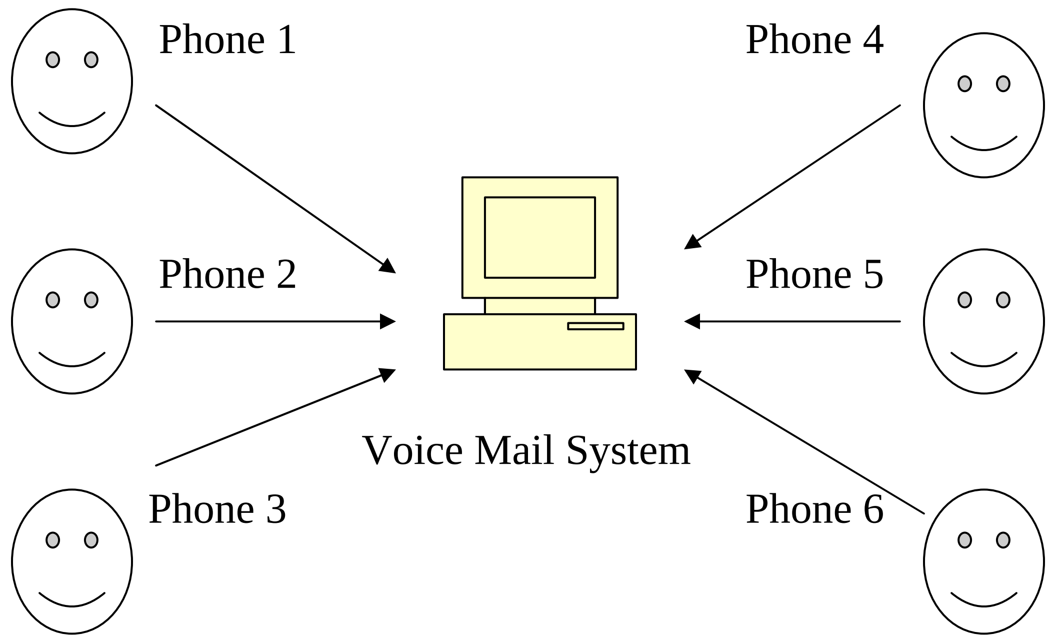
Case Study: Simulation of A Voice Mail System

In a voice mail system, a person dials an extension number and, provided the other party does not pick up the telephone, leaves a message.

The other party can later retrieve the messages, keep them, or delete them.



Design and implement a program that simulates a voice mail system.

The first formal step is the analysis phase. We will define the behavior through a set of use cases.

Use case 1: Leave a Message

1. The caller dials the main number of the voice mail system.
2. The voice mail system speaks a prompt. Enter mailbox number followed by #.
3. The caller types in the extension number of the message recipient.
4. The voice mail system speaks. You have reached mailbox xxxx. Please leave a message now.
5. The caller speaks the message.
6. The caller hangs up.
7. The voice mail system places the recorded message in the recipient's mailbox.

Use case 2: Retrieve Messages

1.The mailbox owner types the passcode, followed by the # key.
2.The voice mail system plays the mailbox menu:
    Enter 1 to retrieve your messages.
    Enter 2 to change your passcode.
    Enter 3 to change your greeting.
3.The mailbox owner selects the "retrieve your messages" menu option.
4. The voice mail system plays the message menu:
    Enter 1 to listen to the current message.

Enter 2 to save the current message.
Enter 3 to delete the current message.
Enter 4 to return to the mailbox menu.

5. The mailbox owner selects the "listen to the current message" menu option.
6. The voice mail system plays the current new message, or, if there are no new messages, the current old message.
7. The voice mail system plays the message menu.
8. The mailbox owner selects "delete the current message". The message is permanently removed.
9. Continue with Step 4.

Use case 3: Change the greeting

1.The mailbox owner selects the “Change your greeting” menu option
2. The mailbox owner speaks the greeting.
3. The mailbox owner presses the # key.
4. The mail system sets the new greeting.

Use case 4: Change the Passcode

1.The mailbox owner selects the “Change your greeting” menu option
2. The mailbox owner dials the new passcode.
3. The mailbox owner press the # key.
4. The mail system set the new passcode.

Discovering Classes

Mailbox:

(1) Keep messages.
(2) Keep track of which messages are new and which are saved.
(3) Mailbox owner should be able to retrieve, save, and delete their messages.

Message and Message Queue:

Message Queue is an appropriate data structure for messages.

