

|  |  |
| --- | --- |
| Reg. # | SP24-BSE-116 |
| Name | Subhan Khalid |
| Course Title | OOP In Java |
| Section | B |
| Assignment # | 1 |
| Submitted to | Dr. M Shahid Bhatti |
| Date | 11/10/2024 |

**Explanation of Java Messaging Application**

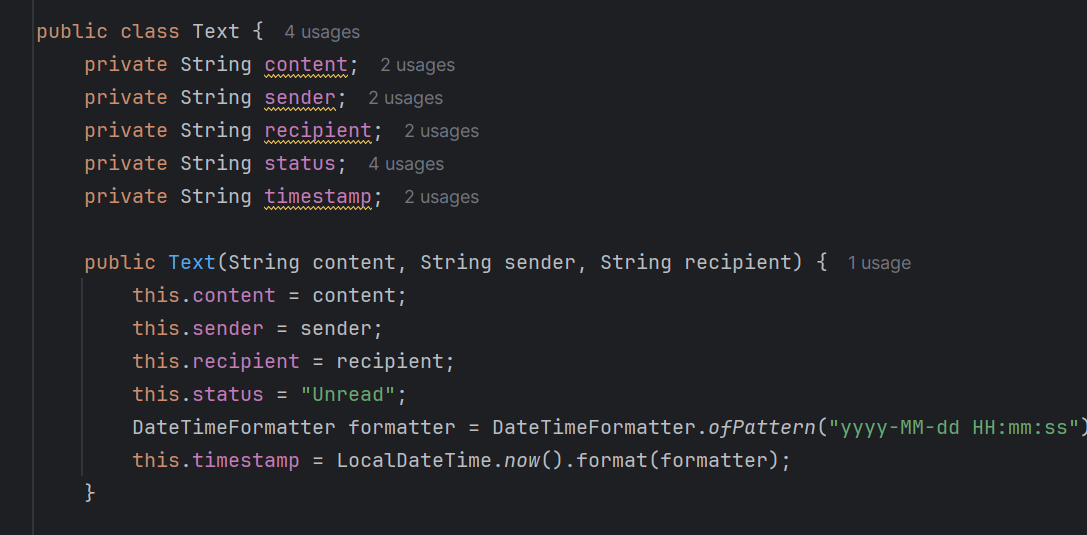
**Table of Contents**

1. Introduction
2. Code Structure
   * 2.1. Class: Text
   * 2.2. Class: MessagingApp
   * 2.3. Class: Main
3. Functionality Overview
4. User Interaction

**1. Introduction**

This document provides an overview of a Java-based messaging application that allows users to manage contacts and send messages. The application is built using Object-Oriented Programming (OOP) principles, featuring three main classes: Text, MessagingApp, and Main.

**2. Code Structure**

**2.1. Class: Text**

The Text class represents a message sent from one user to another. It contains the following attributes:

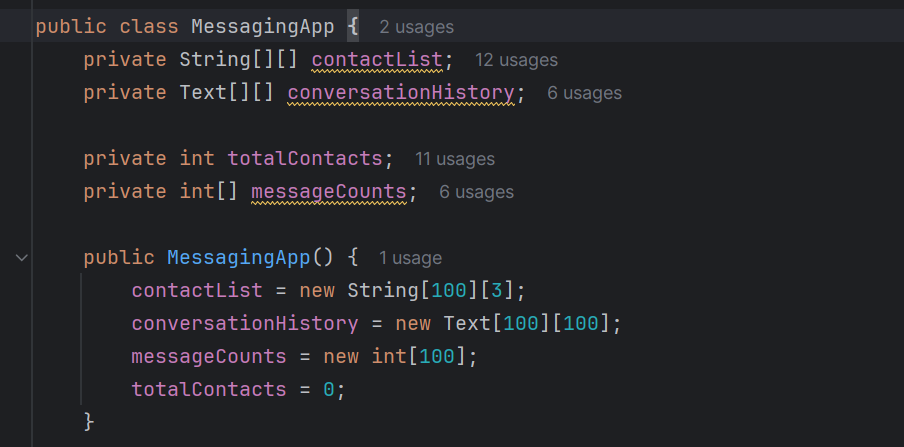
* message: The content of the message.
* sender: The name of the person sending the message.
* receiver: The name of the person receiving the message.
* status: Indicates whether the message has been read or is unread.
* time: The timestamp when the message was created.

