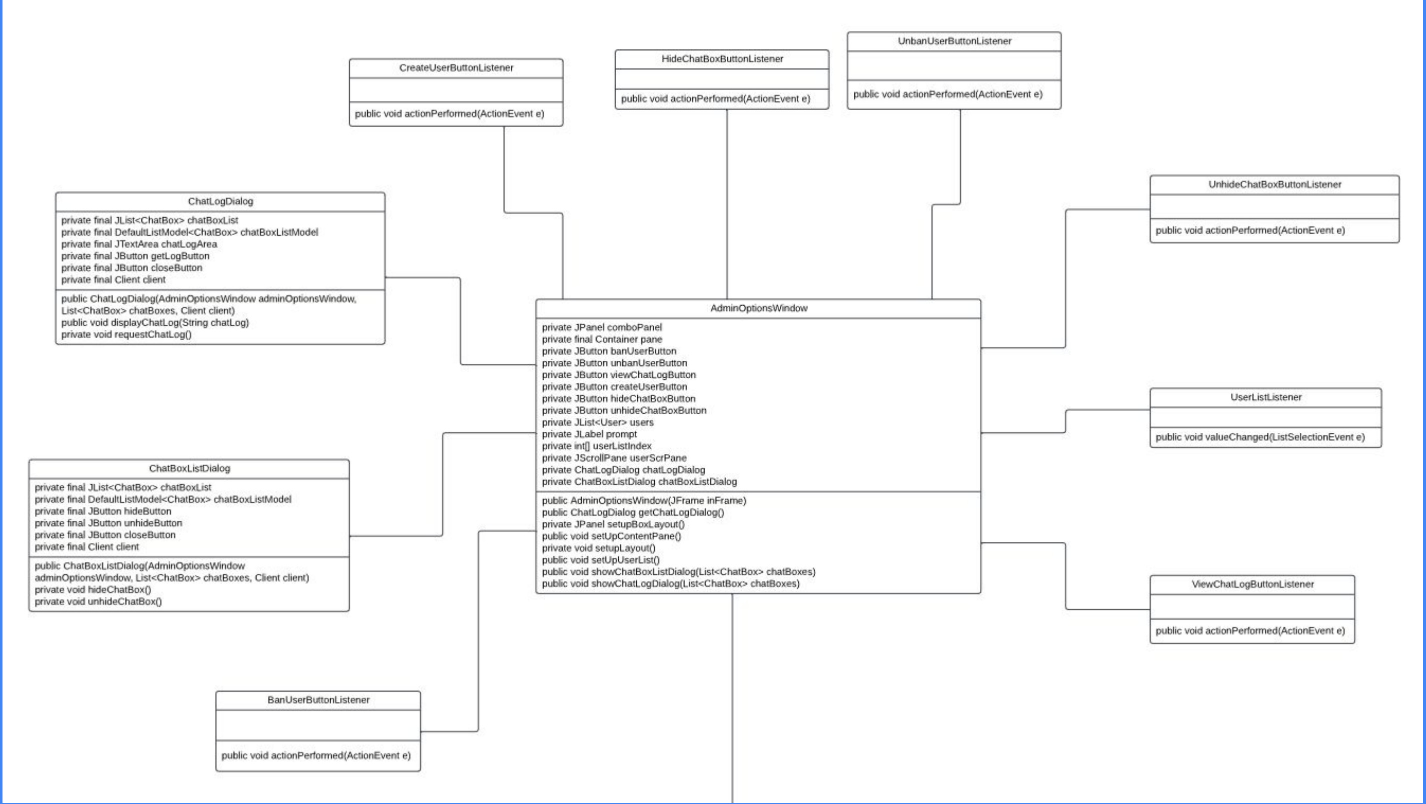
private JPanel comboPanel
private final Container pane
private JTextField chBoxTxt
private JLabel chBoxName
private JButton createButton
private JButton addParticipant
private JButton removeParticipant
private JList<User> users
private JList<User> participants
private JLabel prompt
private DefaultListModel<User> participantModel
private int[] userListIndex
private int[] participantListIndex
private String chatboxName
private JScrollPane userScrPane
private JScrollPane participantScrPane

public CreateChatBoxDialog(JFrame inFrame)
private JPanel setupBoxLayout()
public void setUpContentPane()
private void setupLayout()
public void setUpParticipantList()
public void setUpUserList()

public Componer

public Componer
Object value, int i

GUI (Class Diagram) - cont



CreateUserButtonListener

public void actionPerformed(ActionEvent e)

HideChatBoxButtonListener

public void actionPerformed(ActionEvent e)

UnbanUserButtonListener

public void actionPerformed(ActionEvent e)

ChatLogDialog

private final JList<ChatBox> chatBoxList
private final DefaultListModel<ChatBox> chatBoxListModel
private final JTextArea chatLogArea
private final JButton getLogButton
private final JButton closeButton
private final Client client

public ChatLogDialog(AdminOptionsWindow adminOptionsWindow, List<ChatBox> chatBoxes, Client client)
public void displayChatLog(String chatLog)
private void requestChatLog()

UnhideChatBoxButtonListener

public void actionPerformed(ActionEvent e)

AdminOptionsWindow

private JPanel comboPanel
private final Container pane
private JButton banUserButton
private JButton unbanUserButton
private JButton viewChatLogButton
private JButton createUserButton
private JButton hideChatBoxButton
private JButton unhideChatBoxButton
private JList<User> users
private JLabel prompt
private int[] userListIndex
private JScrollPane userScrPane
private ChatLogDialog chatLogDialog
private ChatBoxListDialog chatBoxListDialog

public AdminOptionsWindow(JFrame inFrame)
public ChatLogDialog getChatLogDialog()
private JPanel setupBoxLayout()
public void setUpContentPane()
private void setupLayout()
public void setUpUserList()
public void showChatBoxListDialog(List<ChatBox> chatBoxes)
public void showChatLogDialog(List<ChatBox> chatBoxes)

UserListListener

public void valueChanged(ListSelectionEvent e)

ChatBoxListDialog

private final JList<ChatBox> chatBoxList
private final DefaultListModel<ChatBox> chatBoxListModel
private final JButton hideButton
private final JButton unhideButton
private final JButton closeButton
private final Client client

public ChatBoxListDialog(AdminOptionsWindow adminOptionsWindow, List<ChatBox> chatBoxes, Client client)
private void hideChatBox()
private void unhideChatBox()

