

**Name:** Hiba Noor

**Assignment No:** 1

**Subject:** OBJECT-ORIENTED PROGRAMMING

**Assignment: Console-Based Message Application**

**Submitted to:** Sir Shahid Bhatti

**CLASSES:** There are two classes along with main in my program.

1.Message class:

**Attributes:**

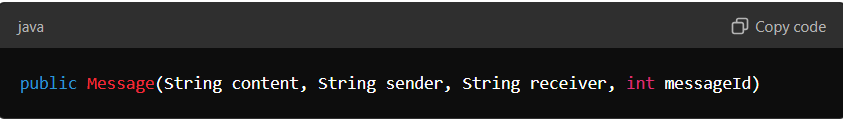
**Attributes:**

1. private String content: Stores the content of the message.
2. private String sender: Stores the name of the sender (default: "System").
3. private String receiver: Stores the name of the receiver.
4. private int messageId: Unique identifier for the message.
5. private LocalDateTime timestamp: Timestamp when the message was created.
6. private MessageStatus status: Enum that stores the status of the message (SEEN or UNSEEN)

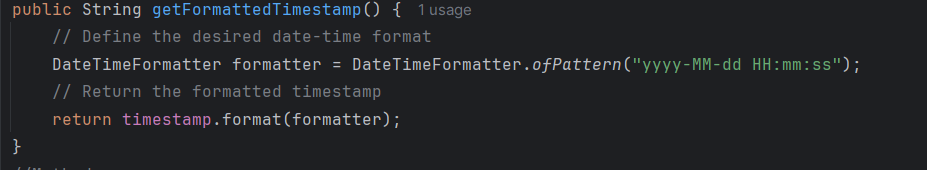
**Methods:**

1. **Constructor**:

* public Message(String content, String sender, String receiver, int messageId, LocalDateTime timestamp)

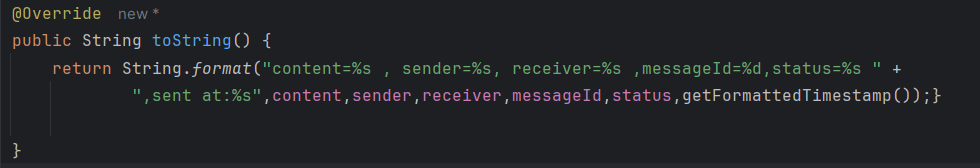


Getter Methods:

* public int getMessageId(): Returns the message ID.
* public String getContent(): Returns the content of the message.
* public String getSender(): Returns the sender of the message.
* public String getReceiver(): Returns the receiver of the message.
* public LocalDateTime getTimestamp(): Returns the timestamp of the message
* **getFormattedTimestamp()**:
  + Returns the formatted timestamp as a string.

1. **markAsSeen()**:A black background with white text

   Description automatically generated
2. **toString()**:
   * Returns a string representation of the message, including content, sender, receiver, messageId, status, and timestamp.



**2.Message App class:**

**Attributes:**

 private Message messages[][]: 2D array to store messages for each receiver.

 private int messageCounter[]: Array to track the number of messages for each receiver.

 private int maxMessagesPerReceiver: Maximum number of messages allowed per receiver.

 private int currentMessageId: Stores the ID to be assigned to the next message.

 private String receivers[]: Array to store the names of the receivers.

**Methods:**

1. **Constructor:**
   * public MessageApp(int receiverCount, int maxMessagesPerReceiver, String[] receivers)
2. **SEND MESSAGE:**

**sendMessage(String content, int receiverIndex)**:

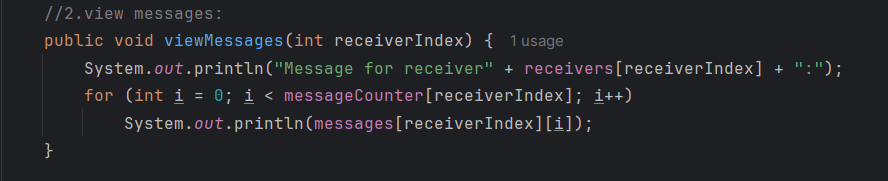
* + Sends a message to a specified receiver.