Two queues, one for the new messages and one for the saved messages.

MailSystem:

Mailboxes are kept in **Mai1System.**

Responsibility is managing the mailboxes.

Telephone:

(1) Take user input (button presses, voice input, and hangup actions)
(2) Play voice output on the speaker.

Connection:

- In a real voice mail system, it is possible for multiple telephones to be connected to the voice mail system.

- Each connection needs to keep track of the current state (recording, retrieving message and so on)

- It is possible that one connection is currently recording a message while another is retrieving messages.

- A connection communicates with a telephone, carries out the user commands, and keeps track of the state of the session.

The Relationships among Classes:

Leave a Message.

1.The user dials an extension.

2.The **Telephone** sends the dialed extension number to the **connection.**

3.The **Connection** asks the **Mai1System** to find the **Mai1box** object with the given extension number.

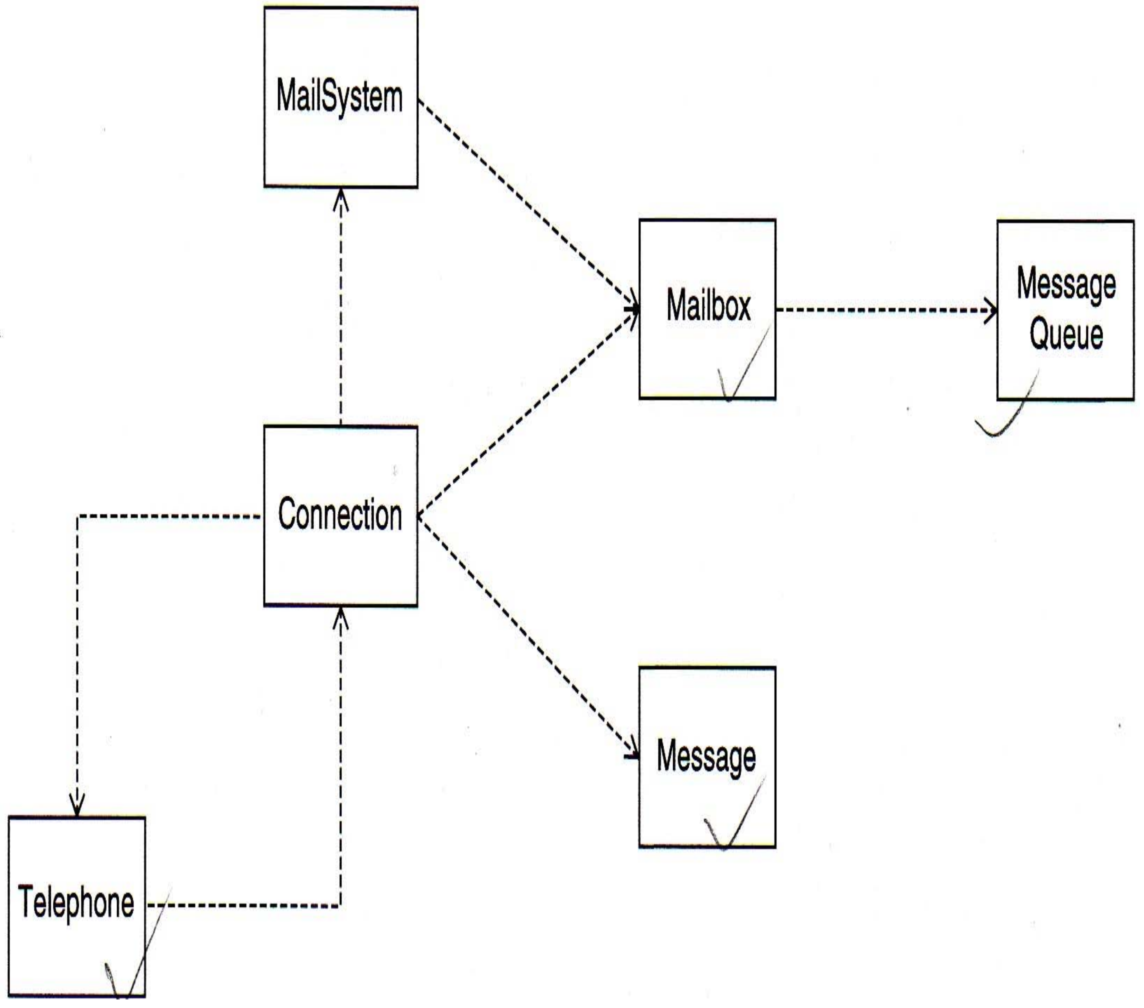
4.The **Connection** asks the **Mai1box** for its greeting.
5.The **connection** asks the **Telephone** to play the greeting on the speaker.
6.The user speaks the message. The **Telephone** asks the **Connection** to record it.
7.The user hangs up. The **Telephone** notifies the **Connection.**
8.The **connection** constructs a **Message** object that contains the recorded message.
9. The **connection** adds the **message** object to the **Mailbox.**

