

**ASSIGNMENT NO 1**

**OBJECT OREINTED PROGRAMMING**

**Title:**

**Messaging App**

**Submitted By:**

**Hareem Mohal**

**Reg No:**

**SP24-BSE-133**

**Submitted to:**

**Sir Shahid Bhatti**

**Date:**

**12th Oct,2024**

**WeChat Java Application - Explanation**

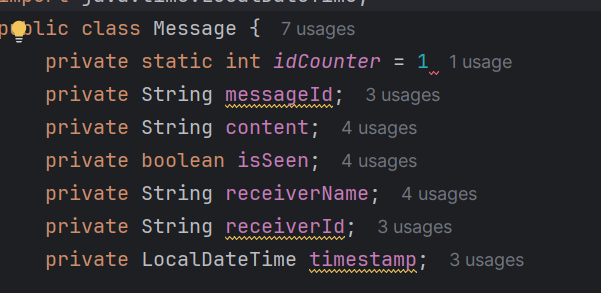
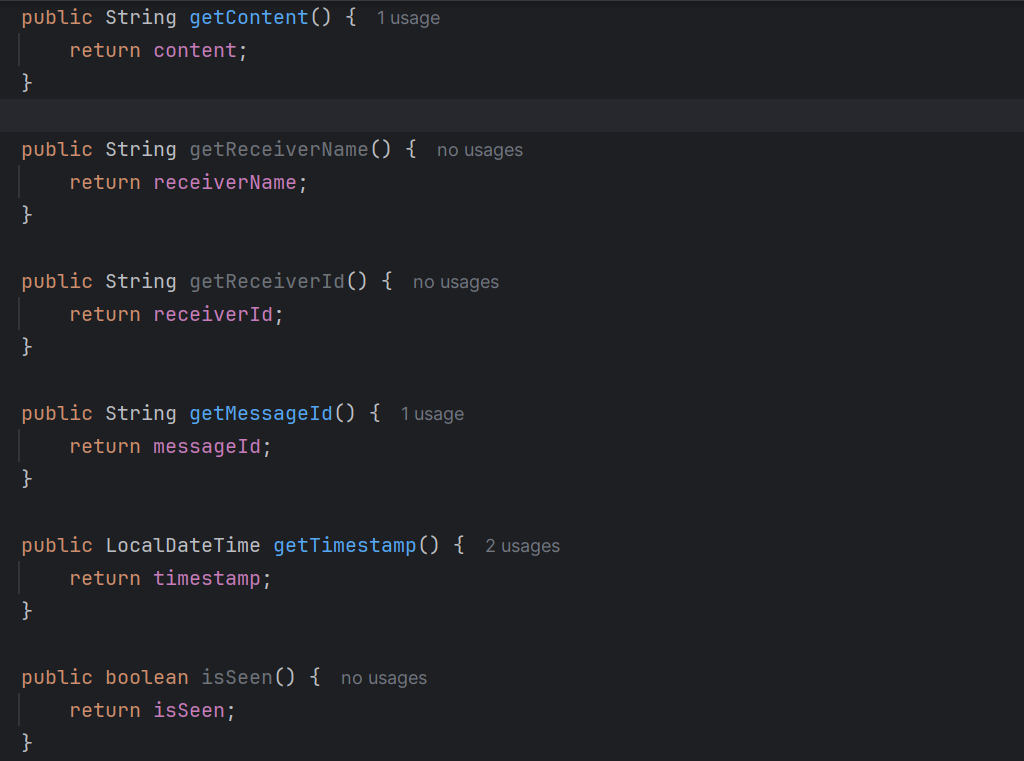
**1. Overview**

The WeChat application is a simple messaging system written in Java. It allows users to create receivers, send messages, display messages, and view a list of all registered receivers. The application uses Object-Oriented Programming (OOP) principles such as encapsulation, composition, and basic array handling to manage messages and receivers.

