Abdurrahman Mohammad

Ying Chang Cui

Eduardo Fragoso

Emmanuel Cestoni

**Design Phase**

Message

Responsibilities:

* Stores a message
* Returns the message
* Compares messages

Fields:

* Message: A String variable that stores a message

Methods:

* Message(String message): Constructs a message with String input
* getMessage(): Returns the saved message as a String
* compare(Message o1, Message o2): Compares 2 messages alphabetically
* comparatorMessage(): Compares 2 messages alphabetically

Greeting

Responsibilities:

* Stores 3 greetings
* Sets a default greeting (one of the greetings in the array)
* Returns the greetings including the default greeting
* Changes and removes greetings

Fields:

* Message[] greetings: An array of Messages to store 3 greeting messages
* defaultGreeting: The index of the default greeting in the greetings array

Methods:

* Greeting(): Constructor constructs a new Greeting object
* addGreeting(int i, Message message): adds a greeting to the array at index i
* removeGreeting(int i): Removes the recorded greeting at index i
* getGreeting(int i): Returns the default greeting as a Message at index i
* setDefault(int i): Sets the index of the default greeting
* getDefaultGreeting(): get the default greeting (at index ‘defaultGreeting’)
* comparatorByDefaultGreeting(): Compared two Greeting objects by the alphabetical order of their default greetings

MessageQueue

Responsibilities:

* Stores messages in a queue
* Returns, adds, and deletes messages in queue
* Can sort messages in alphabetical order

Fields:

* messages: An ArrayList of messages (ArrayList can be sorted with Collections)

Methods:

* MessageQueue(): Constructs an empty message queue
* playMessage(int i): Plays the message at index i
* enqueue(Message message): Adds a message to queue
* dequeue(int i): Removes message at index i
* eraseQueue(): Removes all elements from queue
* size(): Returns the size of queue
* AlphabeticalSort(): Sorts the messages in queue in alphabetical order
* comparatorByNumberOfMessages(): Returns a comparator to sort queues based on the number of messages

Mailbox

Responsibilities:

* Stores a queue of unread and saved messages
* Checks the password with an inputted password
* Checks if the owner is present to answer the call
* The owner can manage mailbox’s greeting object
* The owner can manage the mailbox’s two queues of messages
* The owner can also compare mailboxes

Fields:

* phoneNumber: The assigned phone number
* password: The password of the mailbox
* greeting: Greeting of the mailbox
* isPresent: A Boolean variable to show the user is present
* unreadMessages: A queue of unread messages
* savedMessages: A queue of saved/read messages
* gui: A GUI from the class that user the Mailbox class to maintain a single, simultaneous GUI for optimum user experience.

Methods:

* Mailbox(int phoneNumber, int password): Constructor to construct an empty mailbox
* leaveMessage(): Plays a greeting and lets the user leave a message
* passwordCheck(int pswd): Checks the password too see if the inputted password is correct
* unreadMessageManagement(): Plays unread messages in queue and lets user save or delete messages
* savedMessageManagement(): Plays saved messages in queue and lets user save or delete messages
* greetingManagement(int option, int index, Message message): Method to manage the mailbox greeting object. You may add, remove, retrieve, and change default greeting.
* getPhoneNumber(): Returns the phone number of the mailbox
* isPresent(): Returns true if the user is present
* setPassword(int pswd): Method to set the password of the mailbox
* setGui(): Sets the GUI used by the Mailbox class
* comparatorByPhoneNumber(): a comparator to sort based on phone numbers
* comparatorByUnreadMessages(): a comparator to sort based on the number of unread messages
* comparatorBySavedMessages(): a comparator to sort based on the number of saved messages

MailboxInterface

Responsibilities:

* Contain the basic unimplemented methods necessary for a basic phone line's mailbox

Methods:

* leaveMessage(): Lets the user leave a message
* unreadMessageManagement(): Lets the user check the unread messages
* savedMessageManagement(): Lets the user check the saved messages
* greetingManagement(): Lets the user add, remove, change, and set default mailbox greeting
* getPhoneNumber(): Returns the mailbox's phone number
* passwordCheck(int pswd): Lets the user check the password before accessing the mailbox
* setPassword(int pswd): Sets the mailbox's password. Password format, range, and length is determined by the user.

