

**Assignment No 1**

|  |  |
| --- | --- |
| Course Title | Object Oriented Programming |

**Submitted By**

|  |  |
| --- | --- |
| Name | Sami Ur Rehman |
| Registration No | SP24-BSE-086 |
| Section | B |

**Submitted To**

|  |  |
| --- | --- |
| Moderator | Muhammad Shahid Bhatti |

**Date: 11/10/2024**

**A Console Based Messaging App**

**1. Project Overview**

This project is a Messaging System that simulates a basic messaging app. It allows users to:

* Manage a list of contacts.
* Block and unblock contacts.
* Send and view text messages.
* Delete messages.
* Keep track of unread and read messages.
* View chat history for each contact.

The application includes two main classes:

* **Sms:** Represents a text message.
* **MessagingSystem:** Manages the contacts, messages, and user interactions.

The system preloads contacts and allows users to interact with them through a menu-driven interface.

**2. Class Descriptions**

**Class: Sms**

This class represents a text message and is responsible for handling the creation of messages, setting their status, and displaying their details.

**Attributes:**

* **content:** Stores the text of the message.
* **timestamp:** Stores the date and time when the message was created.
* **isRead**: Tracks whether the message has been read or not.
* **idCounter:** A static variable used to generate unique message IDs.
* **messageId:** A unique ID for each message generated using idCounter.

**Constructor:**

* **Sms(String content):** Initializes the message content, sets the timestamp, marks the message as unread, and generates a unique message ID.

**Methods:**

* **markAsRead():** Marks the message as read.
* **getMessageId():** Returns the unique message ID.
* **getTimestamp():** Returns the timestamp of the message.
* **getMessageStatus():** Returns "Read" or "Unread" based on the message's read status.
* **toString():** Formats and returns the message details, including its content, status, timestamp, and ID.

**Class: MessagingSystem**

This class simulates a basic messaging system. It manages contacts, blocks/unblocks them, handles text messaging, and stores chats.

**Attributes:**

* **contacts:** A 2D array to store contact details (ID, name, phone number).
* **blocked:** A 2D array to store blocked contacts.
* **chats:** A 2D array to store messages sent to or received from contacts.
* **contactCount:** Tracks the total number of contacts.
* **blockCount:** Tracks the number of blocked contacts.
* **chatCounts:** An array to track the number of chats for each contact.
* **senderName:** The name of the sender (hardcoded as "Sami").
* **senderNumber:** The phone number of the sender.

**Constructor Block:**

* Preloads the system with five pre-added contacts.

**Methods**

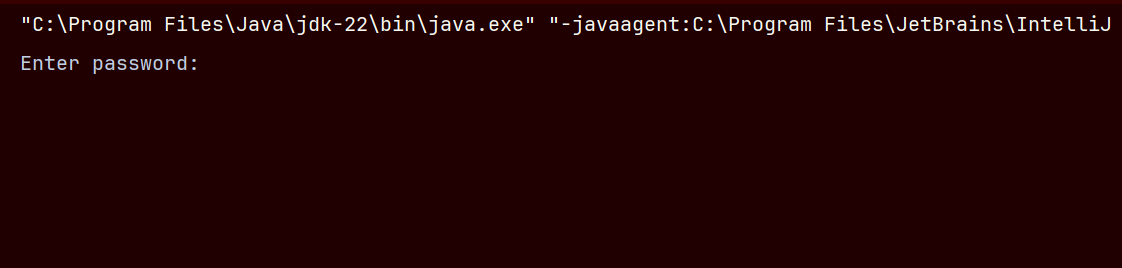
* **addContact(String name, String number):** Adds a new contact to the system.
* **preAddedContacts(String name, String number**): Adds a contact during initialization.
* **searchContactIndexByName(String name):** Finds the index of a contact by its name.
* **searchBlockIndexByName(String name):** Finds the index of a blocked contact by its name.
* **deleteContact(String name):** Removes a contact from the system.
* **viewContacts():** Displays the list of all saved contacts.
* **blockContact(String name):** Blocks a contact.
* **viewBlocked():** Displays a list of all blocked contacts.
* **unblockContact(String name):** Unblocks a contact.
* **isBlocked(String name):** Checks if a contact is blocked.
* **startChat(String name):** Starts a chat session with a contact.
* **viewChats(String name):** Displays the chat history with a contact.
* **deleteMessage(String name, String messageId):** Deletes a specific message by its ID from the chat history of a contact.

