**Summary**  
This Java-based console chat application is built using Object-Oriented Programming (OOP) principles, with the core functionalities handled by separate classes for managing contacts, messages, and the overall application flow. Each feature in the application corresponds to specific methods in the respective classes.

### About the Classes

**Contact Class**:

* 1. **Purpose**: This class represents a contact in the chat application. It stores the contact's name and phone number.
  2. **Fields**:
     1. String name: The contact’s name.
     2. String phone: The contact’s phone number.
  3. **Methods**:
     1. Contact(String name, String phone): A constructor to initialize a contact object with a name and phone number.
     2. getName(): Returns the contact's name.
     3. getPhone(): Returns the contact's phone number.
     4. toString(): A method to provide a formatted string representation of the contact, including name and phone number.

**Message Class**:

* 1. **Purpose**: This class represents a message exchanged between contacts. Each message is associated with a timestamp.
  2. **Fields**:
     1. String messageContent: The content of the message.
     2. Date timestamp: The time when the message was created.
  3. **Methods**:
     1. Message(String messageContent): Constructor to initialize a message with content and automatically set the timestamp to the current time.
     2. getMessageContent(): Returns the message content.
     3. getTimestamp(): Returns the timestamp of the message.
     4. toString(): A method that returns a formatted string, showing the timestamp and the message content.

**ChatApp Class**:

* 1. **Purpose**: This class contains the logic for the core functionalities of the chat application, such as adding contacts, sending messages, and viewing/deleting messages.
  2. **Fields**:
     1. Map<String, Contact> contacts: A HashMap storing contacts, with the phone number as the key.
     2. Map<String, List<Message>> messageStore: A HashMap that stores message history for each contact, using the contact's phone number as the key.
  3. **Methods**:
     1. addContact(String name, String phone): Adds a new contact to the contacts map and initializes an empty message list for that contact in messageStore.
     2. removeContact(String phone): Removes a contact and their messages from the system based on their phone number.
     3. viewContacts(): Displays the list of all contacts in the system.
     4. sendMessage(String phone, String messageContent): Sends a message to a contact based on the phone number and stores it in messageStore.
     5. viewMessages(String phone): Displays the message history with a specific contact.
     6. deleteMessages(String phone): Deletes all messages exchanged with a specific contact.

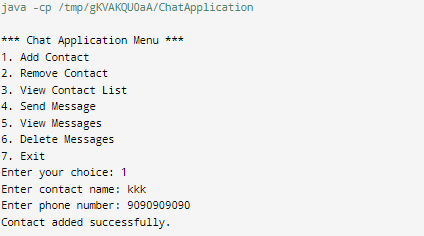
**ChatApplication Class** (Main Class):

* 1. **Purpose**: The entry point of the program, where the user interacts with the system through a menu in the console.
  2. **Methods**:
     1. main(String[] args): Contains the main loop of the program, where users choose options from the menu and interact with the application. This method calls the corresponding methods in the ChatApp class to handle user inputs.

### **Features Overview**

#### 1. ****Add Contact****

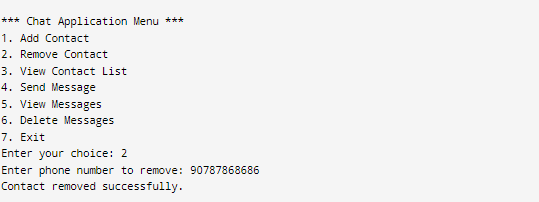
* **Description**: Adds a new contact to the chat application with a name and phone number.
* **Relevant Class**: ChatApp
* **Method Used**:
  + addContact(String name, String phone):
    - This method is responsible for creating a new Contact object and storing it in the contacts map.
    - Additionally, it initializes an empty message list in messageStore for this contact.
* **Contact Class Methods Used**:
  + Contact(String name, String phone): This constructor is used to create a new contact object.

**Screenshot**:  


* + Show a prompt for adding contact name and phone number, followed by a confirmation message.