VoicemailSystem

Responsibilities:

* Calls an extension
* If the user does not pick up, the caller goes to voicemail
* The caller can leave a message
* The owner can enter the voicemail password and manage messages and greeting
* The admin can add a mailbox, remove a mailbox, change mailbox password, and reset the system by removing all the mailboxes

Fields:

* An ArrayList of mailboxes which serve as the extensions
* The default admin password

Methods:

* VoicemailSystem(): Constructs the voicemail system
* initiate(): Initiates the protocol of connecting the caller by combining the other methods
* callExtension(Mailbox ext): Call a mailbox and speak to the mailbox owner. If owner is not present, the caller may leave a message. If the owner calls his own mailbox, he can manage his messages and greetings.
* mailboxManager(Mailbox ext): Manage the messages (unread and saved) and the greeting object of a mailbox
* adminMenu(): Allows you to add a mailbox, remove a mailbox, change a mailbox password, or reset the system (by deleting all the mailboxes)
* addMailbox(int phoneNumber): Adds a mailbox with the specified phone number
* removeMailbox(int phoneNumber): Removes the mailbox with specified phone number
* findMailbox(int ext): Finds the mailbox with a specified extension phone number and returns its reference
* comparatorByMailboxes(): Returns a comparator to sort based on the number of mailboxes

GUI

Responsibilities:

* Creates and sustains the GUI of the voicemail system
* Prints out all the Strings and messages passed in by voicemail system
* Inputs all the numbers/keys requested by voicemail system and returns the keys pressed
* Simulates speaking
* Simulates pressing keys

Fields:

* FIELD\_WIDTH: The width of the text field (constant)
* keyInput: Stores the numeric input from the keypad. Updates at every key press.
* A JTextField called textField that makes a text field for input/output

Methods:

* GUI(): Constructs and initializes the GUI with buttons and defines the acito listeners
* getKey(): get 1 key from the user and return it
* get3Key(): get 3 keys from the user and return them
* printMessage(Message message): Prints out a message object to the text field
* printString(String output): Prints Strings to the text field
* getMessage(): Gets a message from the user
* speak(): Simulates speaking

Telephone (Main)

Responsibilities:

* Creates a VoiceMailSystem and tests the program

Methods:

* Main: This method would test the voice mail system. It would call a number and if the person does not pick up, you can leave a message. You can also access the messages and set up mailboxes. In this method, you simulate the caller, owner, or the admin calling and using the voice mail system.

**Relationships:**

Message:

* **Implements** Comparator

Greeting:

* **Contains/Aggregate** 3 Message objects
* **Implements** Comparator

MessageQueue

* **Contains/Aggregate** an ArrayList
* **Uses** Message
* **Implements** Comparator

Mailbox

* **Implements** MailboxInterface
* **Implements** Comparator
* **Uses** GUI
* **Contains/Aggregate** 1 Greeting
* **Contains/Aggregate** 2 MessageQueue objects

GUI

* **Uses** java.awt.GridLayout;
* **Uses** java.awt.event.ActionEvent;
* **Uses** java.awt.event.ActionListener;
* **Contains/Aggregate** 12 javax.swing.JButton;
* **Contains/Aggregate** 1javax.swing.JFrame;
* **Contains/Aggregate** 1javax.swing.JPanel;
* **Contains/Aggregate** 1 javax.swing.JTextField;
* **Uses** message.Message;

VoicemailSystem

* **Contains/Aggregate** 1 ArrayList
* **Uses** Collections
* **Implements** Comparator
* **Uses** GUI
* **Uses** Mailbox

Telephone

* **Contains/Aggregate** VoicemailSystem