**3. Execution and Output Examples**

To demonstrate the working of the application, I'll show sample outputs for each functionality.

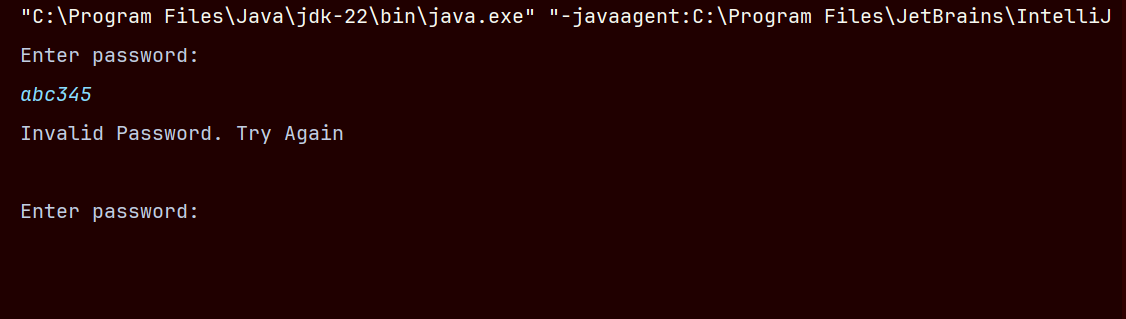
**1. Asking for password:**

In the start, user have to input password to use this console based app.

****

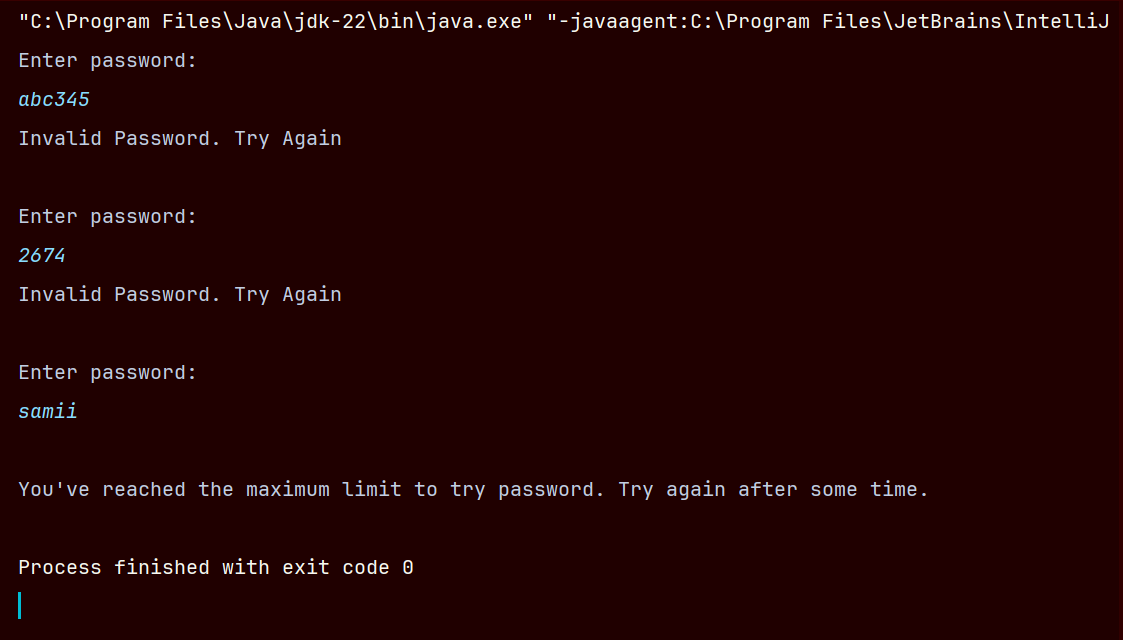
**2. Entering invalid password:**

If the password entered is incorrect, it displays invalid password and ask the user to try again.