Retrieve Messages

1. The mailbox owner types in the passcode. The **Telephone** notifies the **Connection.**
2. The **connection** asks the **Mai1box** to check the passcode.
3. Assuming the passcode was correct, the **Connection** sets the **Mai1box** as the current mailbox and asks the **Telephone** to speak the mailbox menu.
4. The mailbox owner types in the "retrieve messages" menu option. The **Telephone** passes it on to the **Connection**.
5. The **connection** asks the **Telephone** to speak the message menu.
6. The mailbox owner types in the "listen to current message" option. The **Telephone** passes it on to the **Connection**.
7. The **Connection** gets the first message from the current **Mai1box** and sends its contents to the **Telephone**.
8.The **connection** asks the **Telephone** to speak the message menu.

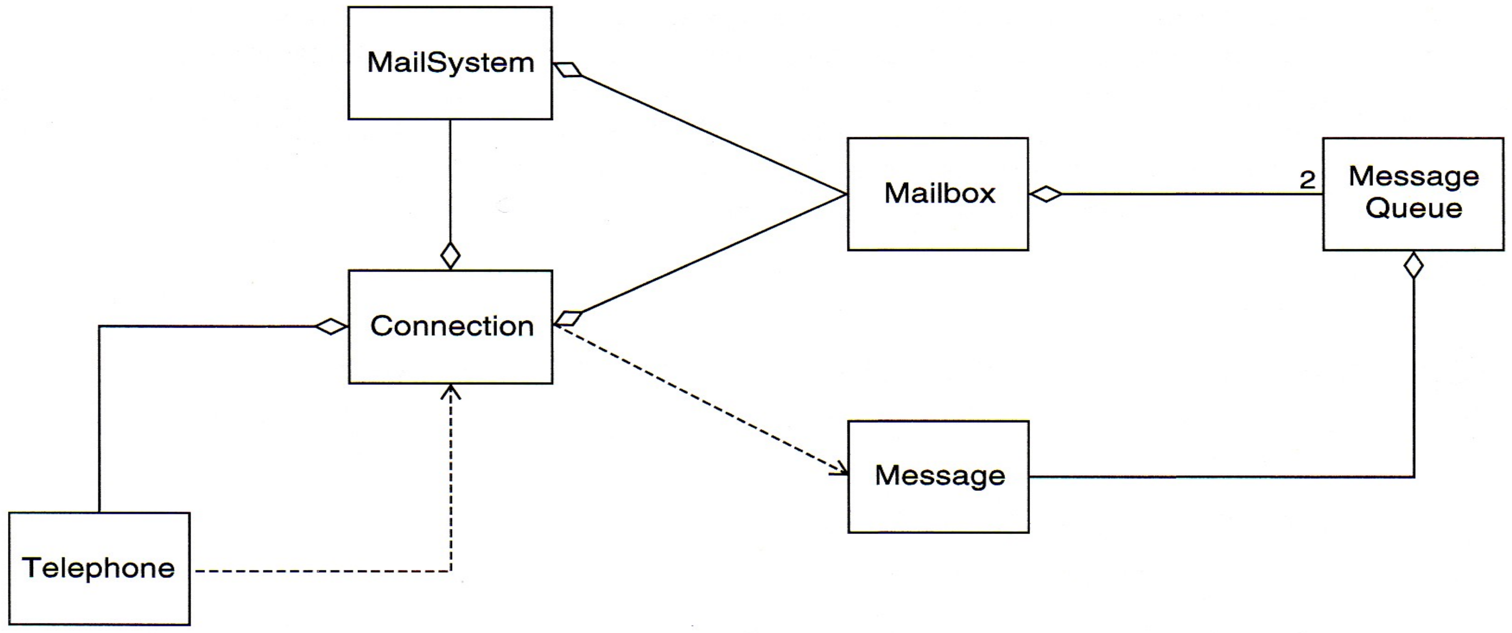
9. The mailbox owner types in the "save current message" menu option. The **Telephone** passes it on to the **Connection**.
10.The **Connection** tells the **Mailbox** to save the current message.
11.The **Connection** asks the **Telephone** to speak the message menu.

