

**Name: Fatima Khalid**

**Roll no: SP24-BSE-132**

**Course name: Object Oriented Programming**

**Submitted to: Muhammad Shahid Bhatti**

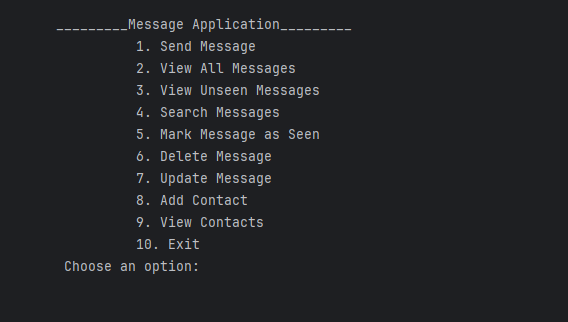
**Assignment no: 01**

**Due Date: 11-10-2024**

**Messaging App:**

**Console:**

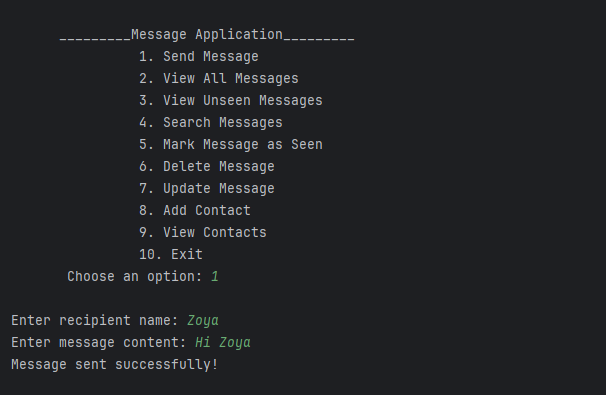
A console is an interface that allows users to interact with a computer system or application by typing text commands and receiving text-based feedback. It is often used for debugging, managing systems, or running command-line programs. In programming, the console is typically accessed through a terminal or command-line interface (CLI), where users can input commands and see outputs directly. In languages like Java, the console is often used to capture user input (via classes like Scanner) and display output (using System.out.println()). The console provides a direct, no-graphical way of interacting with programs, especially during development.

****

**Main Functions in My App:**

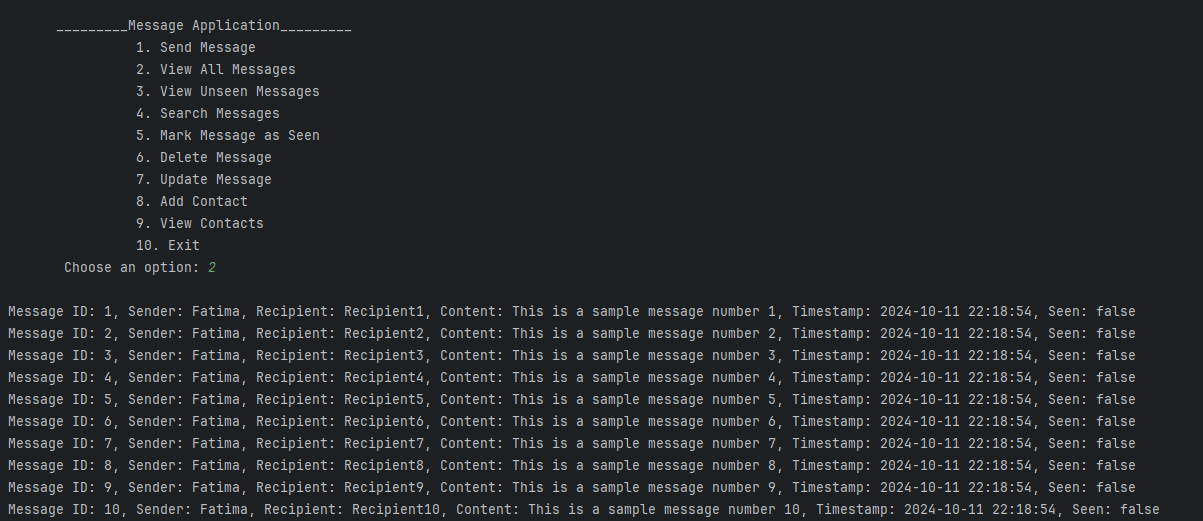
Following are the functions in my code:

1. **Send Message:**

**addMessage (int id, String sender, String recipient, String content, String timestamp)**:   
This function adds a new message to the messages 2D array. It takes the message's ID, sender, recipient, content, and timestamp as inputs and stores them in the array. Each message also includes a "seen" status, which is set to "false" by default, meaning the message is unseen when first sent. The method looks for an empty slot in the array to insert the new message.