**2. Classes and Their Attributes**

**Class: Message**

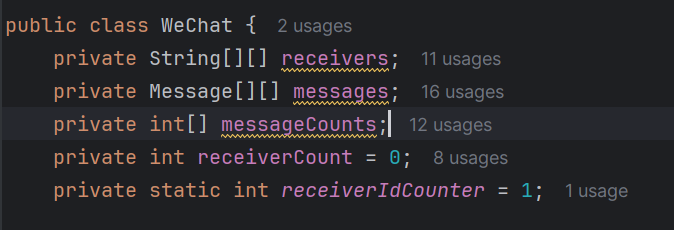
The **Message** class represents a single message in the WeChat application.

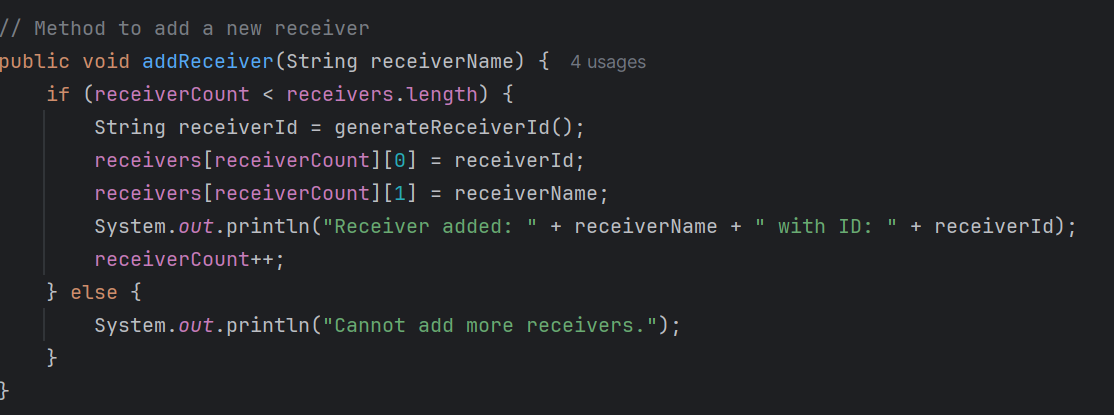
* **Attributes:**
  + messageId (String): A unique identifier for each message which is autogenerated.
  + content (String): The text content of the message.
  + isSeen (boolean): Indicates whether the message has been seen by the receiver.
  + receiverName (String): The name of the receiver of the message.
  + receiverId (String): The unique ID of the receiver.
  + timestamp (LocalDateTime): The date and time when the message was created.
* **Methods:**
  + getContent(): Returns the content of the message.
  + getReceiverName(): Returns the name of the receiver.
  + getReceiverId(): Returns the ID of the receiver.
  + getMessageId(): Returns the unique message ID.
  + getTimestamp(): Returns the timestamp of the message.
  + isSeen(): Returns true if the message has been seen, otherwise false.
  + markAsSeen(): Marks the message as seen.
  + toString(): Returns a string representation of the message, showing its details including status (Seen or Delivered).

**Class: WeChat**

The WeChat class represents the core functionality of the messaging app.