UML Class Diagrams for the Voice Mail System



Consider the aggregation relationships. From the previous discussion, we know the following:

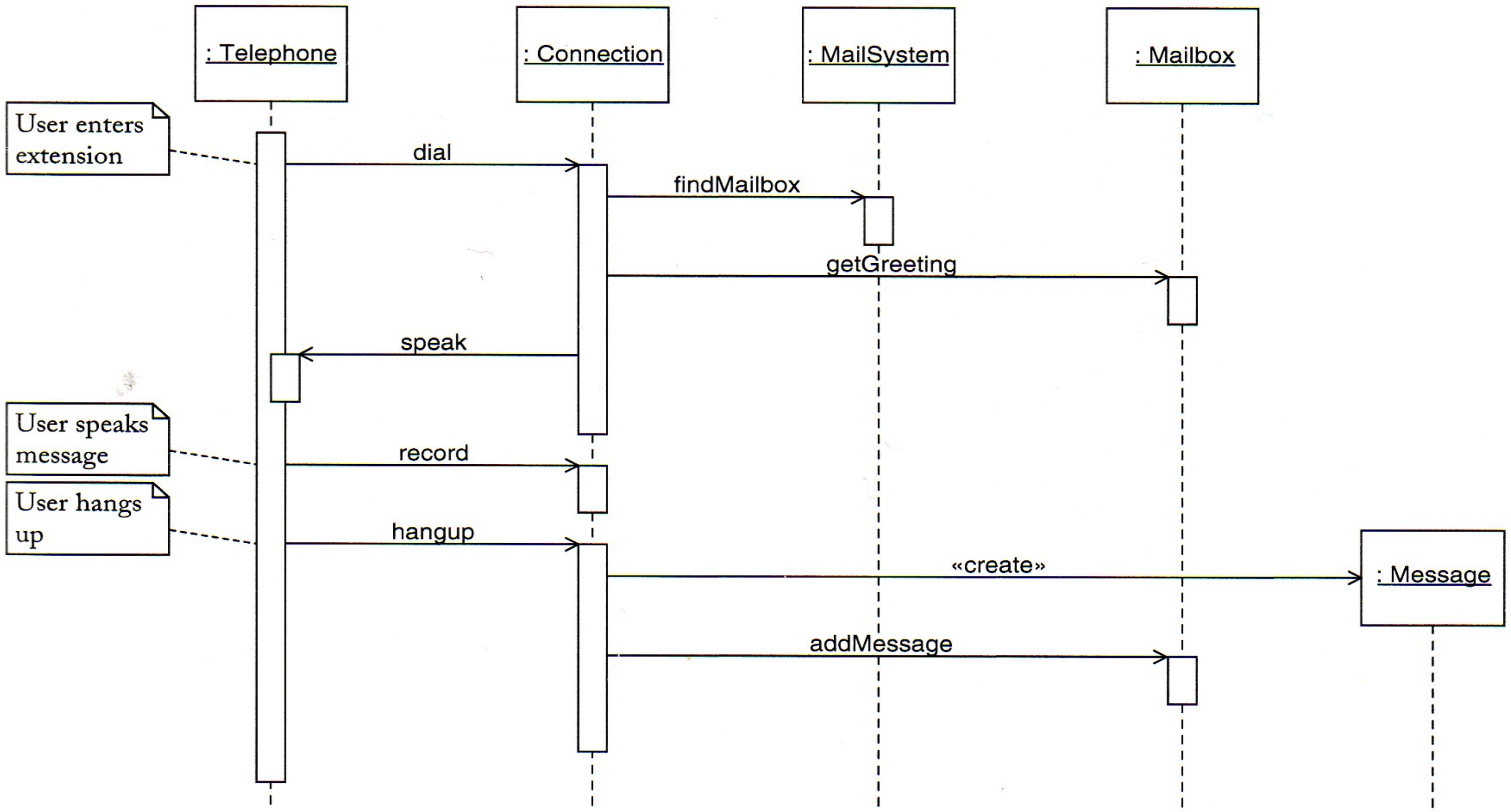
- A **MailSystem** has **mailboxes**.
- A **mailbox** has two **message queues**.
- A **message queue** has some number of **messages**.
- A **Connection** has a current **mailbox**. It also has references to the **Mai1System** and **Telephone** objects that it connects.