1. **VIEW MESSAGE:**

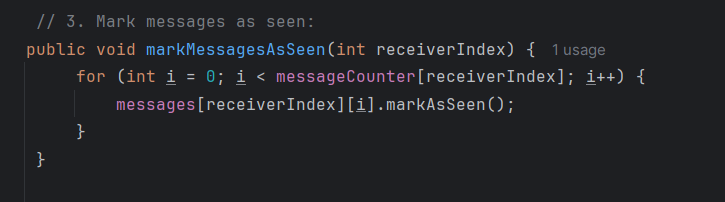
**viewMessages(int receiverIndex)**:

* + Displays all messages for a specified receiver.



1. **MARK MESSAGES AS SEEN:**

**markMessagesAsSeen(int receiverIndex)**:

* + Marks all messages as seen for a specified receiver.

1. **VIEW THE STATUS OF THE MESSAGE**:

**viewMessageStatus(int receiverIndex)**:

* + Displays the status of messages for a specified receiver.



6.**DELETE A MESSAGE:**

**deleteMessage(int receiverIndex, int messageId)**:

* + A computer screen shot of a program code

    Description automatically generatedDeletes a message by its ID for a specified receiver.

1. **SEARCH MESSAGES:**

**searchMessageById(int receiverIndex, int messageId)**:

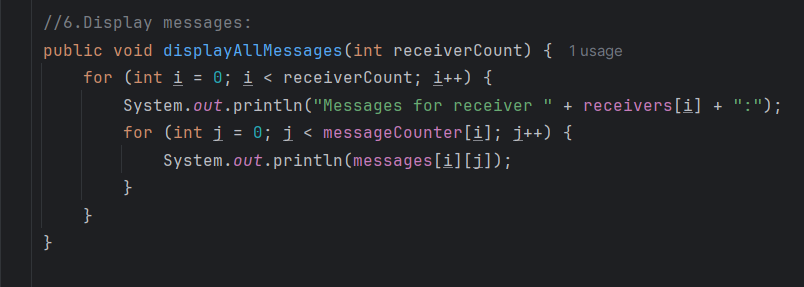
* + A computer screen shot of code

    Description automatically generatedSearches for a message by its ID for a specified receiver.

7.**DISPLAY MESSAGES:**

**displayAllMessages(int receiverCount)**:

* + Displays all messages for all receivers.



**8.SORT MESSAGES BY TIME:**

**sortBytime()**:

* + Sorts messages by timestamp for each receiver.