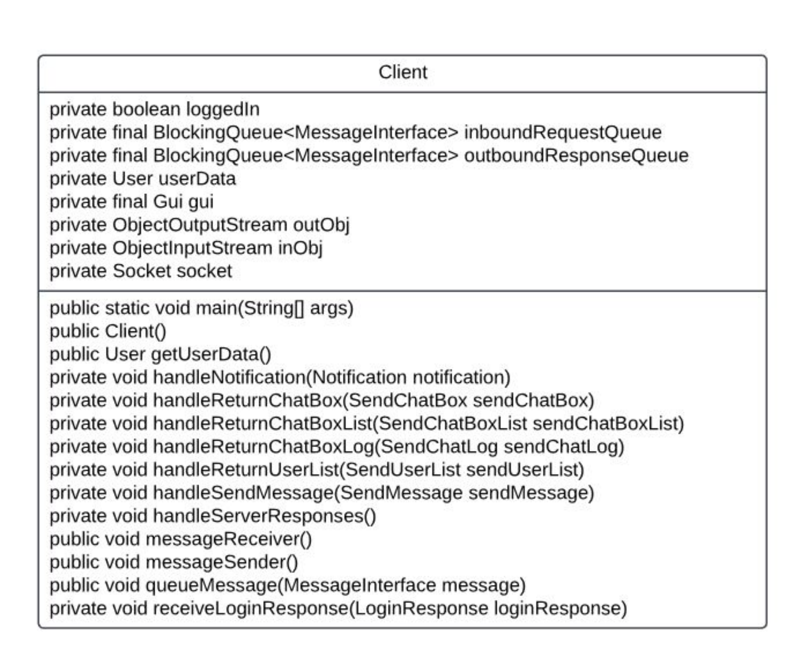
ViewChatLogButtonListener

public void actionPerformed(ActionEvent e)

BanUserButtonListener

public void actionPerformed(ActionEvent e)

Client (Class Diagrams)

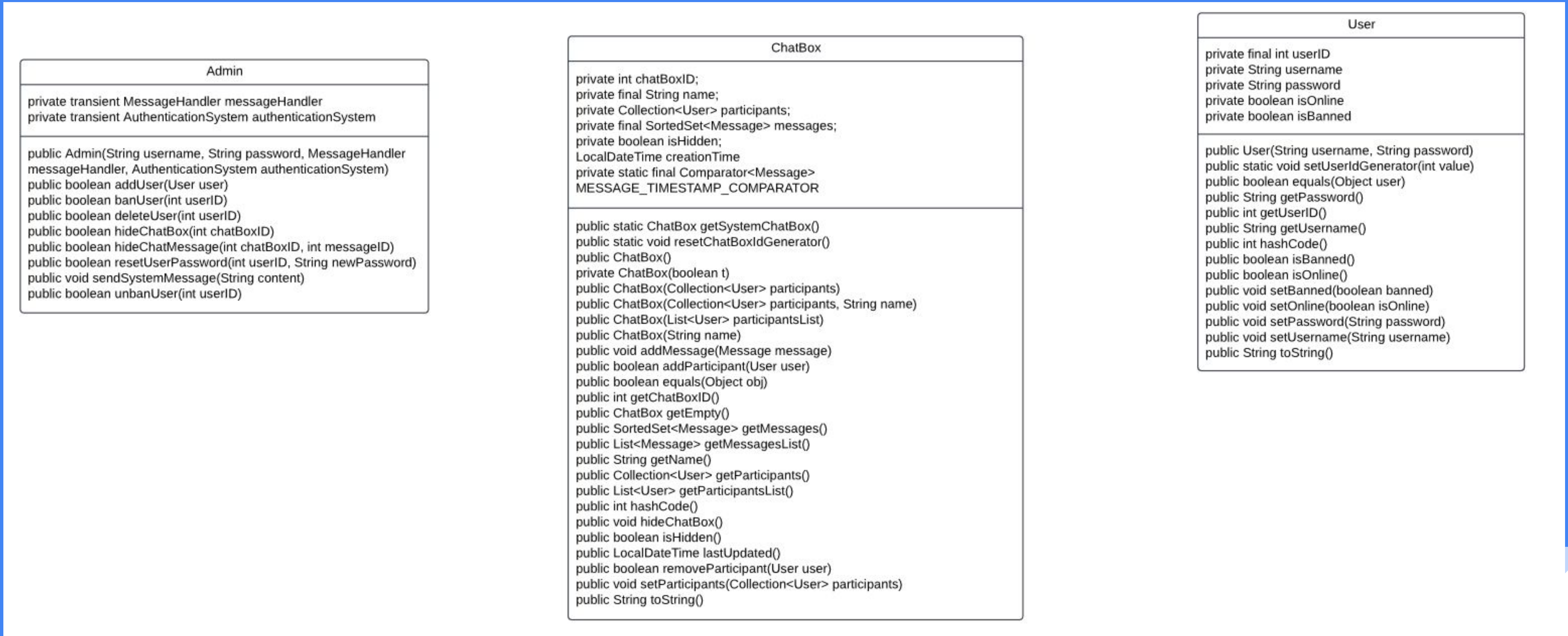


Client

private boolean loggedIn
private final BlockingQueue<MessageInterface> inboundRequestQueue
private final BlockingQueue<MessageInterface> outboundResponseQueue
private User userData
private final Gui gui
private ObjectOutputStream outObj
private ObjectInputStream inObj
private Socket socket

public static void main(String[] args)
public Client()
public User getUserData()
private void handleNotification(Notification notification)
private void handleReturnChatBox(SendChatBox sendChatBox)
private void handleReturnChatBoxList(SendChatBoxList sendChatBoxList)
private void handleReturnChatBoxLog(SendChatLog sendChatLog)
private void handleReturnUserList(SendUserList sendUserList)
private void handleSendMessage(SendMessage sendMessage)
private void handleServerResponses()
public void messageReceiver()
public void messageSender()
public void queueMessage(MessageInterface message)
private void receiveLoginResponse(LoginResponse loginResponse)

Common (Class Diagrams)



Admin

private transient MessageHandler messageHandler
private transient AuthenticationSystem authenticationSystem

public Admin(String username, String password, MessageHandler messageHandler, AuthenticationSystem authenticationSystem)
public boolean addUser(User user)
public boolean banUser(int userID)
public boolean deleteUser(int userID)
public boolean hideChatBox(int chatBoxID)
public boolean hideChatMessage(int chatBoxID, int messageID)
public boolean resetUserPassword(int userID, String newPassword)
public void sendSystemMessage(String content)
public boolean unbanUser(int userID)

ChatBox

private int chatBoxID;
private final String name;
private Collection<User> participants;
private final SortedSet<Message> messages;
private boolean isHidden;
LocalDateTime creationTime
private static final Comparator<Message> MESSAGE_TIMESTAMP_COMPARATOR

public static ChatBox getSystemChatBox()
public static void resetChatBoxIdGenerator()
public ChatBox()
private ChatBox(boolean t)
public ChatBox(Collection<User> participants)
public ChatBox(Collection<User> participants, String name)
public ChatBox(List<User> participantsList)
public ChatBox(String name)
public void addMessage(Message message)
public boolean addParticipant(User user)
public boolean equals(Object obj)
public int getChatBoxID()
public ChatBox getEmpty()
public SortedSet<Message> getMessages()
public List<Message> getMessagesList()
public String getName()
public Collection<User> getParticipants()
public List<User> getParticipantsList()
public int hashCode()
public void hideChatBox()
public boolean isHidden()
public LocalDateTime lastUpdated()
public boolean removeParticipant(User user)
public void setParticipants(Collection<User> participants)
public String toString()