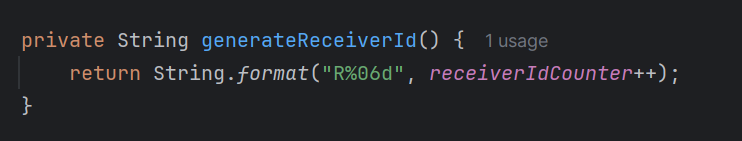
* **Attributes:**



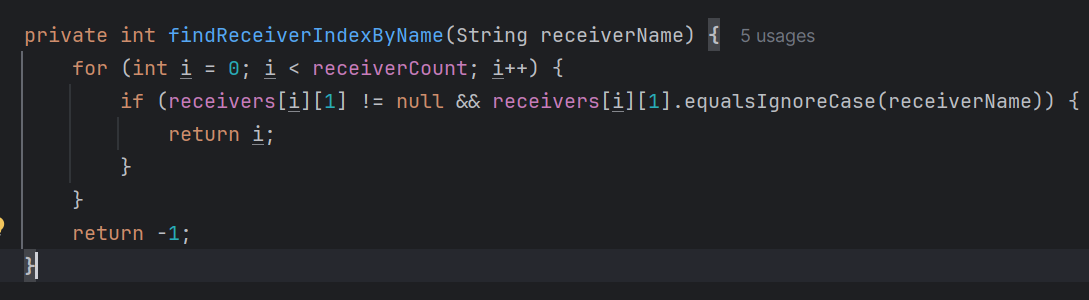
* + receivers (String [][]): A 2D array to store receiver IDs and names.
  + messages (Message [][]): A 2D array to store messages for each receiver.
  + messageCounts (int []): An array to keep track of the number of messages for each receiver.
  + receiverCount (int): Shows number of registered receivers.
  + receiverIdCounter (static int): A counter used to generate unique IDs for receivers.
* **Methods:**
* **Purpose**: Adds a new receiver to the system with an auto-generated unique ID.
* **Logic**: Checks if there is space for a new receiver, generates a unique ID using generateReceiverId(), and stores the name and ID in the receivers array.
* **Example**:
* **GenerateReceiverId():**

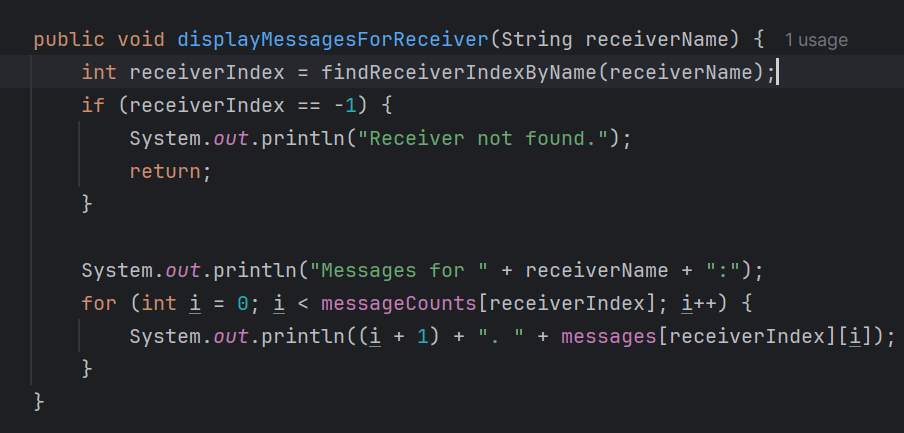
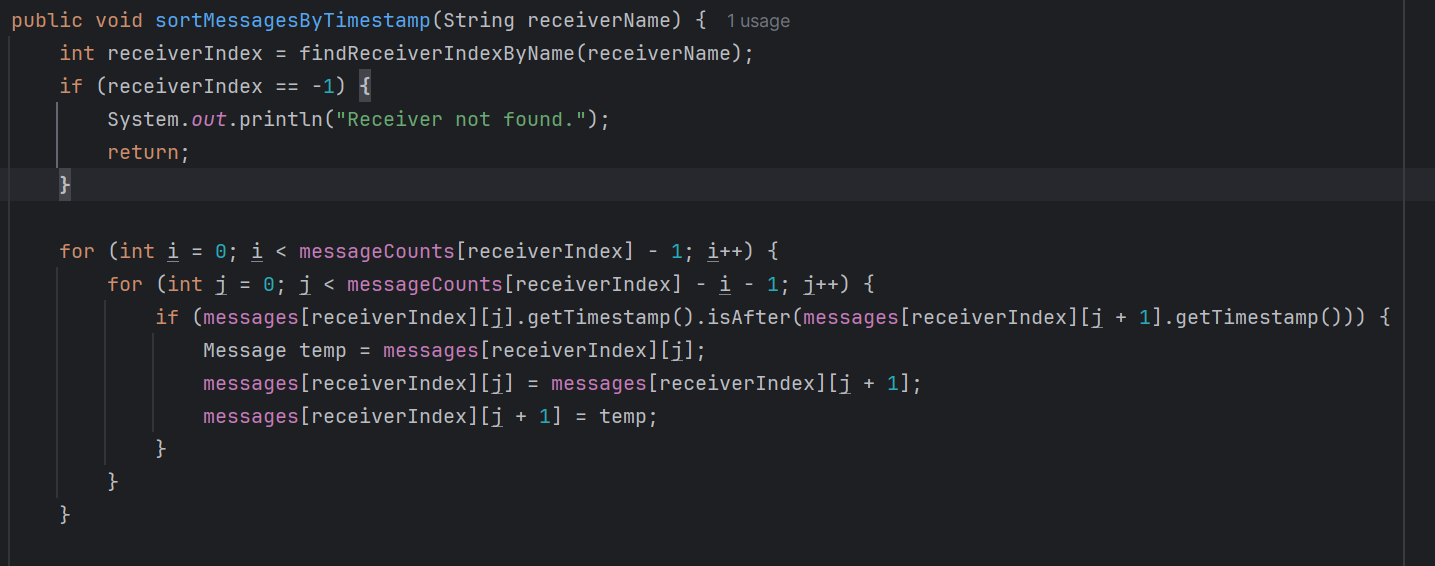
**Purpose**: Generates a unique 6-digit ID for each receiver.