****

**3. Entering invalid password for 3 times:**

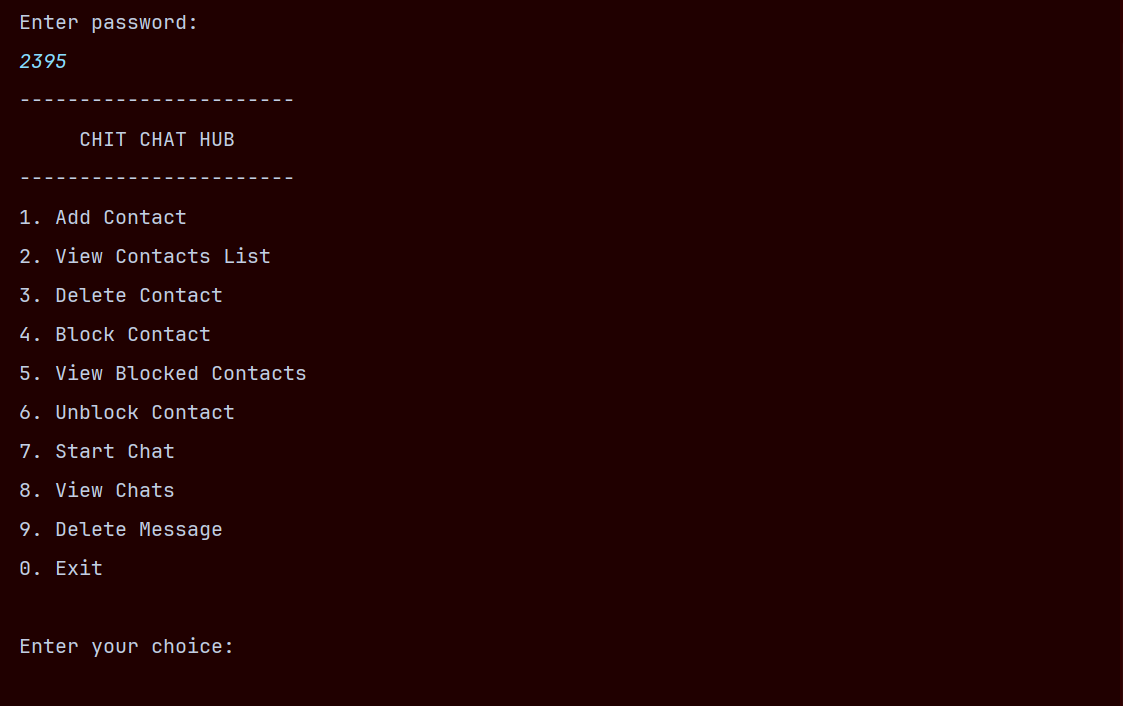
If user enters incorrect password for 3 times, program terminates and displays maximum limit reached to try password.

****

**4. Entering correct password**:

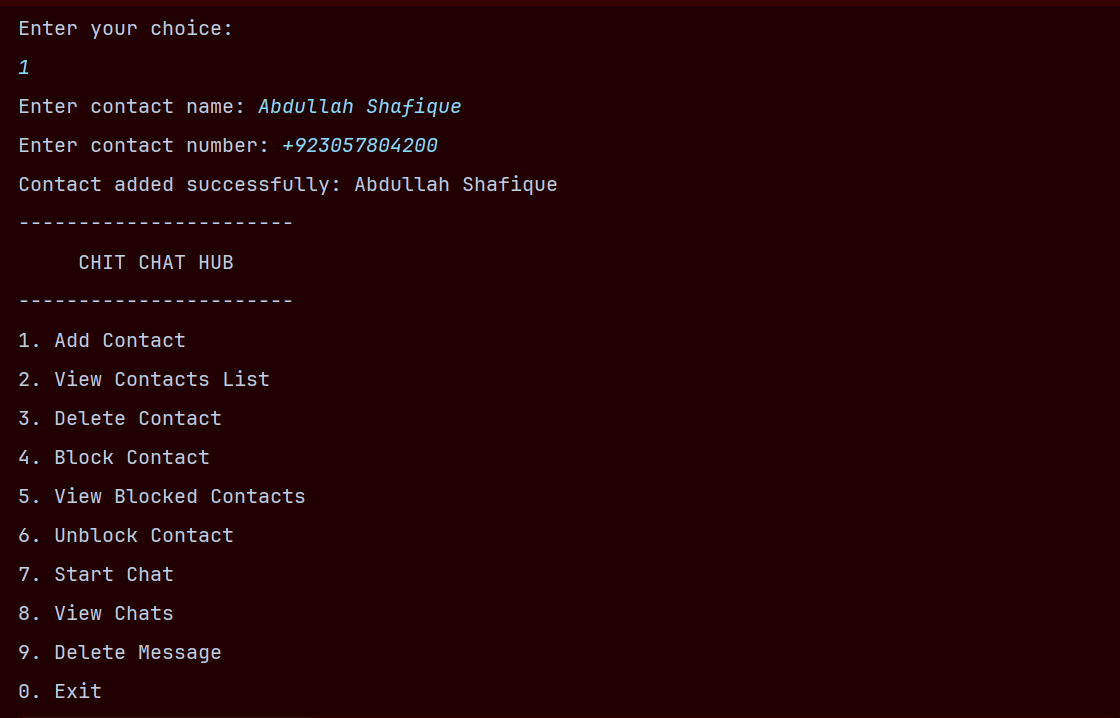
And when the user enters the right password, app opens and shows the menu.