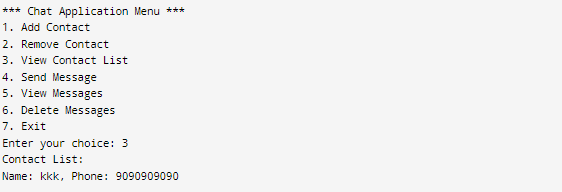
#### 2. ****Remove Contact****

* **Description**: Removes a contact from the application based on their phone number, and deletes their messages.
* **Relevant Class**: ChatApp
* **Method Used**:
  + removeContact(String phone):
    - This method checks if the phone number exists in the contacts map.
    - If found, it removes the contact from both the contacts and messageStore maps.
    - If the contact is not found, an error message is displayed.

**Screenshot**:  


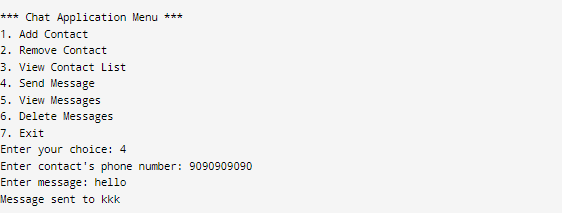
#### ****View Contact List****

* **Description**: Displays all the contacts stored in the system.
* **Relevant Class**: ChatApp
* **Method Used**:
  + viewContacts():
    - This method checks if the contacts map is empty. If it contains contacts, it loops through the map and prints each contact using the toString() method from the Contact class.
    - If no contacts are found, it displays a message saying the contact list is empty.
* **Contact Class Methods Used**:
  + toString(): This method is called to display the contact’s details (name and phone number).

**Screenshot**:  


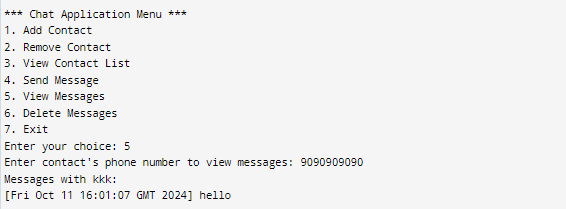
#### 4. ****Send Message****

* **Description**: Sends a message to a specific contact by their phone number.
* **Relevant Class**: ChatApp
* **Method Used**:
  + sendMessage(String phone, String messageContent):
    - This method first checks if the contact exists using the phone number.
    - It then creates a new Message object using the message content and stores it in the messageStore for that contact.
    - A confirmation message is displayed, showing that the message was successfully sent.
* **Message Class Methods Used**:
  + Message(String messageContent): The constructor initializes the message content and timestamp.

**Screenshot**:  


#### 5. ****View Messages****

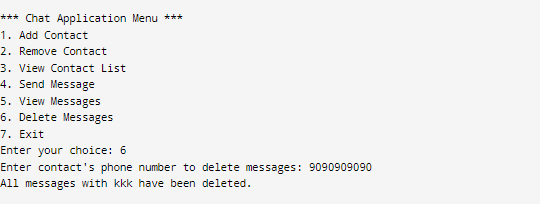
* **Description**: Displays all messages exchanged with a specific contact.
* **Relevant Class**: ChatApp
* **Method Used**:
  + viewMessages(String phone):
    - This method checks if the contact exists using the phone number.
    - It retrieves the message list from messageStore and loops through each message, displaying it using the toString() method from the Message class.
    - If no messages are found, an appropriate message is displayed.
* **Message Class Methods Used**:
  + toString(): Displays the message content along with its timestamp.

**Screenshot**:  


* + Show a list of messages exchanged with the contact or a message stating that there are no messages.

#### 6. ****Delete Messages****

* **Description**: Deletes all messages exchanged with a specific contact.
* **Relevant Class**: ChatApp
* **Method Used**:
  + deleteMessages(String phone):
    - This method checks if the contact exists using the phone number.
    - It then clears the message list for that contact in the messageStore, effectively deleting all messages.
    - A confirmation message is displayed, notifying the user that all messages were deleted.

**Screenshot**:  


#### 7. ****Exit****

* **Description**: Exits the application.
* **Relevant Class**: ChatApplication
* **Method Used**:
  + Exiting is handled within the main loop in the ChatApplication class when the user selects the exit option.

**Screenshot**:

