MESSAGING APP:   
This Program contains 4 Classes

* SMS.Java
* Receiver.Java
* Messenger.Java
* Main.Java

## The Structure and Logic:

1. **SMS Class:**

* SMS Class contains the attributes for a basic Message object such as: MessageId, timestamp, MessageContent etc and their getter/setters.
* This Class provides the fundamental Object of my program “Message” which will be manipulated by other classes.

1. Receiver Class:

* The Receiver Class contains array of SMS.
* The main idea of this class is that, each Receiver object will contain its own ARRAY OF SMS (message objects).
* So, we can have a separate chat history with each receiver and will have more free hand to create more functions to manipulate the receivers and their messages.

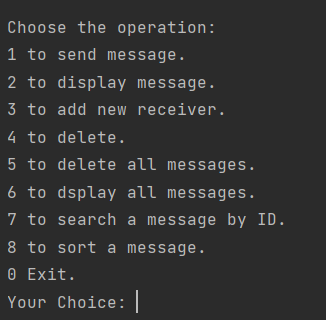
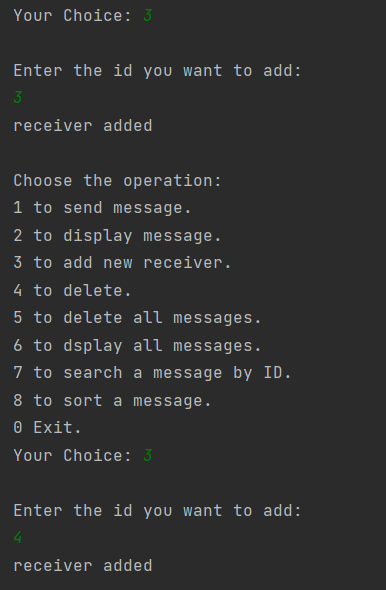
1. Messenger Class:

* This class contains the Array of Receivers, in other words we can say we have a whole contact list in this class.
* This class is responsible for sending messages to receiver via the function: SendMessage();  
  which takes first creates a message object by taking: messageContent and ReceiverID as input and, and then access the message array of that Receiver through its function addMessage(SMS), which takes the SMS object as input and add that to the sms array of that receiver.

1. Main Class:

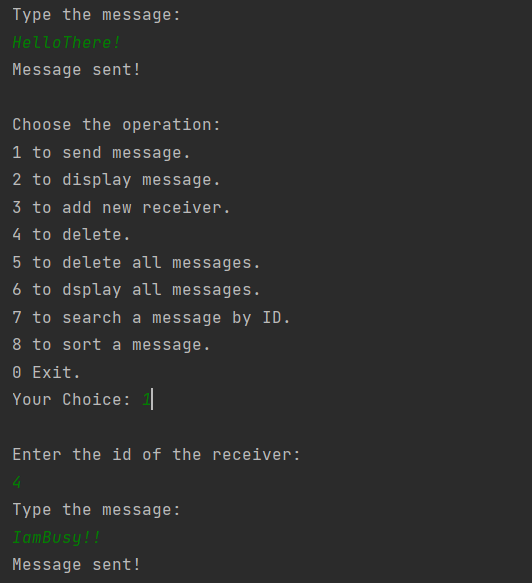
* Contains the main method which contains user interface to access the functions of Messenger Class.