The password I set in the program is (2395), and when user enters 2395 it starts the app.



**5. Adding Contact:**

To add contact user has to enter the name and number.

****

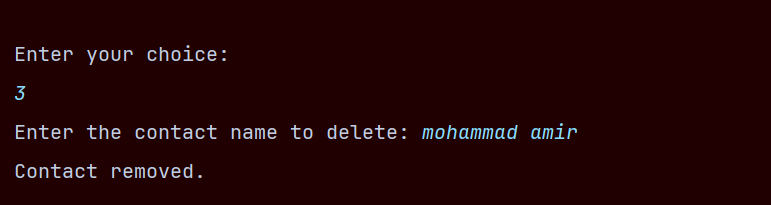
**6. View Contact List:**

After adding the contact, view list shows both pre-added contacts as well as user added contact at run time.

****

**7. Delete Contact:**

User can delete the contact by entering the name of a person whose contact user wants to delete.

****

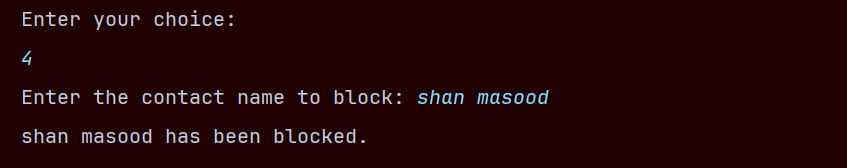
**8. View Contact List after deleting contact:**

Here we can see, the contact is removed from the contact list.

****

**9. Block Contact:**

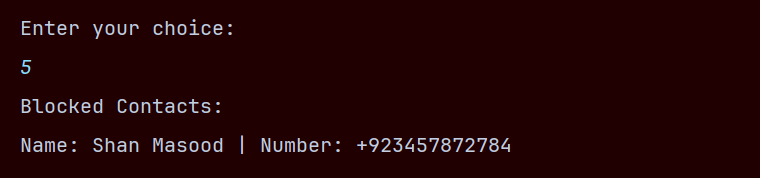
User can block the contact by entering the name. After blocking, program won’t allow to start a chat with a blocked contact. Instead it prints that the person is blocked.

****

**10. View Blocked Contact:**

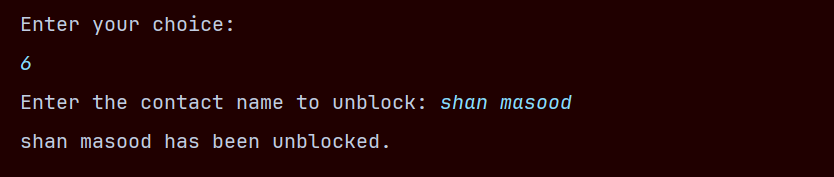
User can see the blocked list by entering 5.

And if the contact is blocked we can’t start a chat with blocked contact. If we enters the name of a blocked contact, it prints “The person is blocked”.

