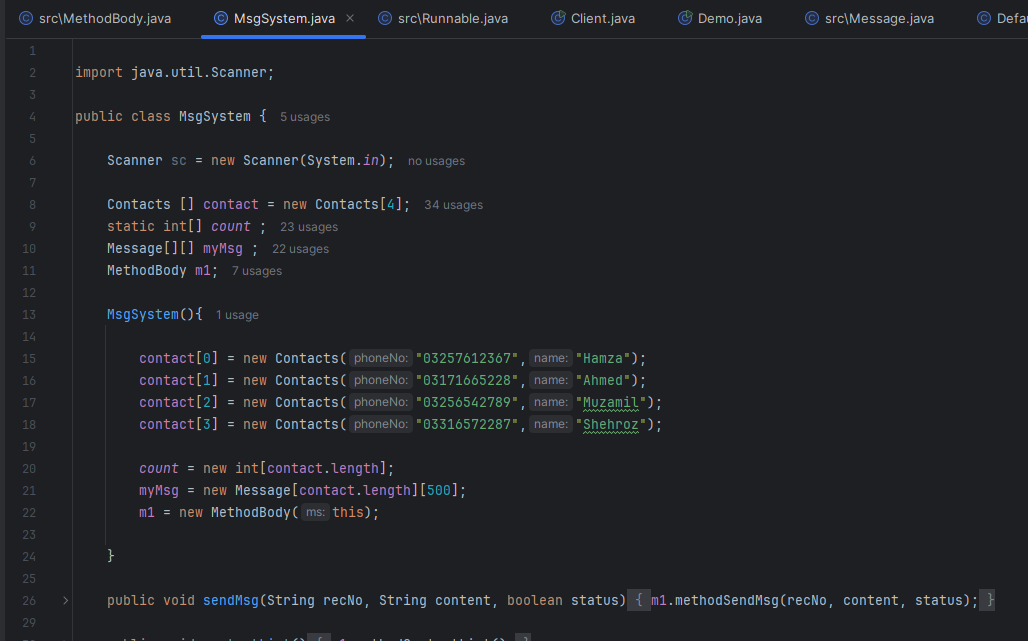
**Java Messaging System Code**

**Introduction**

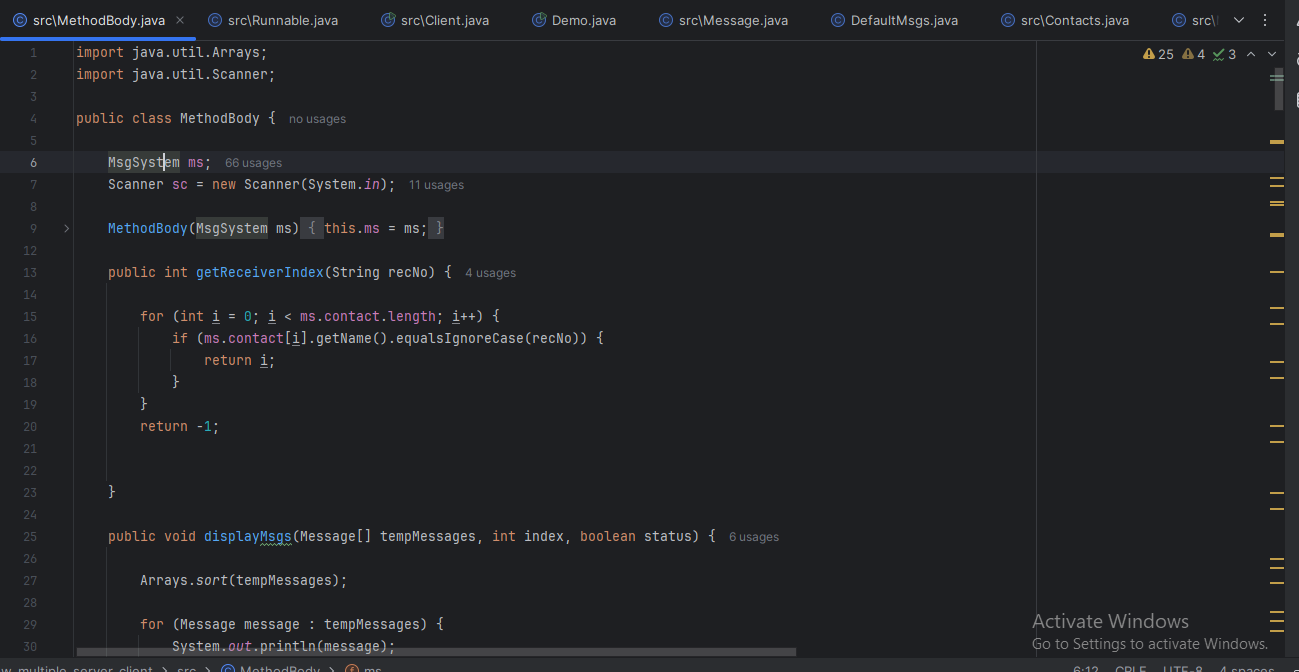
This Java program is designed to simulate a basic messaging system, allowing users to manage contacts, send messages, and view message histories. It also includes functionality for running a server-client communication via sockets. This assignment breaks down each main class and explains how it contributes to the overall messaging system.

