

The Object-Oriented Design Process

Horstmann Section 2.12

CS 4354
Summer II 2016

Jill Seaman

1

2.12 Case Study: A Voice Mail System

Consider the task of writing a program to simulate the following simple telephone voice mail system:

- (Assuming the caller has already dialed into the system)
A person dials an extension number and, provided the other party does not pick up the telephone, receives a greeting and leaves a message. The other party can later retrieve their messages, keep them, or delete them, once they have provided their passcode.
- We will use text to represent voice, phone keys, hanging up:
 - ◆ 1 2 ... 0 # on a single line denotes the corresponding phone key
 - ◆ H on a single line means "hang up"
 - ◆ All other inputs mean voice
- We will begin with the analysis phase and develop use cases.

2

2.12.1 Use cases for the Voice Mail System

Use Case: Reach an Extension

Reach an Extension:

1. The caller dials the main number of the voice mail system.
2. The voice mail system speaks the following prompt:
`Enter mailbox number followed by #.`
3. The caller types in the extension number of the message recipient.
4. The voice mail system speaks the following message:

`You have reached mailbox xxxx. Please leave a message now.`

3

Use Case: Leave a Message

Leave a Message

1. The caller carries out **Reach an Extension**.
2. The caller speaks the message.
3. The caller hangs up.
4. The voice mail system places the recorded message in the recipient's mailbox.

4

Use Case: Log in

Leave a Message

1. The mailbox owner carries out **Reach an Extension**.
2. The mailbox owner types the passcode and the # key (Default passcode = mailbox number.
To change the passcode, see **Change the Passcode**)
3. 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.

5

Use Case: Retrieve Messages

1. The mailbox owner carries out **Log in**.
2. The mailbox owner selects option 1 retrieve your messages.
3. 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.
4. The mailbox owner selects option 1 listen to the current message.
5. The voice mail system plays the current new message, or, if there are no new messages, the current old message. Note that the message is played, not removed from the queue.
6. The voice mail system plays the message menu.
7. The user selects option 3 delete the current message. The message is permanently removed.
8. Continue with Step 3.

6

Use Case: Retrieve Messages - Variations

Variation #1 Saving a message

- 1.1. Start at Step 6
- 1.2. The user selects option 2 save the current message. The message is removed from its queue and appended to the queue of old messages.
- 1.3. Continue with Step 3.

7

Use Case: Change the Greeting

Change the Greeting

1. The mailbox owner carries out **Log in**.
2. The mailbox owner selects option 3 change your greeting.
3. The mailbox owner speaks the greeting.
4. The mailbox owner presses the # key.
5. The mail system sets the new greeting.

Variation #1. Hang up before confirmation

- 1.1. Start at Step 3.
- 1.2. The mailbox owner hangs up the telephone.
- 1.3. The mail system keeps the old greeting.

8

Use Case: Change the Passcode

Change the Passcode

1. The mailbox owner carries out **Log in**.
2. The mailbox owner selects option 2 change your passcode.
3. The mailbox owner dials the new passcode
4. The mailbox owner presses the # key.
5. The mail system sets the new passcode.

Variation #1. Hang up before confirmation

- 1.1. Start at Step 3.
- 1.2. The mailbox owner hangs up the telephone.
- 1.3. The mail system keeps the old passcode.

9

2.12.2 CRC Cards for the Voice Mail System

- Next let's discover the classes for the system.
- These are obvious
 - ◆ Mailbox
 - ◆ Message
 - ◆ MailSystem
- **Mailbox** responsibilities:
 - ◆ keep messages
 - ◆ which are new, which are saved.
 - ◆ users should be able to retrieve, save, and delete messages
 - ◆ need a queue to store the messages in FIFO order: MessageQueue

10

Initial Mailbox CRC Card

- Remember: responsibilities on the left, collaborating classes on the right.

Mailbox	
<i>keep new and saved messages</i>	MessageQueue

11

Initial MessageQueue CRC Card

MessageQueue
<i>add and remove messages in</i>
<i>FIFO order</i>

12

Initial MailSystem CRC Card

- We need a class to contain (and find) the Mailboxes

MailSystem	
<i>manage mailboxes</i>	Mailbox

13

Initial Telephone CRC Card

- We need a class to manage input and output

Telephone	
<i>take user input from touchpad,</i>	
<i>microphone, hangup</i>	
<i>speak output</i>	

14

Initial Connection CRC Card

- We need a class to manage the connection between a telephone and the MailSystem
- The telephone could do this, but it would have too much responsibility.