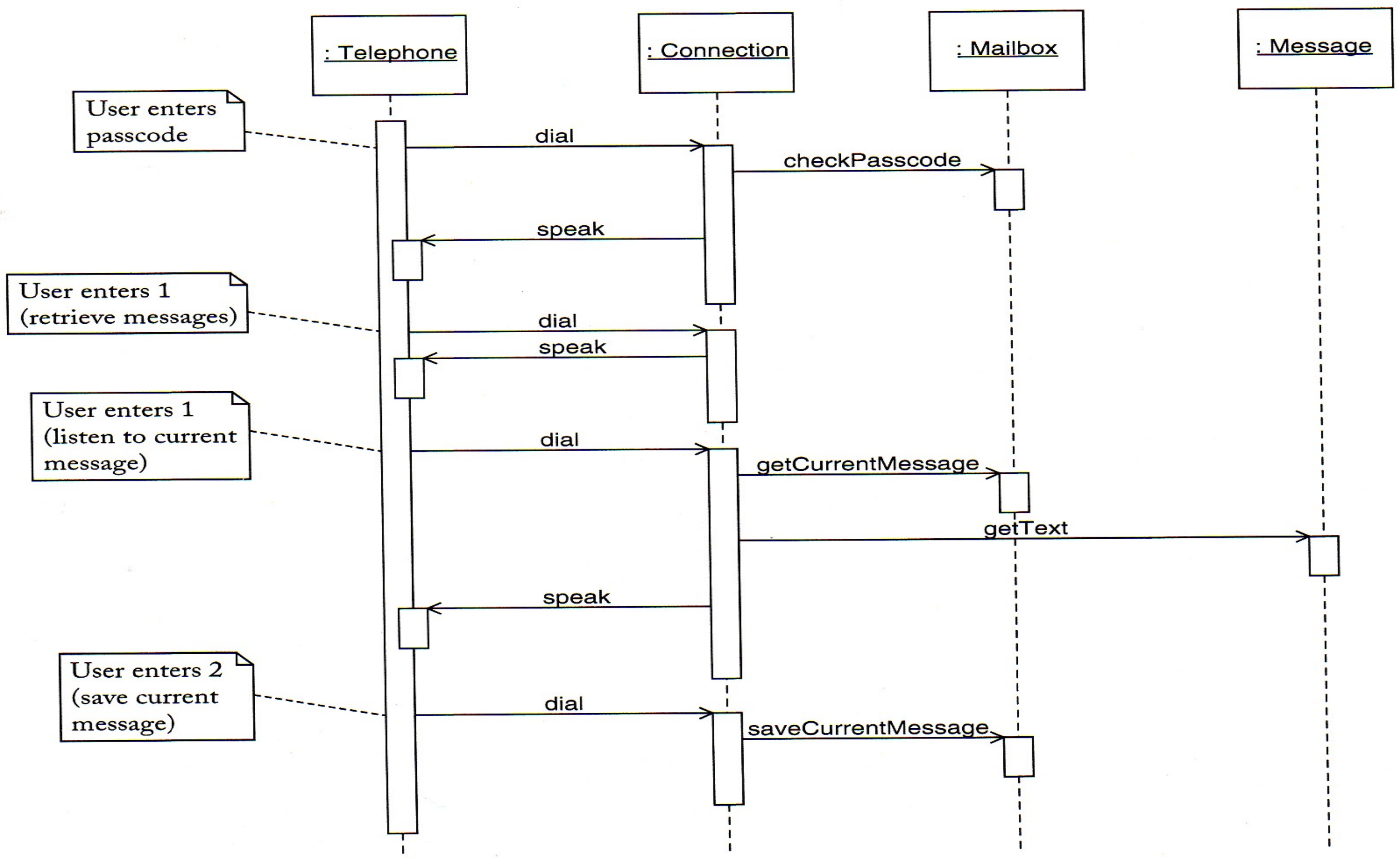
There is no inheritance relationship between the classes.

UML Sequence Diagrams

Sequence Diagram for Leaving a Message



Sequence Diagram for Retrieve Messages


: Telephone
: Connection
: Mailbox
: Message
User enters passcode
dial
checkPasscode
speak
User enters 1 (retrieve messages)
dial
speak
User enters 1 (listen to current message)
dial
getCurrentMessage
getText
speak
User enters 2 (save current message)
dial
saveCurrentMessage