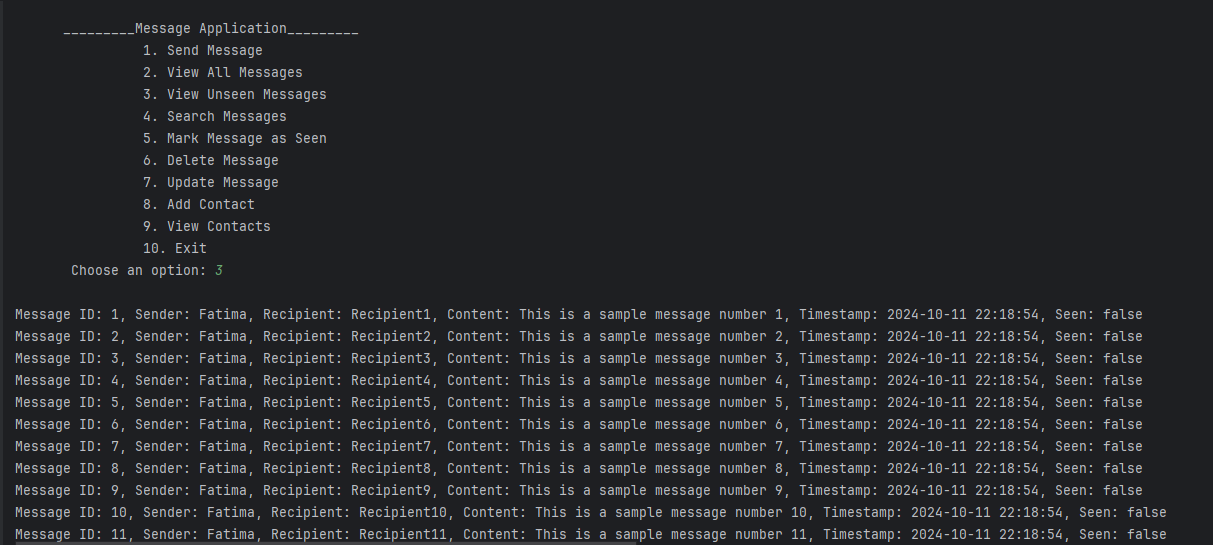
1. **View all messages:**

**displayMessages()**:  
This function loops through the messages array and prints out all the messages stored in it, including the message ID, sender, recipient, content, timestamp, and whether it has been seen or not. If no messages are found, it displays a message indicating that. This method helps users view all messages in the system.



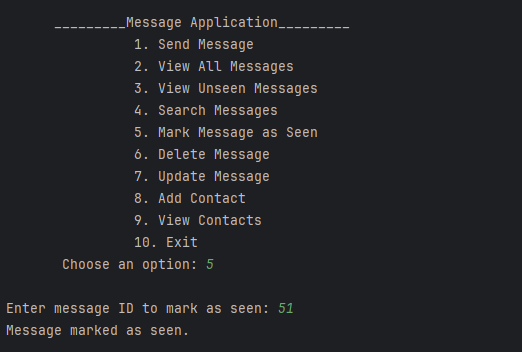
1. **View unseen messages:**

**displayUnseenMessages ()**:   
This function iterates over the messages array and prints out only the messages that are marked as "unseen" (where the seen status is "false"). It allows users to view unread messages, helping them quickly find new or unread messages.

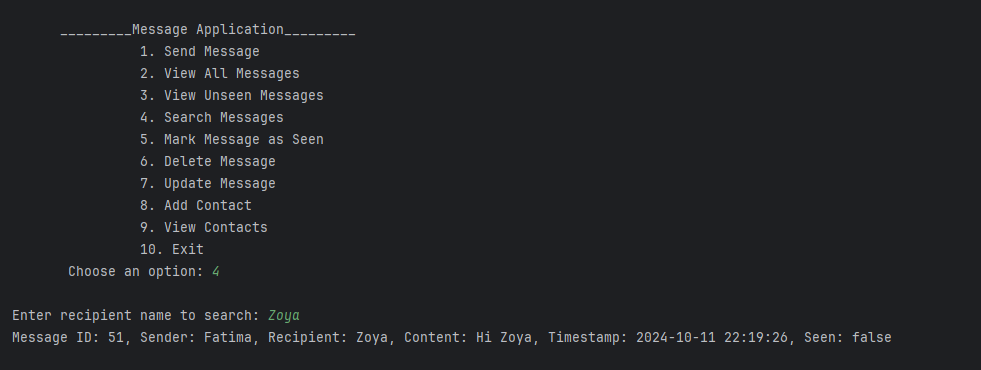


1. **Mark message as seen:**

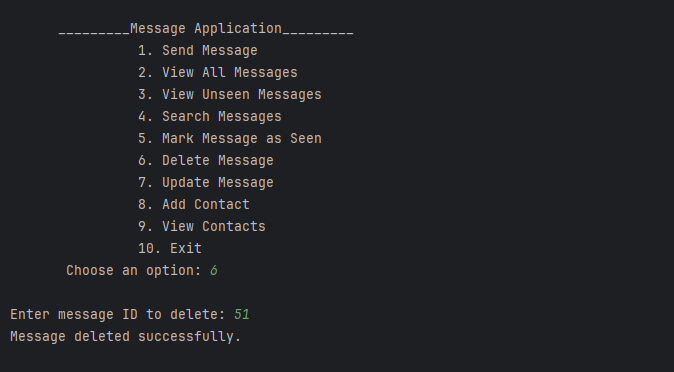
**markMessageAsSeen(int id)**:   
This function marks a specific message as "seen" by searching for the message with the given ID in the messages array. When it finds the message, it updates its seen status to "true." This feature allows users to track which messages have been read or acknowledged.



1. **Search Message:**

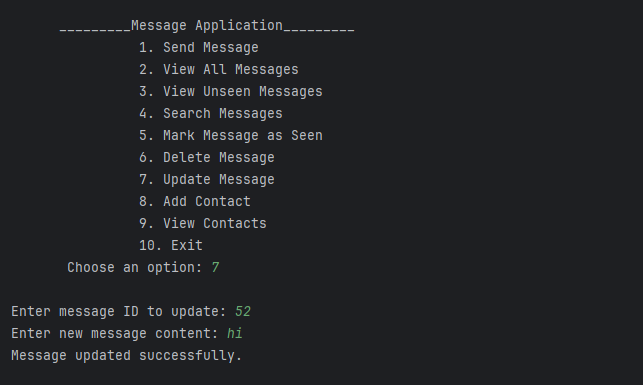
**searchMessagesByRecipient(String recipient)**:  
This function searches the messages array for messages sent to a particular recipient by matching the recipient's name (case-insensitive). It prints all the matching messages, helping users filter messages by recipient and quickly locate relevant conversations.

1. **Delete Message:**

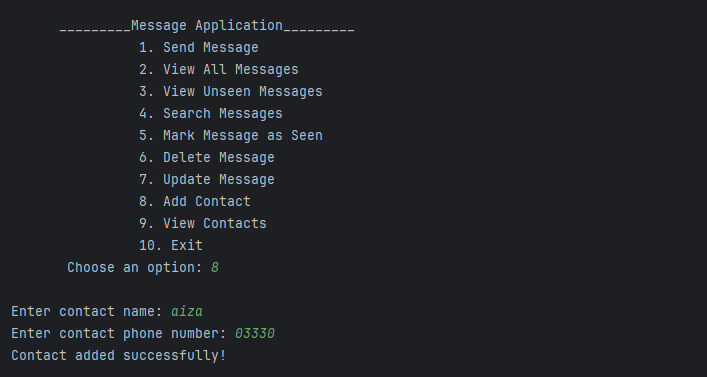
**deleteMessage(int id)**:   
This function deletes a message by its ID. It searches the messages array for the message with the given ID, and if found, it clears all data in the respective row, effectively removing the message from the system. This method allows users to remove unwanted or outdated messages.

1. **Update Message:**

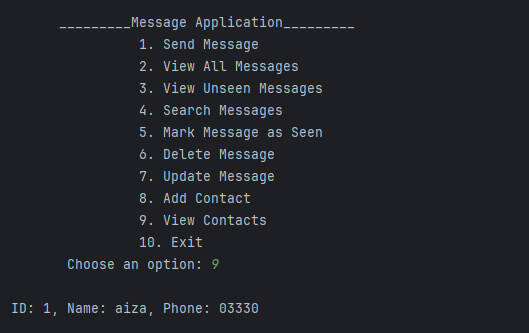
**updateMessageById(int id, String newContent)**:   
This function updates the content of an existing message by its ID. It searches for the message in the messages array and replaces its content with the new content provided by the user. This feature is useful when users want to edit the text of a message.