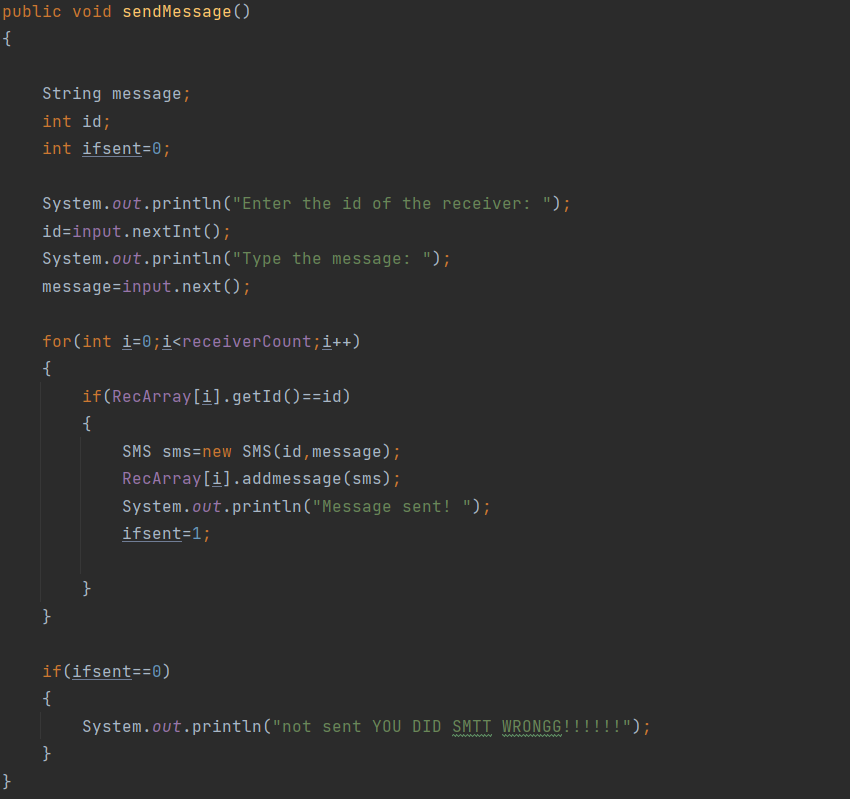
## Work Process With Screenshots:

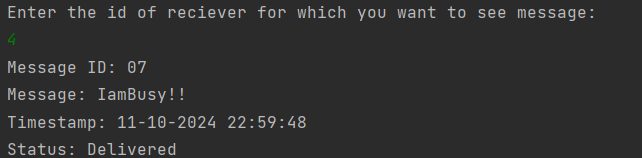
* Here is the the user interface:
* First Lets add new Receivers by choosing option 2:

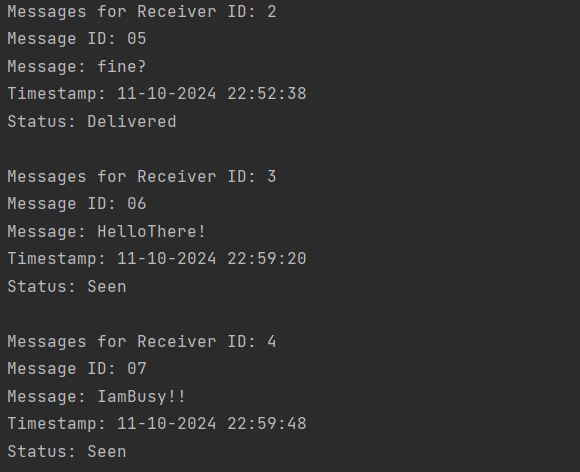
Added 2 new receivers with id 3 and 4.

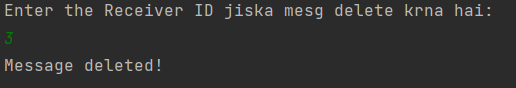
* Now lets send them message by choosing option 1:

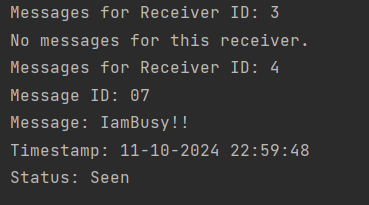
we have sent them these two messages.

And here is the actual code of send function:

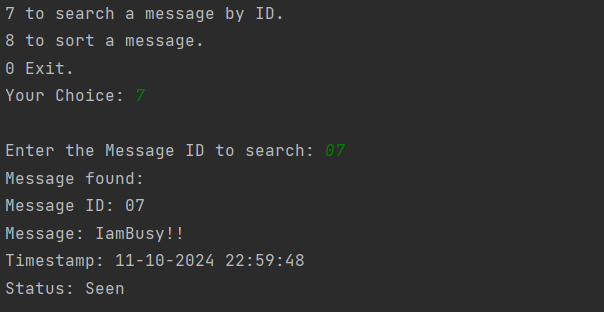
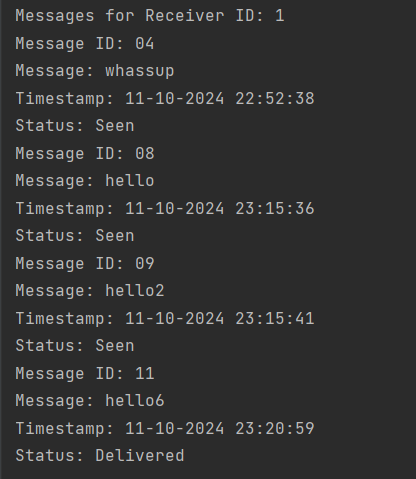
* After calling the display function for these 2 Ids, we will get our messages:

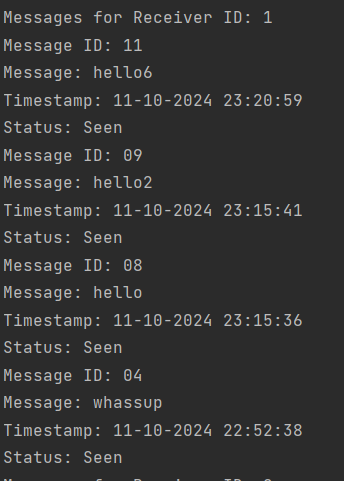
Once the message is delivered, if we call our display all messages function, or call display function again for that same id, that delivered status is updated to “Seen”:

* We can delete the messages too, lets say I want to delete the message of Receiver Id3:

After displaying all messages we get this:



* We can also search messages by Message ID:
* We can also sort message of receivers by their time stamp, so most recent message come on top:  
  Before Sort:  
  After Sort:



## UML diagram and EXPLANATION:

Explanation:

**1. Receiver Class**

* **Attributes:**
  + name: String: Name of the receiver.
  + sms: SMS[]: An array of SMS objects related to the receiver.
  + counter: int: Counter for the number of messages.
  + id: int: Unique identifier for the receiver.
  + messageCount: int: Keeps track of the number of messages.
* **Methods:**
  + displayMessage(): Displays messages for the receiver.
  + setMessageCount(int): Sets the message count for the receiver.
  + getMessageContent(): Retrieves the content of messages.
  + setName(String): Sets the name of the receiver.
  + smsOfReceiver(): Retrieves the SMS array.
  + getName(): Gets the name of the receiver.
  + getId(): Gets the receiver's ID.
  + addMessage(SMS): Adds a new SMS to the receiver's messages.
  + getMessageCount(): Retrieves the message count.
  + setId(int): Sets the receiver's ID.
  + addNewContact(): Adds a new contact for the receiver.

**2. SMS Class**

* **Attributes:**
  + RecID: int: ID of the SMS.
  + receiver: Receiver: The receiver associated with the SMS.
  + timestamp: LocalDateTime: Timestamp of when the SMS was received or sent.
  + counter: int: Tracks the order or number of messages.
  + id: String: Unique ID for the message.
  + status: String: The status of the message (e.g., sent, received).
  + messageContent: String: The content of the SMS.
* **Methods:**
  + setMessageContent(String): Sets the content of the SMS.
  + getMessageContent(): Retrieves the SMS content.
  + setTimestamp(LocalDateTime): Sets the timestamp of the SMS.
  + getId(): Retrieves the SMS ID.
  + getTimestamp(): Retrieves the timestamp of the SMS.
  + setId(String): Sets the SMS ID.
  + getStatus(): Retrieves the status of the SMS.
  + setStatus(String): Sets the SMS status.
  + getFormattedTimestamp(): Gets the formatted timestamp.
  + toString(): Converts the SMS object to a string.

**3. Messenger Class**

* **Attributes:**
  + input: Scanner: A scanner object to take input (e.g., user input).
  + receiver: Receiver: The receiver object being interacted with.
  + maxReceiver: int: The maximum number of receivers allowed.
  + RecArray: Receiver[]: An array of Receiver objects.
  + messageCounter: int: Keeps count of total messages sent/received.
  + receiverCount: int: Keeps count of total receivers.
* **Methods:**
  + displayMessage(): Displays messages.
  + delete(): Deletes a specific message.
  + deleteAllMessages(): Deletes all messages.
  + receiveMessage(): Receives a message and adds it to the system.
  + addReceivers(): Adds a new receiver to the system.
  + searchMessage(): Searches for a message.
  + sendMessage(): Sends a message.
  + displayAllMessages(): Displays all messages for all receivers.
  + sortMessages(): Sorts messages (possibly by timestamp).