**Overview of Classes and Their Functions**

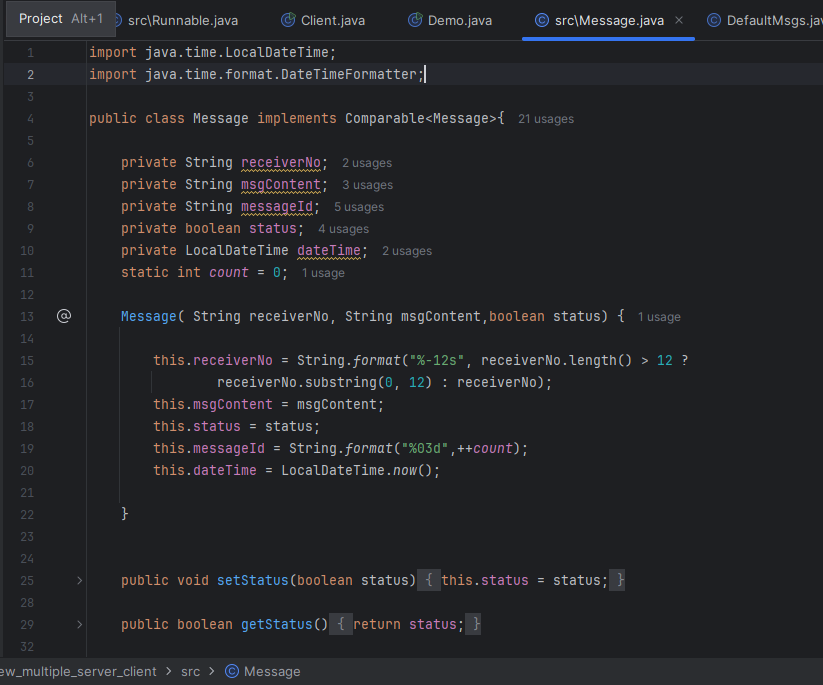
1. **MsgSystem**:
   * This is the main class that initializes the messaging system with a set of contacts and manages the operations related to contacts and messages.
   * It creates an array of contacts, stores messages for each contact, and counts the messages sent to each.
   * **Key Methods**:
     + sendMsg: Sends a message to a specified contact.
     + contactList: Displays the list of contacts.
     + receiverMsgs: Shows all messages received by a particular contact.
     + statusHistory: Shows the status of messages (read/unread).
     + addContact: Adds a new contact to the list.
     + deleteContact: Removes a contact from the list.



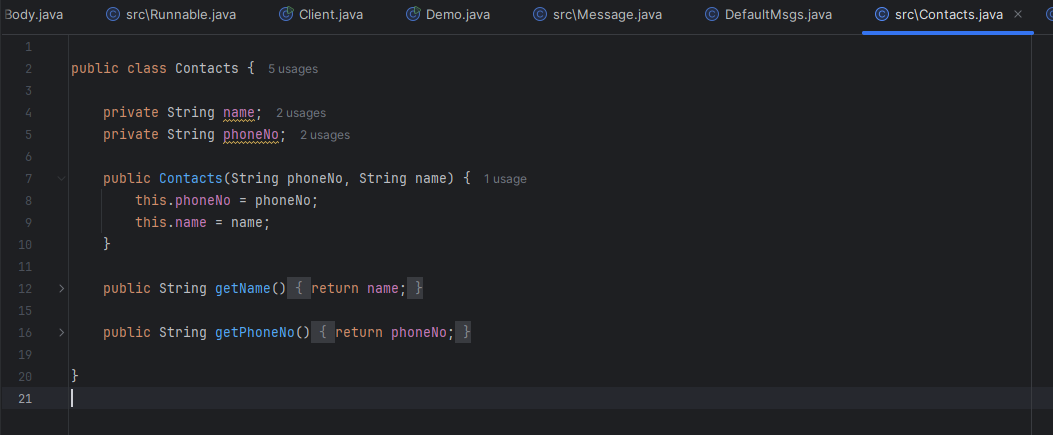
1. **MethodBody**:
   * This class is responsible for implementing the core functionalities of the messaging system. It contains methods that interact with MsgSystem to handle different tasks.
   * **Key Methods**:
     + getReceiverIndex: Finds the index of a contact based on the receiver’s name.
     + displayMsgs: Sorts and displays messages, updating their status if needed.
     + methodSendMsg: Sends a message to a specific contact.
     + methodContactList: Displays all contacts.
     + methodReciverMsgs: Manages viewing options for received messages.
     + methodStatusHistory: Displays messages based on their read/unread status.
     + methodAddContact: Adds a new contact.
     + methodDeleteContact: Deletes an existing contact.