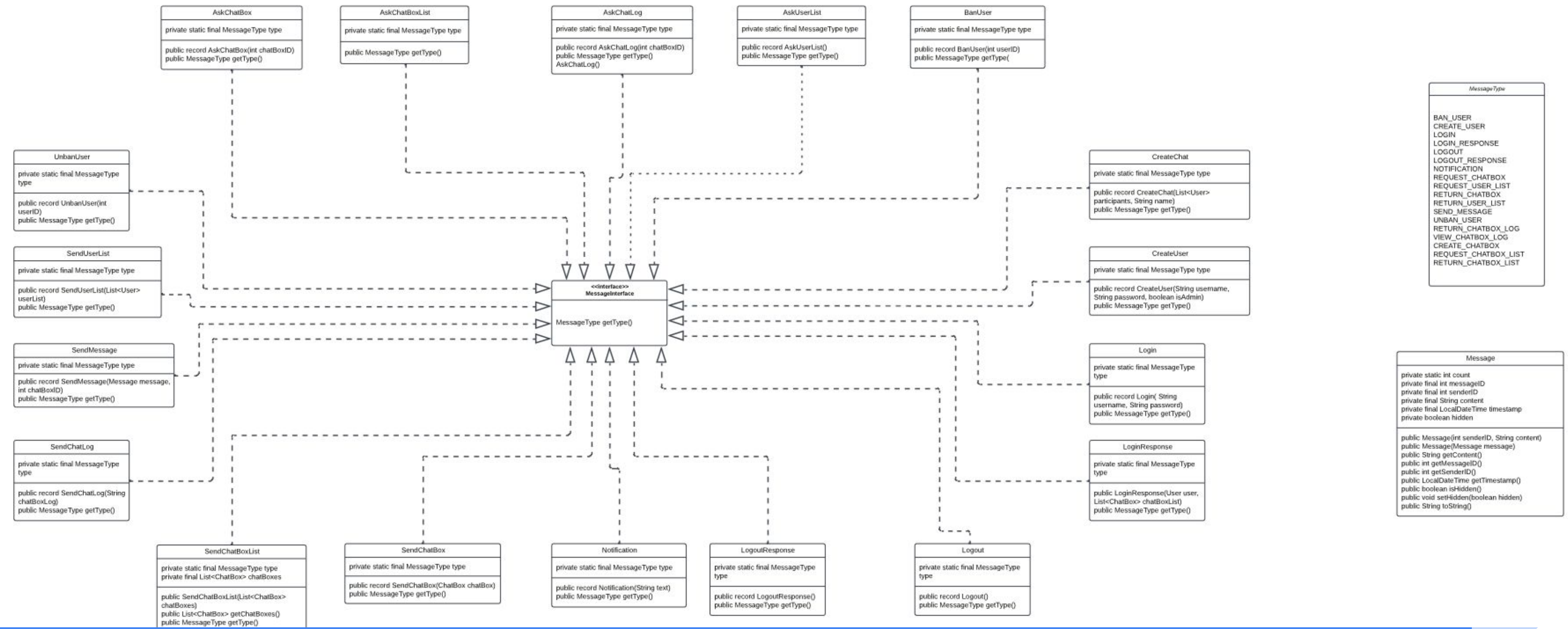
User

private final int userID
private String username
private String password
private boolean isOnline
private boolean isBanned

public User(String username, String password)
public static void setUserIdGenerator(int value)
public boolean equals(Object user)
public String getPassword()
public int getUserID()
public String getUsername()
public int hashCode()
public boolean isBanned()
public boolean isOnline()
public void setBanned(boolean banned)
public void setOnline(boolean isOnline)
public void setPassword(String password)
public void setUsername(String username)
public String toString()

Message (Class Diagrams)

Message Classes



AskChatBox
private static final MessageType type
public record AskChatBox(int chatBoxID)
public MessageType getType()

AskChatBoxList
private static final MessageType type
public MessageType getType()

AskChatLog
private static final MessageType type
public record AskChatLog(int chatBoxID)
public MessageType getType()
AskChatLog()

AskUserList
private static final MessageType type
public record AskUserList()
public MessageType getType()

BanUser
private static final MessageType type
public record BanUser(int userID)
public MessageType getType(

UnbanUser
private static final MessageType type
public record UnbanUser(int userID)
public MessageType getType()

SendUserList
private static final MessageType type
public record SendUserList(List<User> userList)
public MessageType getType()

SendMessage
private static final MessageType type
public record SendMessage(Message message, int chatBoxID)
public MessageType getType()

SendChatLog
private static final MessageType type
public record SendChatLog(String chatBoxLog)
public MessageType getType()

<<interface>>
MessageInterface
MessageType getType()

CreateChat
private static final MessageType type
public record CreateChat(List<User> participants, String name)
public MessageType getType()

CreateUser
private static final MessageType type
public record CreateUser(String username, String password, boolean isAdmin)
public MessageType getType()

Login
private static final MessageType type
public record Login(String username, String password)
public MessageType getType()

LoginResponse
private static final MessageType type
public LoginResponse(User user, List<ChatBox> chatBoxList)
public MessageType getType()

SendChatBoxList
private static final MessageType type
private final List<ChatBox> chatBoxes
public SendChatBoxList(List<ChatBox> chatBoxes)
public List<ChatBox> getChatBoxes()
public MessageType getType()

SendChatBox
private static final MessageType type
public record SendChatBox(ChatBox chatBox)
public MessageType getType()

Notification
private static final MessageType type
public record Notification(String text)
public MessageType getType()

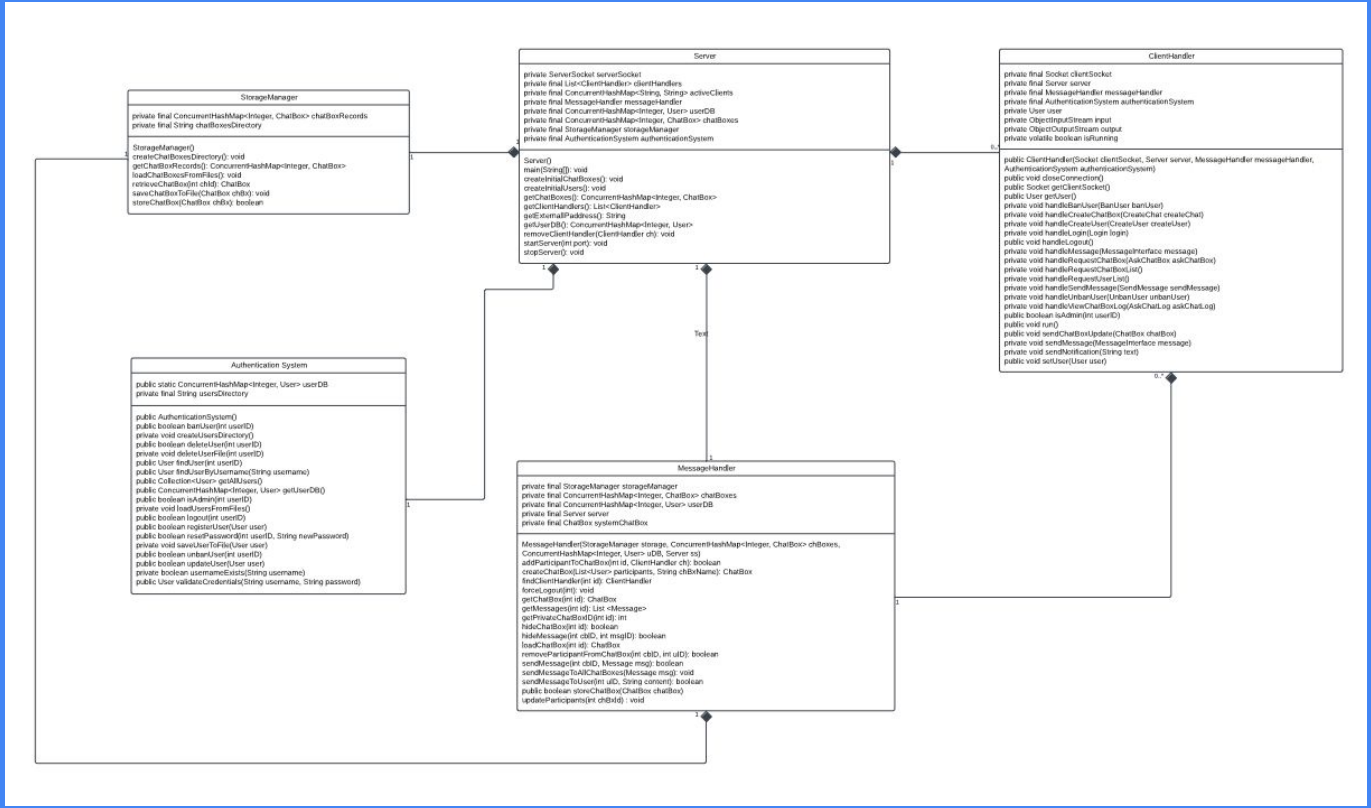
LogoutResponse
private static final MessageType type
public record LogoutResponse()
public MessageType getType()