Connection	
<i>get input from telephone</i>	Telephone
<i>carry out user commands</i>	MailSystem
<i>keep track of state</i>	

15

CRC Card process: Walkthrough Leave a Message

- 1 User dials extension. Telephone sends number to Connection
(Add collaborator Connection to Telephone)
- 2 Connection asks MailSystem to find matching Mailbox
- 3 Connection asks Mailbox for greeting
(Add responsibility "manage greeting" to Mailbox,
add collaborator Mailbox to Connection)
- 4 Connection asks Telephone to play greeting
- 5 User speaks message. Telephone asks Connection to record it.
(Add responsibility "record voice input" to Connection)
- 6 User hangs up. Telephone notifies Connection.
- 7 Connection constructs Message
(Add card for Message class,
add collaborator Message to Connection)
- 8 Connection adds Message to Mailbox

16

Changed/Added CRC Cards

Telephone		Connection	
<i>take user input from touchpad,</i>	Connection	<i>get input from telephone</i>	Telephone
<i>microphone, hangup</i>		<i>carry out user commands</i>	MailSystem
<i>speak output</i>		<i>keep track of state</i>	Mailbox
		<i>record voice input</i>	Message

Mailbox		Message	
<i>keep new and saved messages</i>	MessageQueue	<i>manage message contents</i>	
<i>manage greeting</i>			

17

CRC Card process: Walkthrough Retrieve Messages

- 1 User types in passcode. Telephone notifies Connection
- 2 Connection asks Mailbox to check passcode.
(Add responsibility "manage passcode" to Mailbox)
- 3 Connection sets current mailbox and asks Telephone to speak menu
- 4 User selects "retrieve messages". Telephone passes key to Connection
- 5 Connection asks Telephone to speak menu
- 6 User selects "listen to current message". Telephone passes key to Connection
- 7 Connection gets first message from current mailbox.
(Add "retrieve messages" to responsibility of Mailbox).
- 8 Connection asks Telephone to speak message
- 9 Connection asks Telephone to speak menu
- 10 User selects "save current message". Telephone passes key to Connection
- 11 Connection tells Mailbox to save message
(Modify responsibility of Mailbox to "retrieve,save,delete messages")
- 12 Connection asks Telephone to speak menu

18

Updated Mailbox CRC Card

- New responsibilities assigned after Retrieve Messages walkthrough

Mailbox	
<i>keep new and saved messages</i>	MessageQueue
<i>manage greeting</i>	
<i>manage passcode</i>	
<i>retrieve, save, delete messages</i>	

19

CRC Cards process

- Use one card per class
- State responsibilities at high level
- Use scenario walkthroughs to fill in cards, make new ones, move responsibilities from one card to another, delete unneeded cards.
- Usually, the first design isn't perfect.
- The process should be iterative.
- (You just saw the author's third design of the mail system)

20

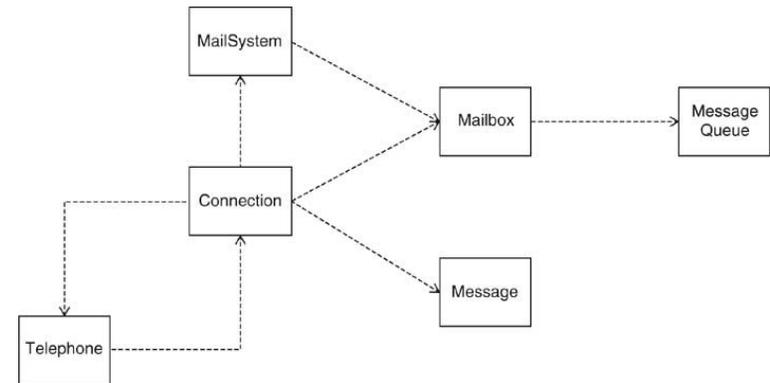
2.12.3 Class diagrams for the Voice Mail System

CRC card collaborators yield dependencies:

- Mailbox depends on MessageQueue
- MailSystem depends on MailBox
- Connection depends on Telephone, MailSystem, Message, Mailbox
- Telephone depends on Connection
- Message doesn't depend on anything
- MessageQueue doesn't depend on anything

21

The Voice Mail System Dependencies from the CRC Cards



22

Class diagrams for the Voice Mail System

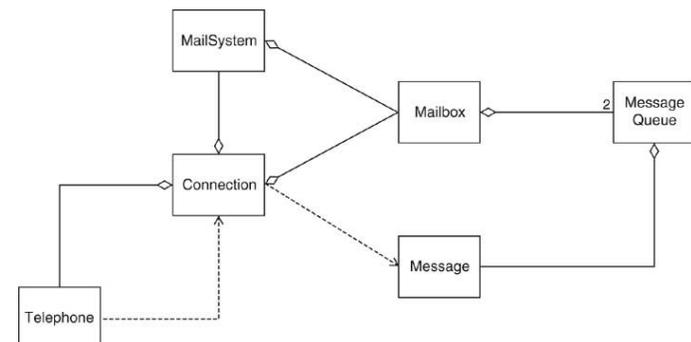
Next, consider the aggregation relationships:

- A MailSystem has Mailboxes
- A Mailbox has two Message Queues
- A Message Queue has some number of Messages
- A Connection has a current Mailbox.
- A Connection has references to a MailSystem and a Telephone

23

The Voice Mail System from the CRC Cards

- There is no inheritance among these classes
- Aggregation associations replace dependency associations