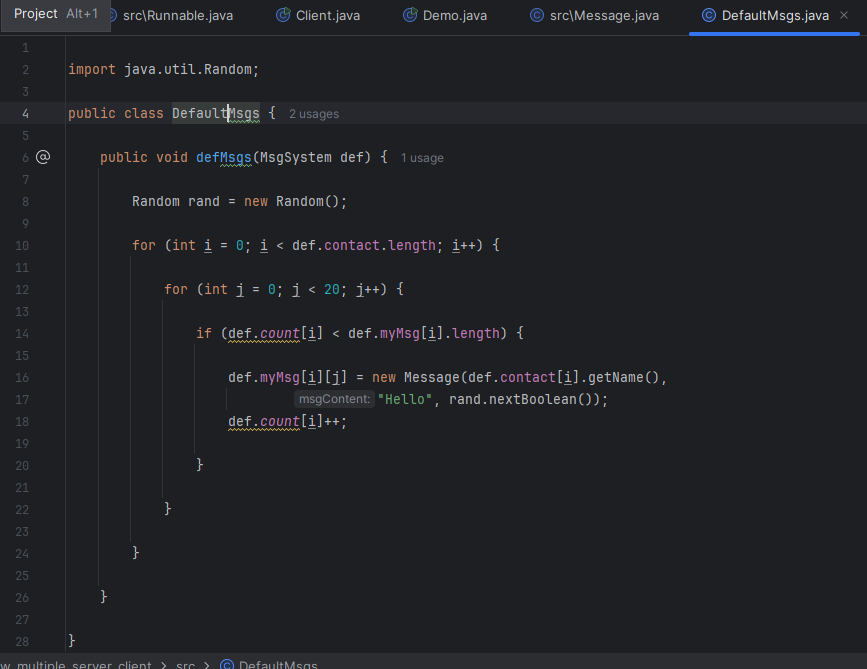
1. **Message**:
   * Represents a message object with attributes like receiverNo, msgContent, status, and dateTime.
   * **Key Methods**:
     + setStatus: Updates the message status to read/unread.
     + compareTo: Compares messages based on their ID for sorting.
     + toString: Converts message details into a readable format.



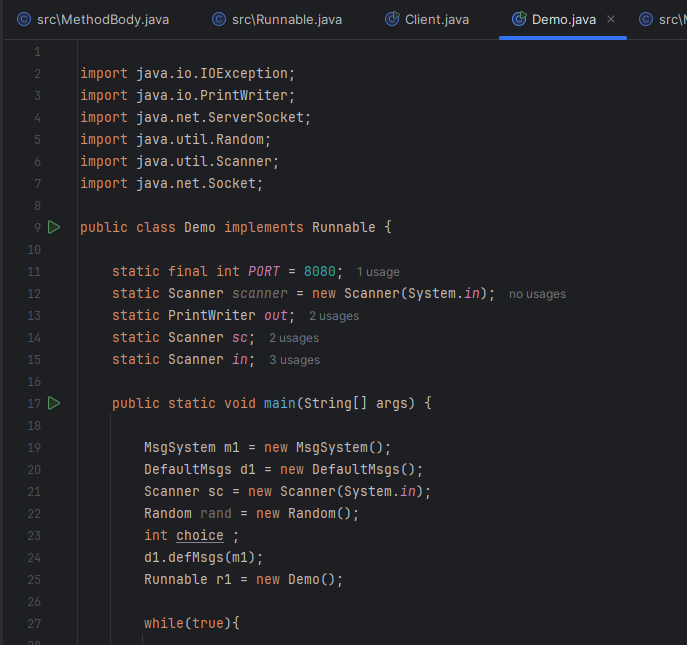
1. **Contacts**:
   * Represents a contact with a name and phone number.
   * Simple class with getter methods for accessing contact details.



1. **DefaultMsgs**:
   * Fills each contact with a set of default messages when the system starts.



1. **Demo (implements Runnable)**:
   * Contains the main method that allows users to interact with the messaging system through a menu.
   * Additionally, this class sets up server-client communication using sockets.



**How the Program Works**

1. **Initialization**:
   * The program starts by initializing MsgSystem and DefaultMsgs, which sets up predefined contacts and fills them with sample messages.
2. **Menu Interaction**:
   * The user interacts with the messaging system through a menu interface, where they can view contacts, send messages, view message history, and manage contacts.
3. **Server-Client Communication**:
   * The Demo class establishes a server-client communication using ServerSocket for basic message exchange, allowing a client to connect, send messages, and disconnect by typing “bye”.

**Summary**

This messaging system demonstrates the following:

* Basic object-oriented programming concepts in Java.
* Management of contacts and messages.
* Handling of arrays and two-dimensional arrays for storing contacts and messages.
* Use of interfaces, method overriding, and socket programming for client-server communication.