****

**11. Unblock Contact:**

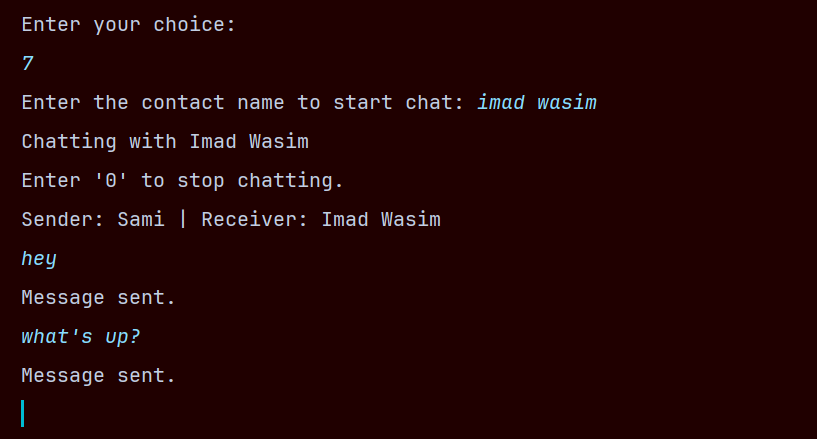
Blocked contacts can be unblocked by this function.

****

**12. Start Chat:**

Chat can be started by entering the name of a person present in your contact list.

By entering 0, chat will be stopped.

****

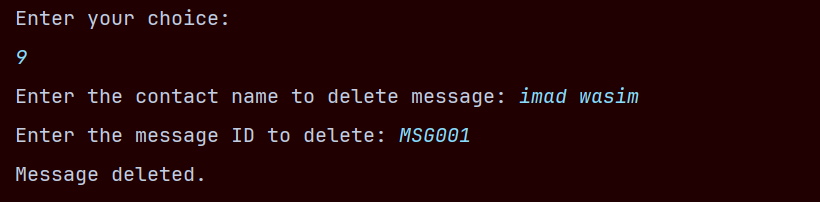
**13. View Chat:**

Chats can be viewed by entering the name of a specific person whose chat we wants to see.

****

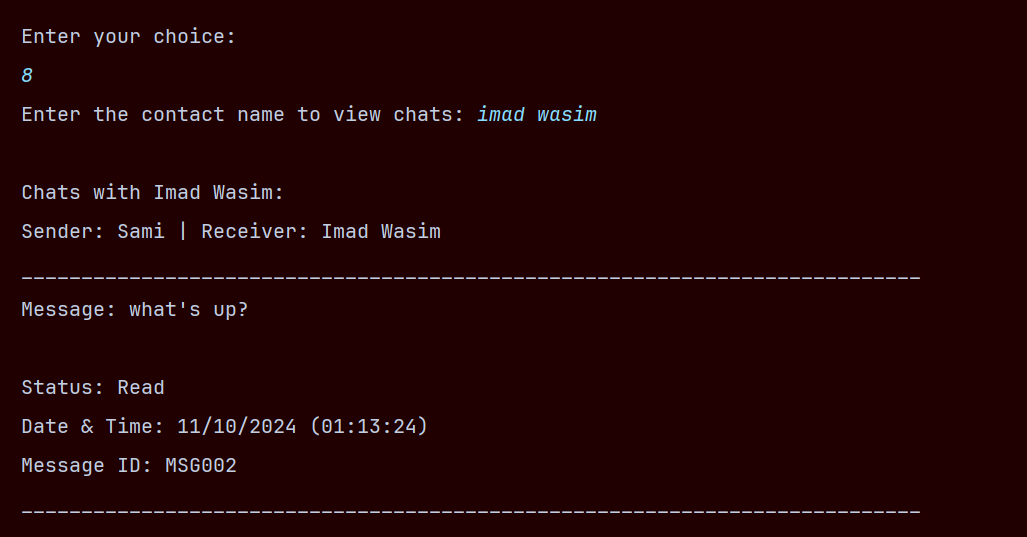
**14. Delete Message:**

In order to delete a message with anyone, first we have to give the name of a person and then the message id, that we want to be deleted.

****

**15. View Chat after deleting message:**

Here we can see that the message is deleted.  
Moreover, when we viewed the chat for the second time, the status is changed to “Read”, because we’ve already viewed the message and now viewing it for the second time.  
This program changes the status to read when we viewed the chat for once, after that it shows “Read”.

****

**Conclusion:**

This project demonstrates a simple messaging system with contacts management, chat functionality, and the ability to block/unblock contacts. The system is menu-driven, allowing users to interact with various features in a user-friendly way.