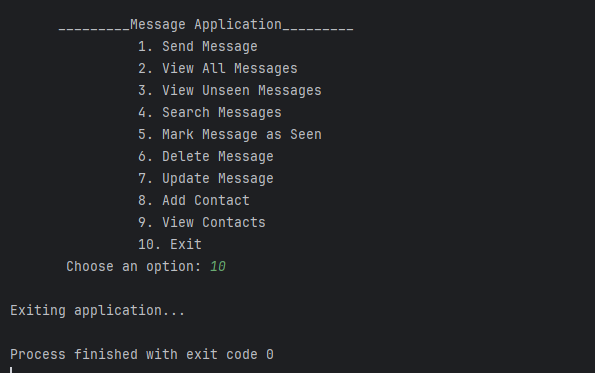
1. **Add Contact:**

**addContact (Contact contact):**  
This function takes a Contact object as an argument and adds it to the contacts list, which stores all the contacts. Each contact contains an ID, name, and phone number. This method allows users to add new contacts to the system so they can easily manage and message their contacts.

1. **View Contacts:**

**displayContacts ()**:   
This function prints out all contacts stored in the contacts list, including their ID, name, and phone number. If no contacts are available, it informs the user. This method allows users to view their saved contacts and easily access their contact information.

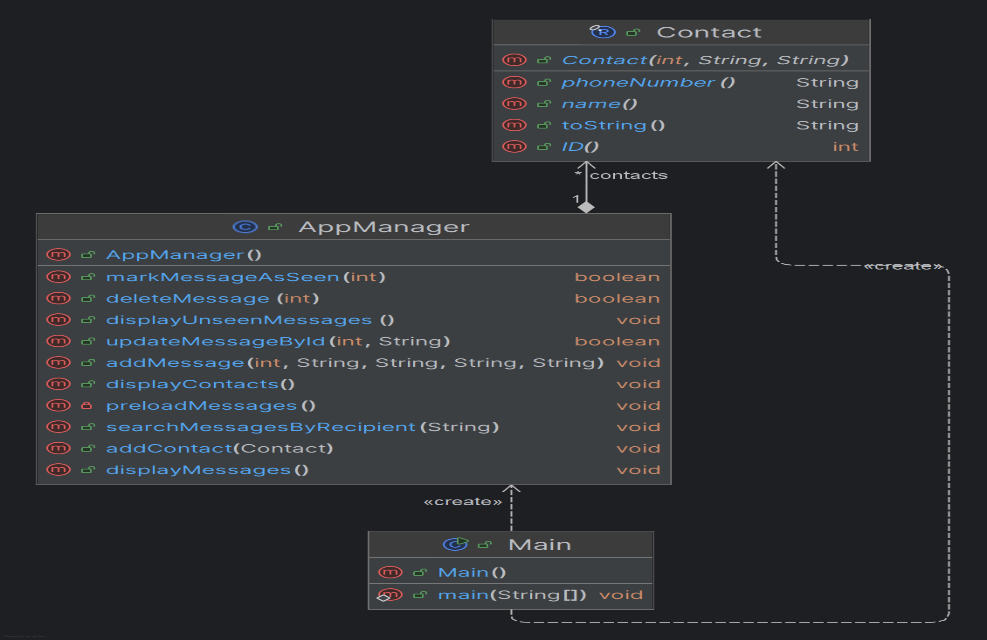
1. **Exit Program:**

****

### 2D Array Explanation:

In this program, the messages 2D array is used to store message data, where each row represents a single message, and each column holds different details about that message. The array is declared as String[MAX\_MESSAGES][6], meaning it can store up to 100 messages (rows), with 6 pieces of information (columns) for each message: message ID, sender, recipient, content, timestamp, and seen status. For example, messages[i][0] stores the message ID, while messages[i][5] stores whether the message has been seen ("true" or "false"). This 2D structure helps in organizing message data efficiently, allowing easy access to and management of messages.

**UML Diagram:**

****The UML (Unified Modeling Language) diagram helps visualize the structure of a software system by depicting the relationships between different components. For the provided code, the key classes are Main, AppManager, and Contact.