**Key Methods:**

* **Constructor**: Initializes the message attributes and sets the current time.
* **markAsRead()**: Changes the message status to "Read."
* **getDetails()**: Returns a formatted string with message details.

**2.2. Class: MessagingApp**

The MessagingApp class manages the contact list and conversation history. It contains the following attributes:



* contactList: A 2D array storing contact information (ID, name, phone number).
* conversationHistory: A 2D array storing message objects (of type Text) for each contact.
* totalContacts: Keeps track of the number of contacts added.
* messageCounts: An array to count the number of messages exchanged with each contact.

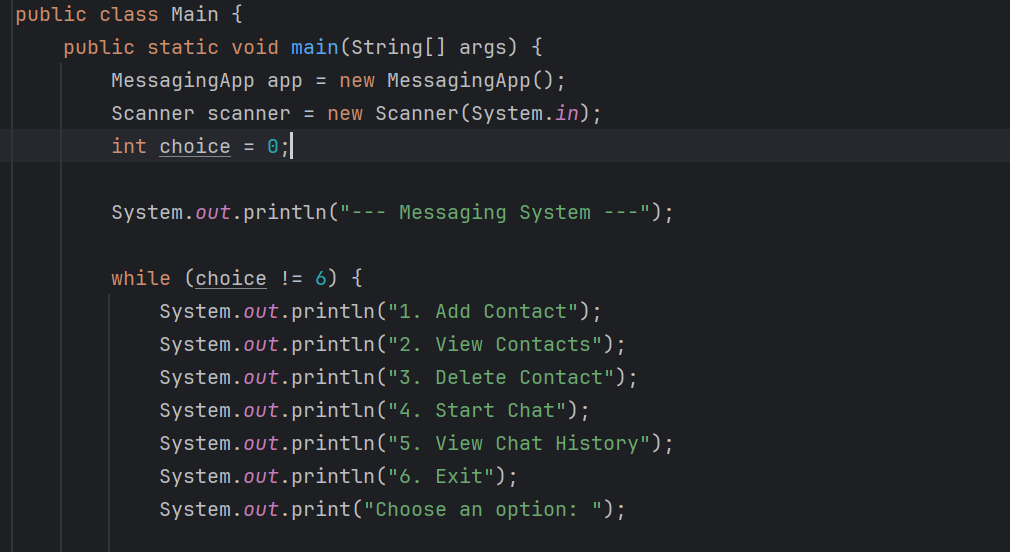
**Key Methods:**

* **addNewContact(String name, String phoneNumber)**: Adds a new contact to the contact list.
* **displayContacts()**: Displays all contacts in the contact list.
* **removeContact(String name)**: Removes a contact by name.
* **deleteMessage(String name, int index):** Deletes a message from the chat history with a specified contact based on the message index.
* **initiateChat(String name, String senderName)**: Starts a chat session with a specified contact.
* **displayChatHistory(String name)**: Displays the chat history with a specified contact.

**2.3. Class: Main**

The Main class contains the main method that serves as the entry point for the application. It manages user interaction through a menu system.

**Key Features:**



* Displays a menu of options for the user to manage contacts and chat.
* Uses a while loop to allow users to repeatedly select options until they choose to exit.
* Implements a switch-case structure to execute the appropriate method based on user input.

**3. Functionality Overview**

The application allows users to perform the following actions:

* Add new contacts with names and phone numbers.
* View the list of contacts.
* Delete contacts from the list.
* Delete the last message
* Initiate a chat session with a contact and send messages.
* View the chat history with a specific contact.

**4. User Interaction**

When the application runs, users are presented with a menu. They can select an option by entering the corresponding number. Depending on the user's choice, they will be prompted to provide additional information (like contact name or message) or be presented with the relevant output (like the list of contacts or chat history).

The application continues to run until the user selects the option to exit.