Logout
private static final MessageType type
public record Logout()
public MessageType getType()

MessageType
BAN_USER
CREATE_USER
LOGIN
LOGIN_RESPONSE
LOGOUT
LOGOUT_RESPONSE
NOTIFICATION
REQUEST_CHATBOX
REQUEST_USER_LIST
RETURN_CHATBOX
RETURN_USER_LIST
SEND_MESSAGE
UNBAN_USER
RETURN_CHATBOX_LOG
VIEW_CHATBOX_LOG
CREATE_CHATBOX
REQUEST_CHATBOX_LIST
RETURN_CHATBOX_LIST

Message
private static int count
private final int messageID
private final int senderID
private final String content
private final LocalDateTime timestamp
private boolean hidden
public Message(int senderID, String content)
public Message(Message message)
public String getContent()
public int getMessageID()
public int getSenderID()
public LocalDateTime getTimestamp()
public boolean isHidden()
public void setHidden(boolean hidden)
public String toString()

Server (Class Diagrams)



```
StorageManager

private final ConcurrentHashMap<Integer, ChatBox> chatBoxRecords
private final String chatBoxesDirectory

StorageManager()
createChatBoxesDirectory(): void
getChatBoxRecords(): ConcurrentHashMap<Integer, ChatBox>
loadChatBoxesFromFiles(): void
retrieveChatBox(int chId): ChatBox
saveChatBoxToFile(ChatBox chBx): void
storeChatBox(ChatBox chBx): boolean
```

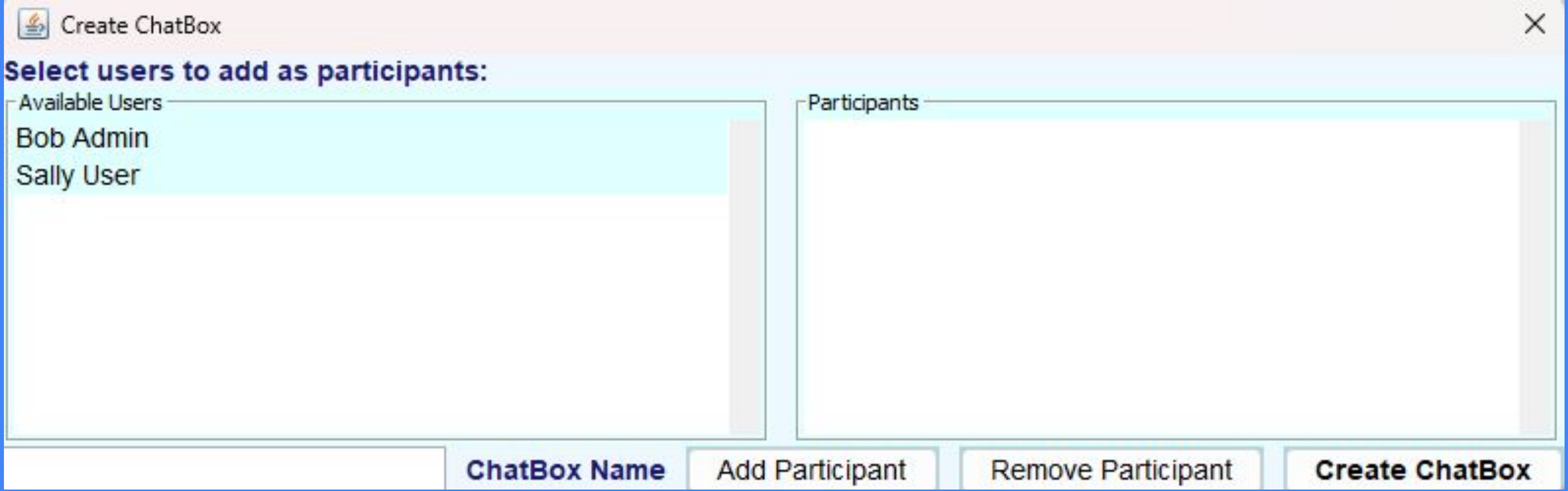
```
Server

private ServerSocket serverSocket
private final List<ClientHandler> clientHandlers
private final ConcurrentHashMap<String, String> activeClients
private final MessageHandler messageHandler
private final ConcurrentHashMap<Integer, User> userDB
private final ConcurrentHashMap<Integer, ChatBox> chatBoxes
private final StorageManager storageManager
private final AuthenticationSystem authenticationSystem

Server()
main(String[]): void
createInitialChatBoxes(): void
createInitialUsers(): void
getChatBoxes(): ConcurrentHashMap<Integer, ChatBox>
getClientHandlers(): List<ClientHandler>
getExternalIPaddress(): String
getUserDB(): ConcurrentHashMap<Integer, User>
removeClientHandler(ClientHandler ch): void
startServer(int port): void
stopServer(): void
```

```
ClientHandler

private final Socket clientSocket
private final Server server
private final MessageHandler messageHandler
private final AuthenticationSystem authenticationSystem
private User user
private ObjectInputStream input
private ObjectOutputStream output
private volatile boolean isRunning

public ClientHandler(Socket clientSocket, Server server, MessageHandler messageHandler,
AuthenticationSystem authenticationSystem)
public void closeConnection()
public Socket getClientSocket()
public User getUser()
private void handleBanUser(BanUser banUser)
private void handleCreateChatBox(CreateChat createChat)
private void handleCreateUser(CreateUser createUser)
private void handleLogin(Login login)
public void handleLogout()
private void handleMessage(MessageInterface message)
private void handleRequestChatBox(AskChatBox askChatBox)
private void handleRequestChatBoxList()
private void handleRequestUserList()
private void handleSendMessage(SendMessage sendMessage)
private void handleUnbanUser(UnbanUser unbanUser)
private void handleViewChatBoxLog(AskChatLog askChatLog)
public boolean isAdmin(int userID)
public void run()
public void sendChatBoxUpdate(ChatBox chatBox)
private void sendMessage(MessageInterface message)
private void sendNotification(String text)
public void setUser(User user)
```

```
Authentication System

public static ConcurrentHashMap<Integer, User> userDB
private final String usersDirectory

public AuthenticationSystem()
public boolean banUser(int userID)
private void createUsersDirectory()
public boolean deleteUser(int userID)
private void deleteUserFile(int userID)
public User findUser(int userID)
public User findUserByUsername(String username)
public Collection<User> getAllUsers()
public ConcurrentHashMap<Integer, User> getUserDB()
public boolean isAdmin(int userID)
private void loadUsersFromFiles()
public boolean logout(int userID)
public boolean registerUser(User user)
public boolean resetPassword(int userID, String newPassword)
private void saveUserToFile(User user)
public boolean unbanUser(int userID)
public boolean updateUser(User user)
private boolean usernameExists(String username)
public User validateCredentials(String username, String password)
```

```
MessageHandler

private final StorageManager storageManager
private final ConcurrentHashMap<Integer, ChatBox> chatBoxes
private final ConcurrentHashMap<Integer, User> userDB
private final Server server
private final ChatBox systemChatBox

MessageHandler(StorageManager storage, ConcurrentHashMap<Integer, ChatBox> chBoxes,
ConcurrentHashMap<Integer, User> uDB, Server ss)
addParticipantToChatBox)int id, ClientHandler ch): boolean
createChatBox(List<User> participants, String chBxName): ChatBox
findClientHandler(int id): ClientHandler
forceLogout(int): void
getChatBox(int id): ChatBox
getMessages(int id): List <Message>
getPrivateChatBoxID(int id): int
hideChatBox(int id): boolean
hideMessage(int cbID, int msgID): boolean
loadChatBox(int id): ChatBox
removeParticipantFromChatBox(int cbID, int uID): boolean
sendMessage(int cbID, Message msg): boolean
sendMessageToAllChatBoxes(Message msg): void
sendMessageToUser(int uID, String content): boolean
public boolean storeChatBox(ChatBox chatBox)
updateParticipants(int chBxId) : void
```