A computer screen shot of a program code

Description automatically generated

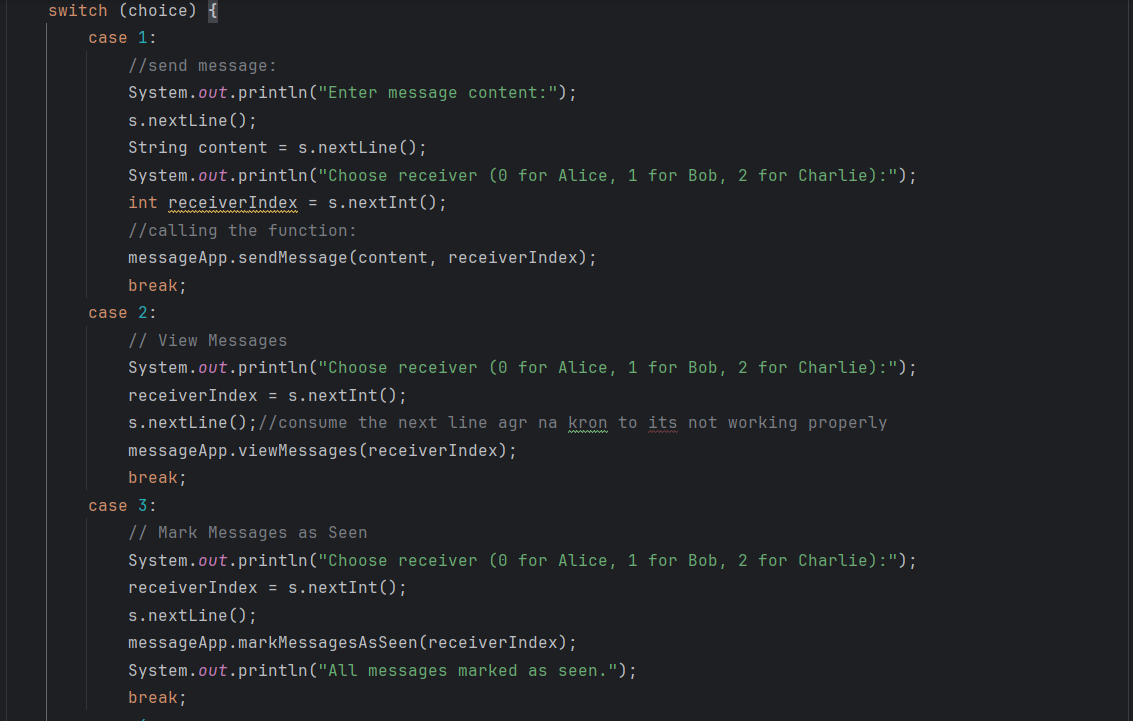
**3.MAIN CLASS:**

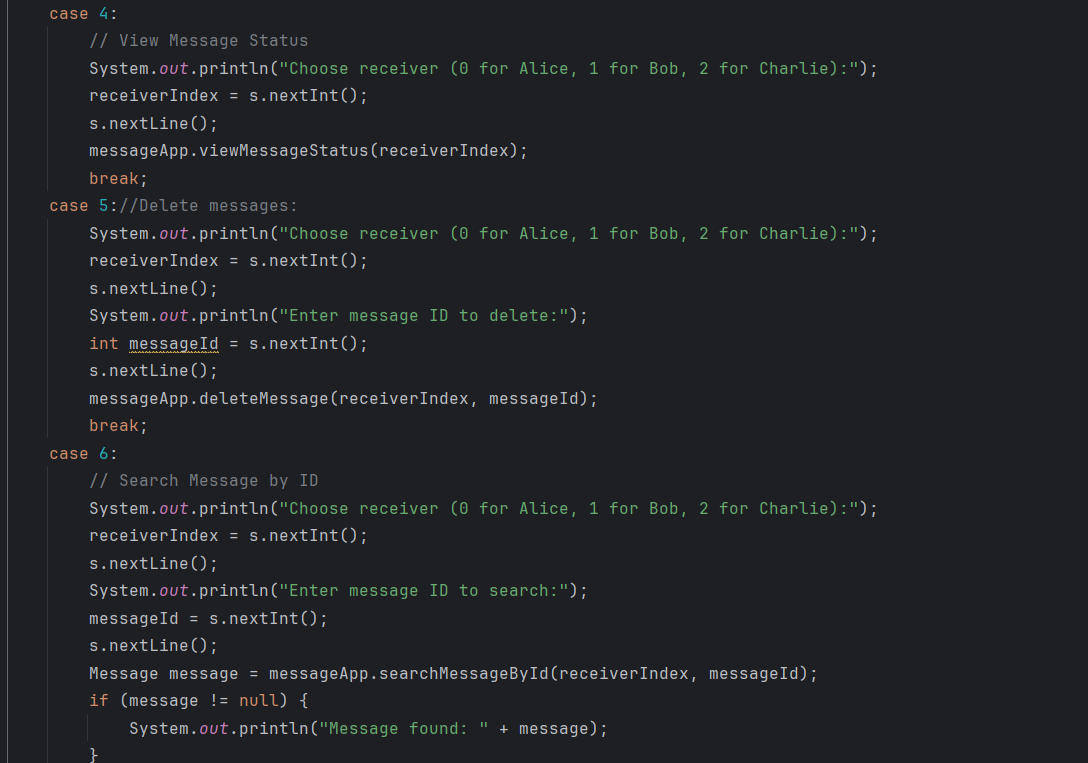
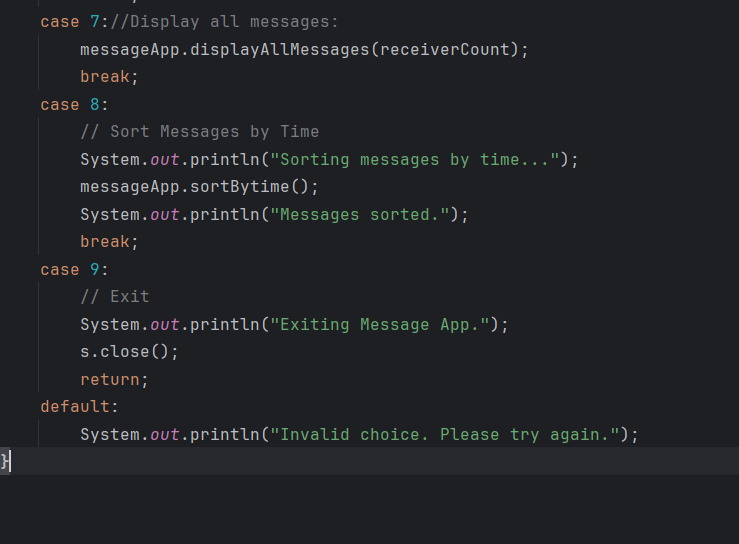
Menu Creation in main:

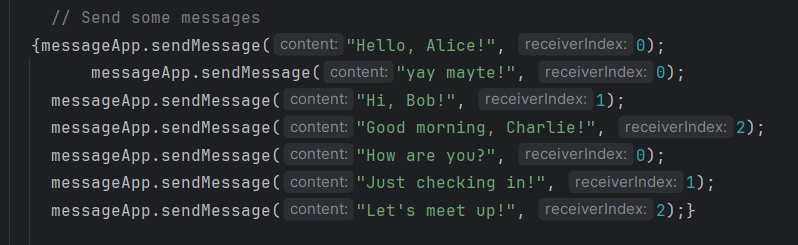
****

**For each function to work properly:**

We need to pass certain arguments :

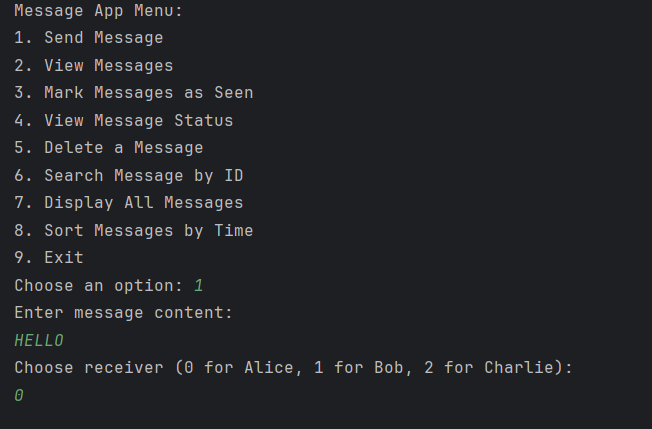
****



We also created a non-static block where we can actually put in some of messages so we can check the working of other functions properly:

The output of each function is given below:

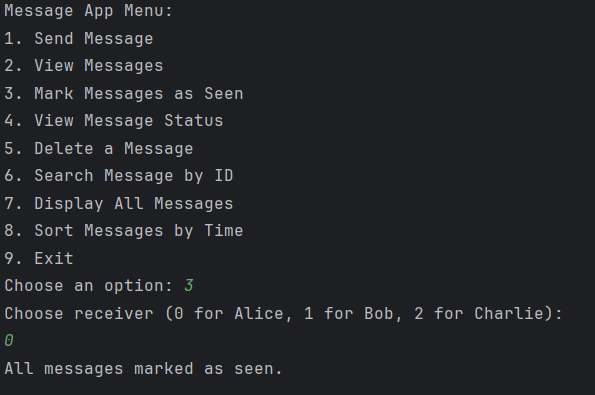
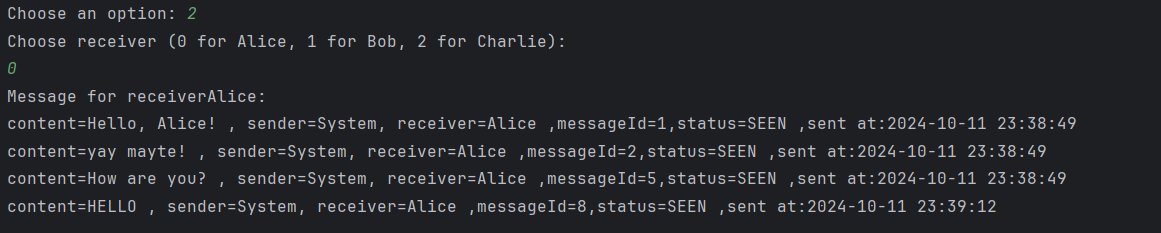
1.Send method adds the content of the message:



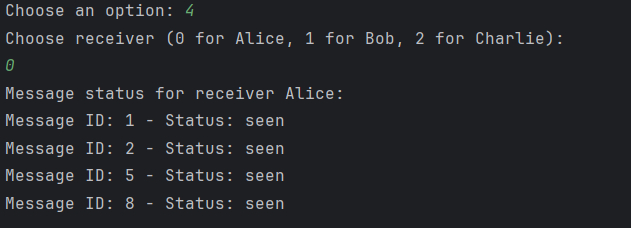
A computer screen shot of a computer screen

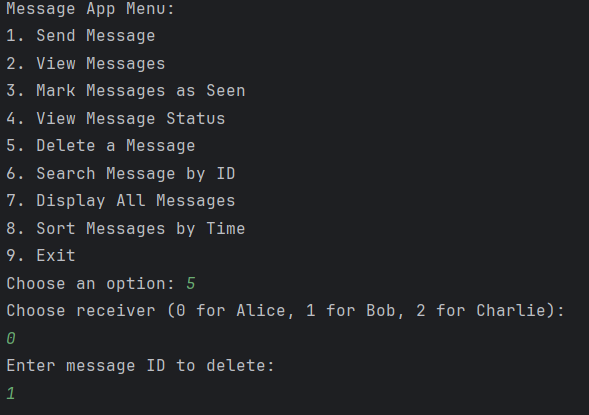
Description automatically generated2.View message shows us the messages added:

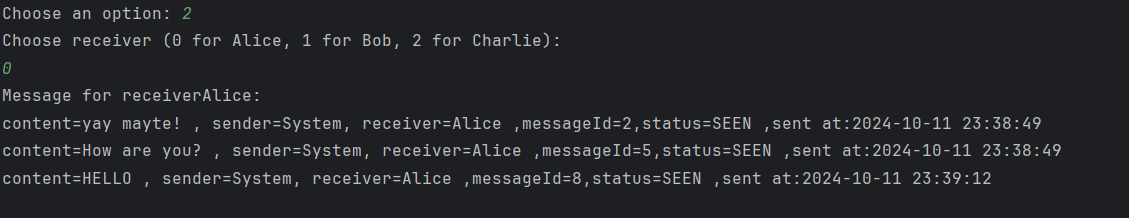
3.Mark messages mark the messages of any receiver as seen:



4.Shows the status of the message:



5.Deletes a message:

One message is deleted:

6.Searches the message by id:

**A computer screen shot of a black screen

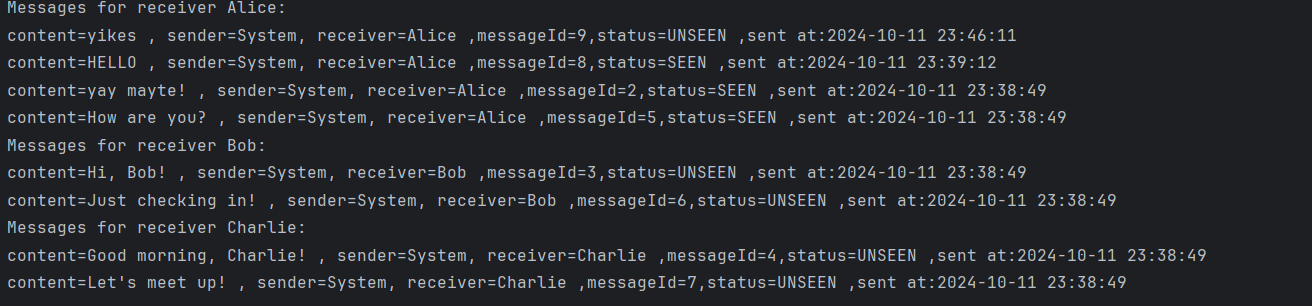
Description automatically generated**

**7.**Displays all the messages:

A computer screen shot of text

Description automatically generated

8.Sorts the message which is added new and then in display the most recent message appears to be the at the top:



UML DIAGRAM:

A screenshot of a computer

Description automatically generatedEXPLANATION OF UML DIAGRAM:

This UML diagram represents three main classes: Message, MessageApp, and Main. Here's an explanation of each component:

**1. Message Class:**

* **Attributes** (in red):
  + status: Type MessageStatus (probably an enum representing SEEN or UNSEEN).
  + sender, receiver: Strings representing who sent and received the message.
  + messageId: An integer unique to each message.
  + timestamp: Type LocalDateTime, stores when the message was created.
  + content: String representing the message's content.
* **Methods** (in red, prefixed with m):
  + **Getters**: Methods to retrieve attribute values (getSender(), getReceiver(), getContent(), getMessageId(), getStatus(), getTimestamp(), getFormattedTimestamp()).
  + markAsSeen(): Changes the status of the message to SEEN.
  + toString(): Likely converts the message details into a string format for easy display.

**2. MessageApp Class:**

* **Attributes**:
  + messageCounter: Array of integers, tracking the number of messages for each receiver.
  + receivers: Array of Strings, storing the names of the receivers.
  + messages: 2D array of Message objects. It stores messages for each receiver.
  + maxMessagesPerReceiver: Integer defining how many messages a receiver can have.
  + currentMessageId: Integer used to assign unique IDs to new messages.
* **Methods**:
  + viewMessageStatus(int): Views the status (SEEN/UNSEEN) of all messages for a receiver (based on index).
  + deleteMessage(int, int): Deletes a specific message (given the receiver and message ID).
  + searchMessageById(int, int): Searches for a message by its ID.
  + markMessagesAsSeen(int): Marks all messages for a given receiver as SEEN.
  + sortBytime(): Sorts messages by their timestamp (likely in descending order).
  + sendMessage(String, int): Sends a message with a given content and receiver index.
  + viewMessages(int): Views all messages for a particular receiver.
  + displayAllMessages(int): Displays all messages for all receivers.

**3. Main Class:**

* This contains a main method, which is the entry point of the application (void main(String[] args)), where instances of MessageApp and Message are likely created and managed.

**Relationships:**

* **Composition**:
  + The MessageApp class creates (create) instances of the Message class, showing that MessageApp manages multiple Message objects.
* **Messages Array**:
  + The MessageApp class has an array of Message objects (messages), indicating it manages messages for different receivers.

This UML gives a clear overview of how messages are created, stored, and managed for different receivers, with methods for various functionalities like sending, viewing, deleting, searching, and sorting messages.

4o