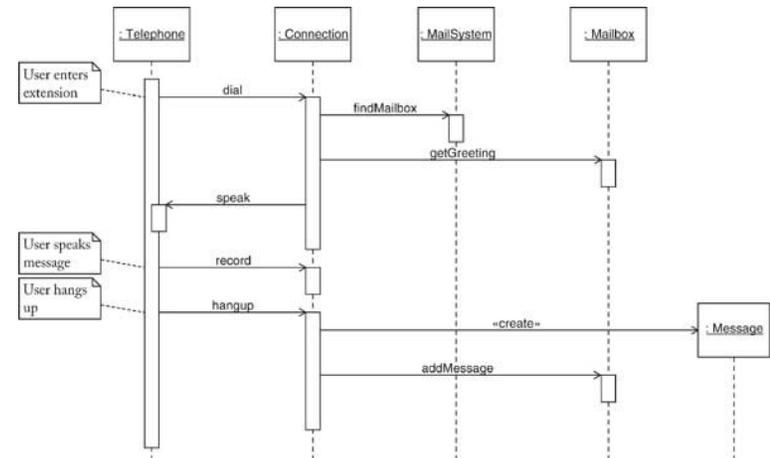
24

2.12.4 UML Sequence and State Diagrams

- The purpose of a sequence diagram is to understand a complex control flow that involves multiple objects..
- A sequence diagram usually represents the steps of a use case.
- They are not needed for simple interactions.
- For the Voice Mail System, the interactions between the Telephone, Connection, MailSystem, and Mailbox classes are not easy to understand.
- So we'll draw a sequence diagram for Leave a Message.

25

Sequence Diagram for Leave a Message



26

Sequence Diagram: Leave a Message

- During this process, we decide what methods the classes need.
- Connection needs three methods to receive data from the Telephone:
 - dial passes on a button press.
 - record passes on (some) speech.
 - hangup tells the connection that the telephone has hung up.
- Connection needs to get greeting to play
 - Each mailbox knows its greeting
 - Connection must find mailbox object:
 - Call findMailbox on MailSystem object, then call getGreeting on that Mailbox
 - Call speak on the Telephone to play the greeting.
- Telephone records the (next line of the) message, and passes it to the Connection by calling record.
- Telephone calls hangup.

27

Sequence Diagram: Leave a Message

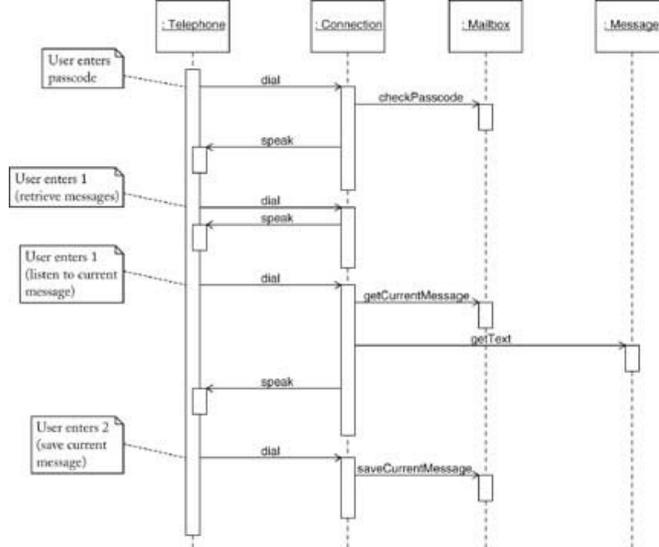
- Connection must construct a Message and add it to the mailbox (addMessage)
 - which mailbox object?
 - the one from the previous call to findMailbox

Note:

- Each key press results in separate call to dial, but only one is shown
- Parameters are not displayed (e.g. mailbox number)
- Return values are not displayed (e.g. found mailbox)
- Note that connection holds on to that mailbox over multiple calls

28

Sequence Diagram for Retrieve Messages



29

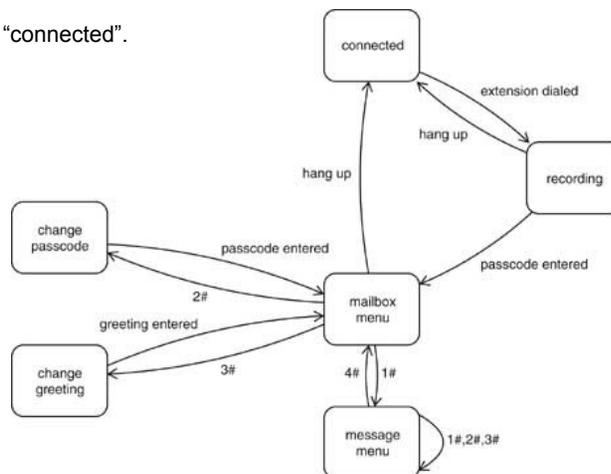
State diagrams

- Input is not controlled by the voice mail system
 - ◆ It responds to input from the Telephone
- The Connection needs to keep track of what state it is in so it knows how to respond to the given input.

30

State diagram of the Connection class

The starting state is "connected".



31

2.12.5 Java Implementation

- Compare the Java code to the CRC cards, the class diagram, the sequence diagrams, and the state diagram
- A link to the code is on the class website.

32