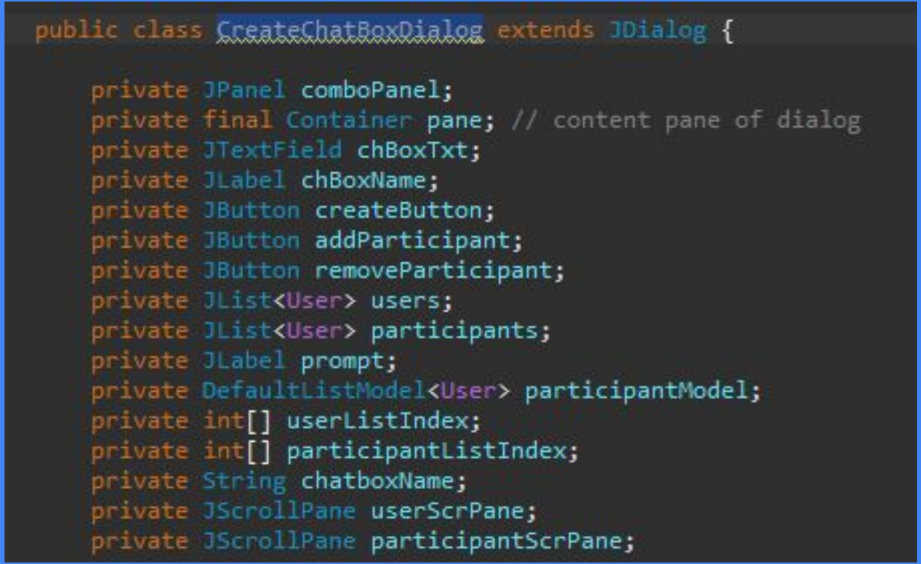
Implementation - GUI



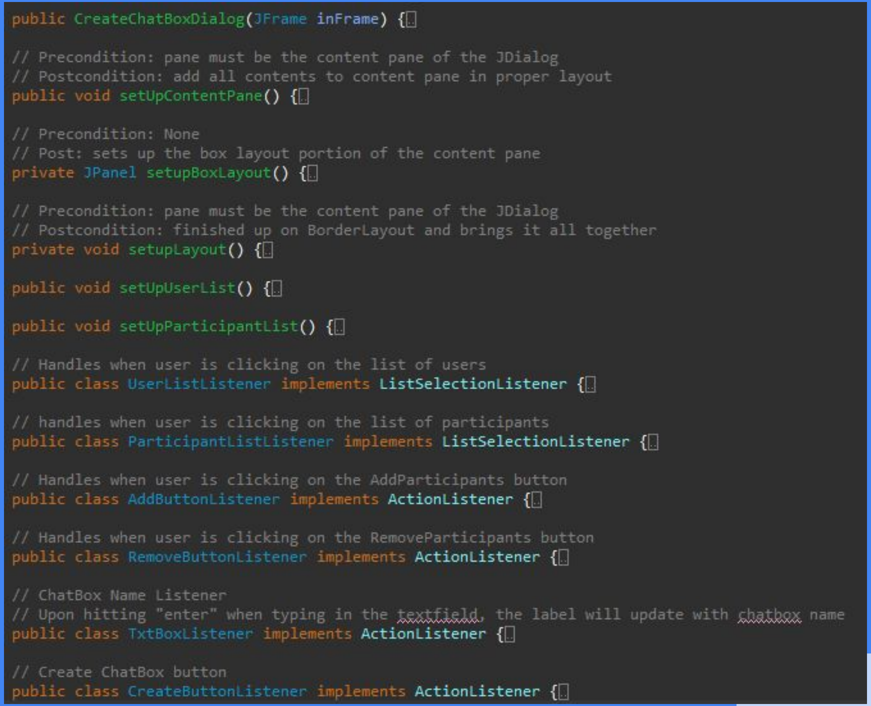
Select users to add as participants:

Bob Admin
Sally User

Implementation - GUI



```java
public class CreateChatBoxDialog extends JDialog {

    private JPanel comboPanel;
    private final Container pane; // content pane of dialog
    private JTextField chBoxTxt;
    private JLabel chBoxName;
    private JButton createButton;
    private JButton addParticipant;
    private JButton removeParticipant;
    private JList<User> users;
    private JList<User> participants;
    private JLabel prompt;
    private DefaultListModel<User> participantModel;
    private int[] userListIndex;
    private int[] participantListIndex;
    private String chatboxName;
    private JScrollPane userScrPane;
    private JScrollPane participantScrPane;
```