**Logic**: Uses receiverIdCounter to generate IDs in the format R000001, R000002, etc.



* **FindReceiverIndexByName(String receiverName):** **Purpose**: Finds the index of a receiver using their name.
* **Logic**: Iterates through the receivers array and checks if the name matches (case-insensitive).
* **Returns**: Index of the receiver if found, -1 if not found.

****

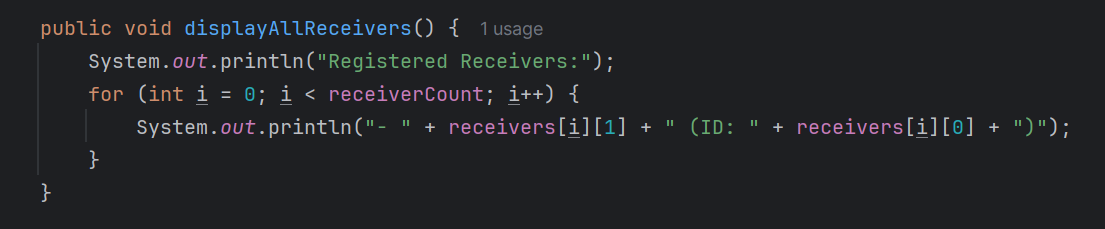
* **SendMessage(String messageContent, String receiverName):**
* **Purpose**: Sends a message to a specified receiver.
* **Logic**: Finds the receiver index, creates a Message object with the content and receiver's details, and stores it in the messages array.
* **displayMessagesForReceiver(String receiverName):** **Purpose**: Displays all messages for a specific receiver.
* **Logic**: Finds the receiver index and iterates through their messages, displaying each one.
* **sortMessagesByTimestamp(String receiverName):**
* **Purpose**: Sorts messages for a specified receiver based on their timestamps.
* **Logic**: Uses Bubble Sort to arrange messages chronologically by comparing timestamps.
* **deleteMessage(String receiverName, int messageNumber)**
* **Purpose**: Deletes a specific message for a receiver.
* **Logic**: Finds the receiver, shifts the messages to fill the gap left by the deleted message, and decrements the message count.