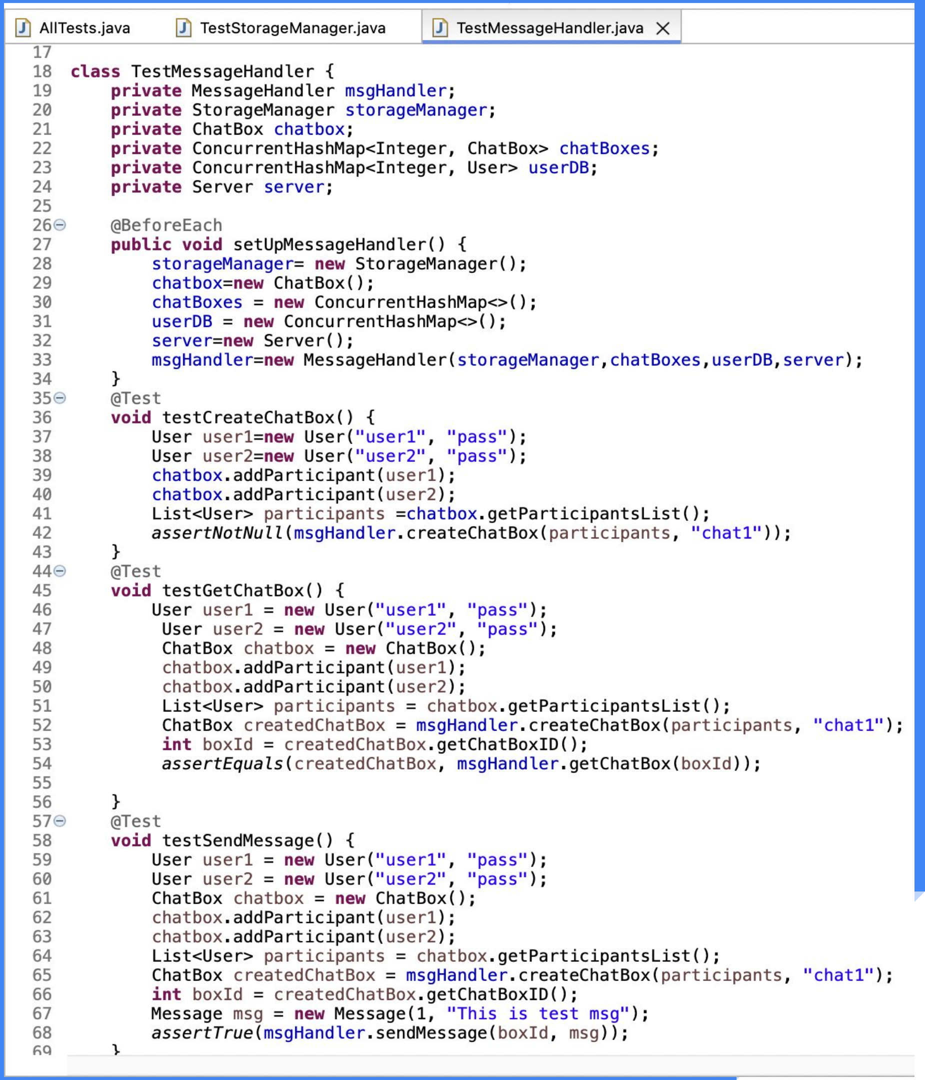


```java
public CreateChatBoxDialog(JFrame inFrame) {□

// Precondition: pane must be the content pane of the JDialog
// Postcondition: add all contents to content pane in proper layout
public void setUpContentPane() {□

// Precondition: None
// Post: sets up the box layout portion of the content pane
private JPanel setupBoxLayout() {□

// Precondition: pane must be the content pane of the JDialog
// Postcondition: finished up on BorderLayout and brings it all together
private void setupLayout() {□

public void setUpUserList() {□

public void setUpParticipantList() {□

// Handles when user is clicking on the list of users
public class UserListListener implements ListSelectionListener {□

// handles when user is clicking on the list of participants
public class ParticipantListListener implements ListSelectionListener {□

// Handles when user is clicking on the AddParticipants button
public class AddButtonListener implements ActionListener {□

// Handles when user is clicking on the RemoveParticipants button
public class RemoveButtonListener implements ActionListener {□

// ChatBox Name Listener
// Upon hitting "enter" when typing in the textfield, the label will update with chatbox name
public class TxtBoxListener implements ActionListener {□

// Create ChatBox button
public class CreateButtonListener implements ActionListener {□
```

Testing

What we tested:

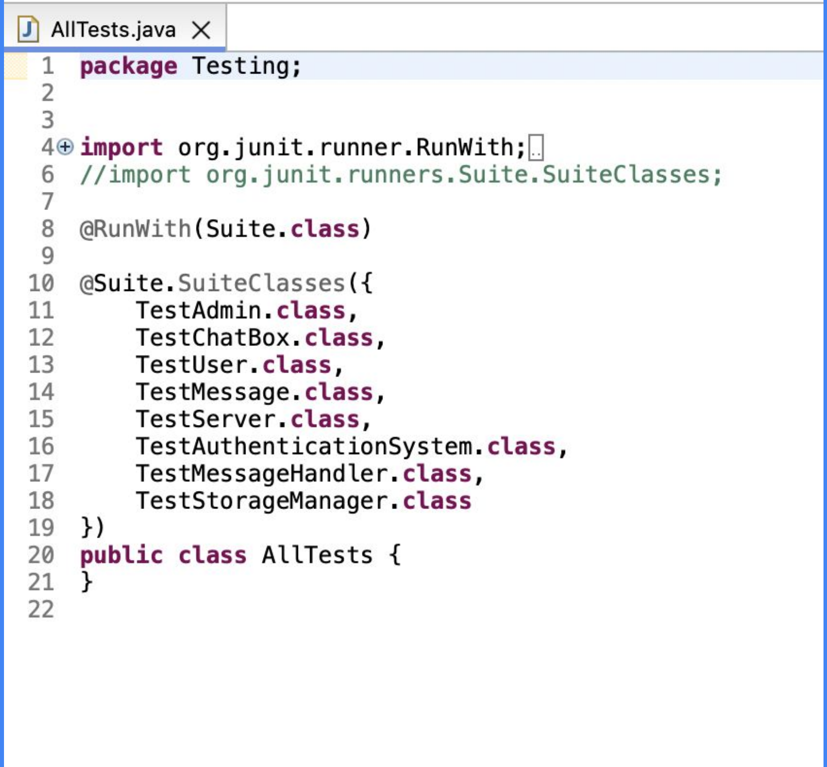
- Authentication (login, registration, banned users)
- Chat Boxes (creating, retrieving, managing participants)
- Message Handling (sending out messages, retrieving logs)
- Storage management (saving data, retrieving data)
- Admin
- Message
- User



```java

class TestMessageHandler {
    private MessageHandler msgHandler;
    private StorageManager storageManager;
    private ChatBox chatbox;
    private ConcurrentHashMap<Integer, ChatBox> chatBoxes;
    private ConcurrentHashMap<Integer, User> userDB;
    private Server server;

    @BeforeEach
    public void setUpMessageHandler() {
        storageManager= new StorageManager();
        chatbox=new ChatBox();
        chatBoxes = new ConcurrentHashMap<>();
        userDB = new ConcurrentHashMap<>();
        server=new Server();
        msgHandler=new MessageHandler(storageManager,chatBoxes,userDB,server);
    }
    @Test
    void testCreateChatBox() {
        User user1=new User("user1", "pass");
        User user2=new User("user2", "pass");
        chatbox.addParticipant(user1);
        chatbox.addParticipant(user2);
        List<User> participants =chatbox.getParticipantsList();
        assertNotNull(msgHandler.createChatBox(participants, "chat1"));
    }
    @Test
    void testGetChatBox() {
        User user1 = new User("user1", "pass");
         User user2 = new User("user2", "pass");
         ChatBox chatbox = new ChatBox();
         chatbox.addParticipant(user1);
         chatbox.addParticipant(user2);
         List<User> participants = chatbox.getParticipantsList();
         ChatBox createdChatBox = msgHandler.createChatBox(participants, "chat1");
         int boxId = createdChatBox.getChatBoxID();
         assertEquals(createdChatBox, msgHandler.getChatBox(boxId));

    }
    @Test
    void testSendMessage() {
        User user1 = new User("user1", "pass");
        User user2 = new User("user2", "pass");
        ChatBox chatbox = new ChatBox();
        chatbox.addParticipant(user1);
        chatbox.addParticipant(user2);
        List<User> participants = chatbox.getParticipantsList();
        ChatBox createdChatBox = msgHandler.createChatBox(participants, "chat1");
        int boxId = createdChatBox.getChatBoxID();
        Message msg = new Message(1, "This is test msg");
        assertTrue(msgHandler.sendMessage(boxId, msg));
    }
```

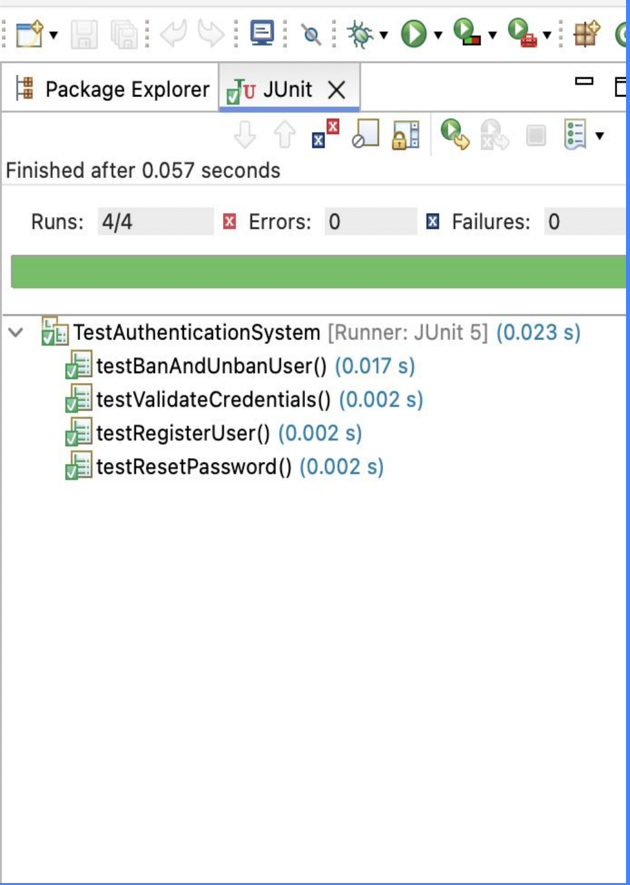
Testing - cont

Test Suite

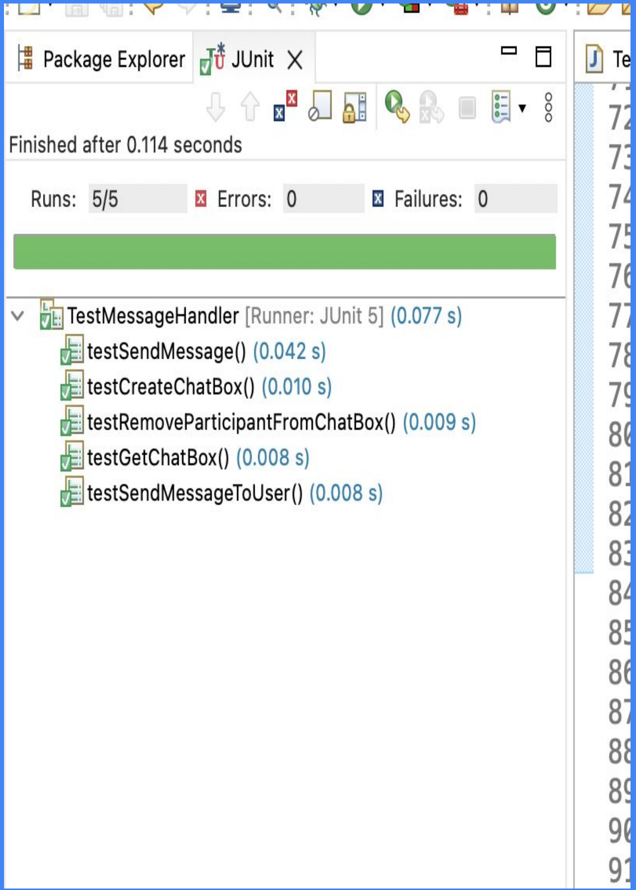


```java
package Testing;


import org.junit.runner.RunWith;
//import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)

@Suite.SuiteClasses({
    TestAdmin.class,
    TestChatBox.class,
    TestUser.class,
    TestMessage.class,
    TestServer.class,
    TestAuthenticationSystem.class,
    TestMessageHandler.class,
    TestStorageManager.class
})
public class AllTests {
}

```

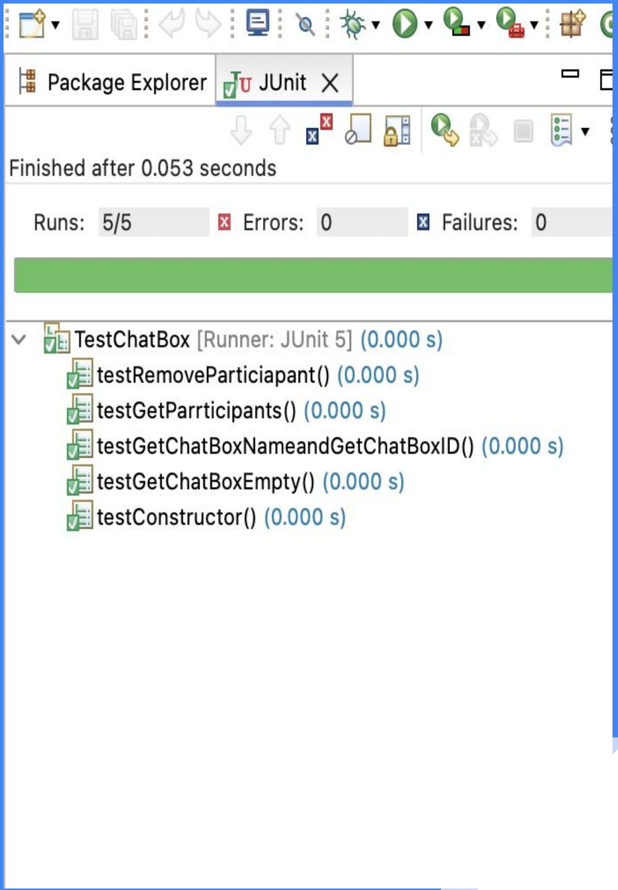
Test Results



```
Finished after 0.057 seconds

Runs:  4/4     Errors:  0     Failures:  0

TestAuthenticationSystem [Runner: JUnit 5] (0.023 s)
    testBanAndUnbanUser() (0.017 s)
    testValidateCredentials() (0.002 s)
    testRegisterUser() (0.002 s)
    testResetPassword() (0.002 s)
```