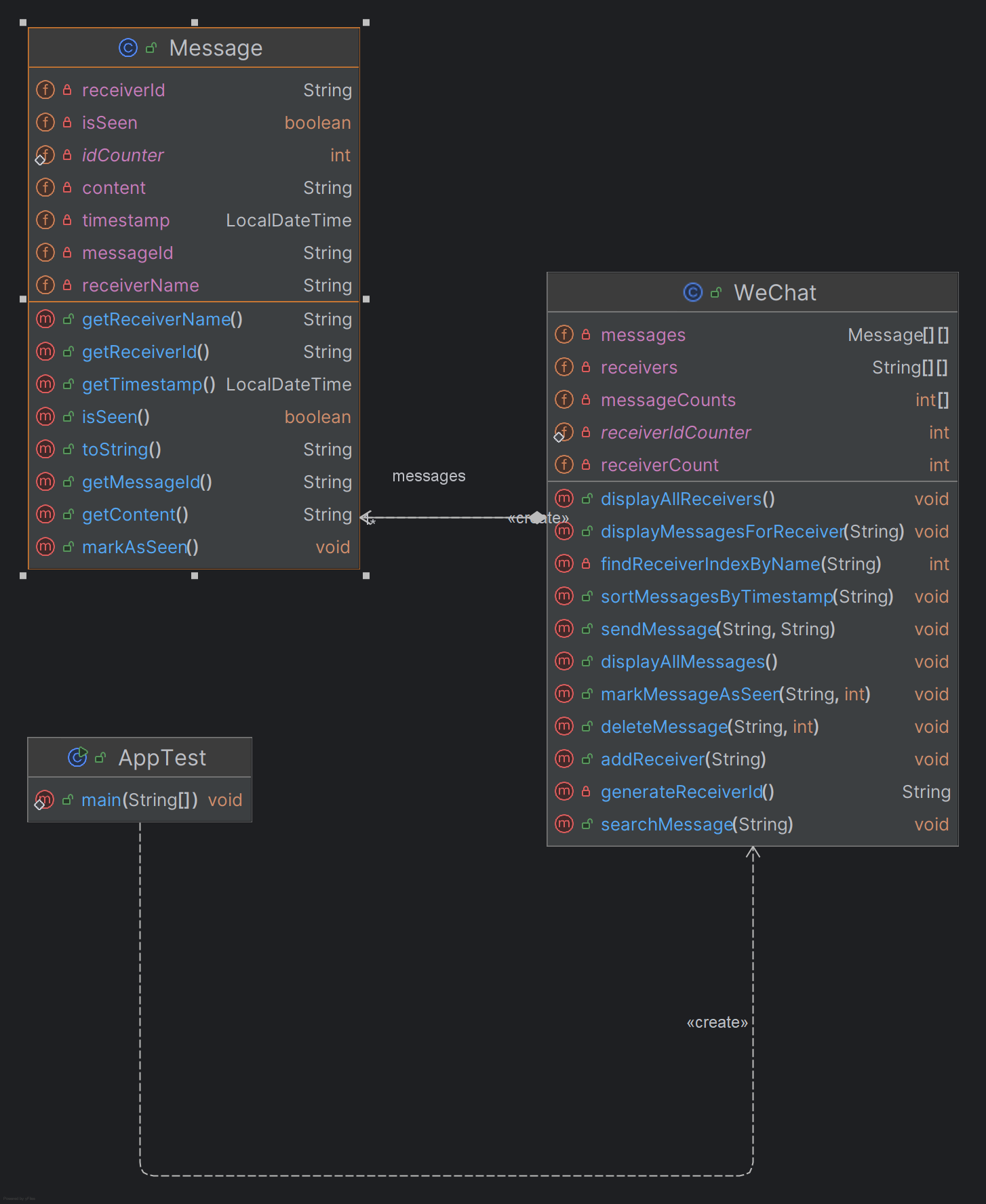
**displayAllReceivers():**

 **Purpose**: Displays all registered receivers with their IDs.

 **Logic**: Iterates through the receivers array and prints each receiver's name and ID.



**UML DIAGRAM**

****

**Explaination:**

### ****Key Insights****

* The WeChat class uses arrays (receivers, messages, messageCounts) to manage receivers and messages, suggesting that the number of receivers and messages per receiver is limited by predefined array sizes.
* WeChat handles sorting and searching of messages using custom methods, which provides flexibility but might be less efficient compared to using collections like lists or maps.
* Message class encapsulates the message content and metadata (such as timestamp and isSeen status), promoting a separation of concerns and allowing WeChat to focus on message management rather than message details.