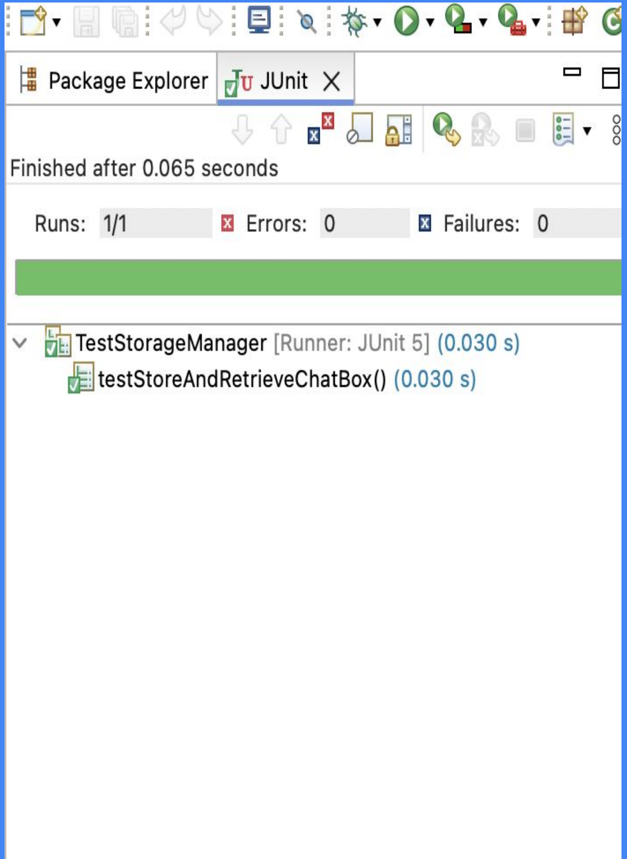


```
Finished after 0.114 seconds

Runs:  5/5     Errors:  0     Failures:  0

TestMessageHandler [Runner: JUnit 5] (0.077 s)
    testSendMessage() (0.042 s)
    testCreateChatBox() (0.010 s)
    testRemoveParticipantFromChatBox() (0.009 s)
    testGetChatBox() (0.008 s)
    testSendMessageToUser() (0.008 s)
```

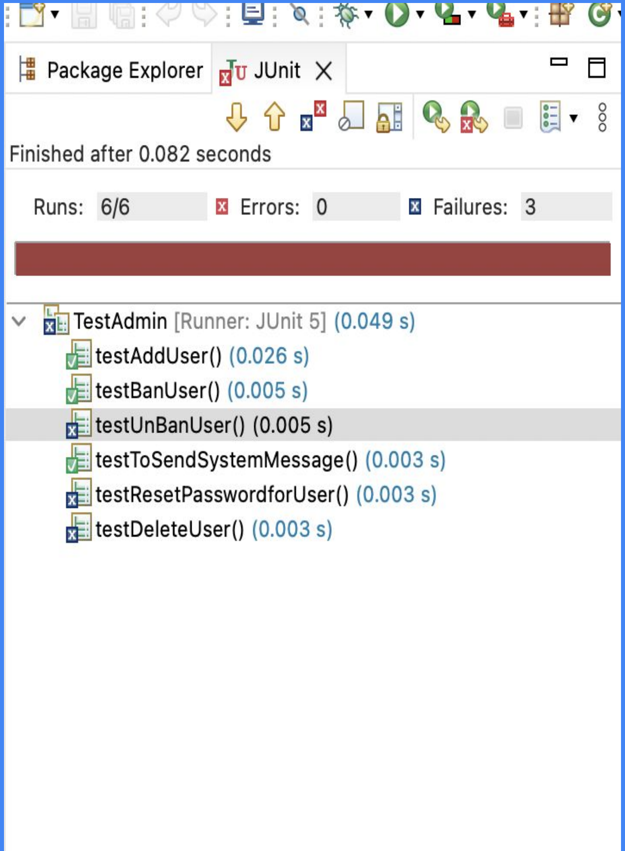


```
Finished after 0.053 seconds

Runs:  5/5     Errors:  0     Failures:  0

TestChatBox [Runner: JUnit 5] (0.000 s)
    testRemoveParticiapant() (0.000 s)
    testGetParrticipants() (0.000 s)
    testGetChatBoxNameandGetChatBoxID() (0.000 s)
    testGetChatBoxEmpty() (0.000 s)
    testConstructor() (0.000 s)
```

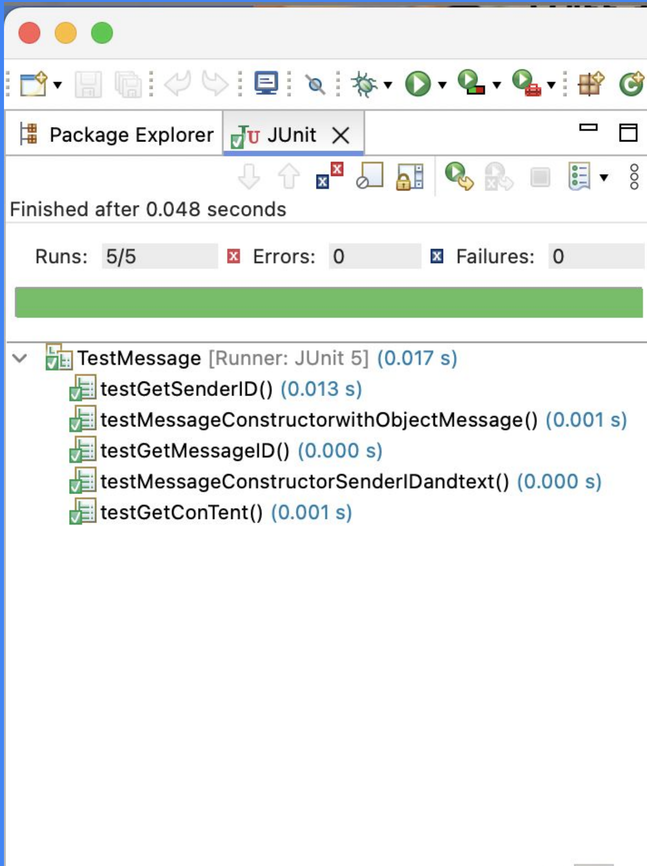
Test Results



```
Finished after 0.065 seconds

Runs:  1/1      Errors:  0      Failures:  0

TestStorageManager [Runner: JUnit 5] (0.030 s)
    testStoreAndRetrieveChatBox() (0.030 s)
```



```
Finished after 0.082 seconds

Runs:  6/6      Errors:  0      Failures:  3

TestAdmin [Runner: JUnit 5] (0.049 s)
    testAddUser() (0.026 s)
    testBanUser() (0.005 s)
    testUnBanUser() (0.005 s)
    testToSendSystemMessage() (0.003 s)
    testResetPasswordforUser() (0.003 s)
    testDeleteUser() (0.003 s)
```



```
Finished after 0.048 seconds

Runs:  5/5      Errors:  0      Failures:  0

TestMessage [Runner: JUnit 5] (0.017 s)
    testGetSenderID() (0.013 s)
    testMessageConstructorwithObjectMessage() (0.001 s)
    testGetMessageID() (0.000 s)
    testMessageConstructorSenderIDandtext() (0.000 s)
    testGetConTent() (0.001 s)
```