**Design Phase**

Message

Responsibilities:

* Stores a message
* Returns the message

Fields:

* Message: A String variable that stores a message

Methods:

* Default Constructor: The default constructor constructs a Message object with no actual message
* Overloaded Constructor: The overloaded constructor of the Message class creates a new Message object and stores a message passed in
* Get Method: returns the saved message as a String
* Set Method: Modifies/Replaces the saved message

Greeting

Responsibilities:

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

Fields:

* An array of Strings to store 3 greetings
* The index of the default greeting as an integer

Methods:

* Constructor
* Add greeting: adds a greeting to the array at index i
* Remove Greeting: Removes the recorded greeting at index i
* Set Default Greeting: Sets the index of the default greeting
* Get Greeting: Returns the default greeting as a String

MessageQueue

Responsibilities:

* Stores messages in a queue

Fields:

* An array of messages (max 10)
* A variable to record size

Methods:

* Play Message: Plays the message at index i
* Enqueue: Adds a message to queue
* Dequeue: Removes a message and fixes up queue to fill in any null spaces
* Get Size: Returns the size of queue
* Is Full: Returns true if the queue is full

Mailbox

Responsibilities:

* Plays a default greeting for the caller
* Checks the password with an inputted password
* Connects the caller to the owner if the owner is present
* If the owner is not present, the caller can add a message to the unread message queue
* The owner can record a greeting and replace a greeting stored in the greeting object at index i
* The owner can set the default greeting
* Adds messages to the queues saved messages and unread messages
* Removes messages from the queues saved messages and unread messages
* Lets the user listen to the message queues
* Lets the Admin change the password

Fields:

* A queue to store unread messages
* A queue to store read messages
* A Boolean variable to show the status of the mailbox’s activation
* A Boolean variable to show the user is present
* A Boolean variable showing the status of the mailbox being full (optional)

Methods:

* Constructor
* Play the default greeting: Lets the caller hear the default greeting
* Record a greeting: Adds a greeting to the list of 3 greetings
* Remove a greeting: Removes a greeting
* Set a default greeting: Sets a default greeting among the 3
* Leave a Message: Lets the caller leave a message and adds it to the unread messages queue
* Play Unread Messages: Lets the user listen to the unread messages
* Play Read Messages: Lets the owner listen to the queue of read messages
* Delete Selected Unread: Deletes the selected unread message
* Delete Selected Read: Deletes the selected read message
* ~~Delete All Unread: Deletes all the messages in the unread messages queue~~
* ~~Delete All Read: Deletes all the messages in the unread messages queue~~
* Password Check: Checks the password inputted with the password of the mailbox
* Set Password: Lets the admin the password
* Is Activated: Lets the admin activate or deactivate the mailbox
* Set Activated: Sets the activation of the mailbox
* Mailbox is full: Returns true if the mailbox is full (optional)

IO

Responsibilities:

* Prints out all the Strings passed in by VoiceMailSystem
* Inputs all the Strings requested by VoiceMailSystem
* The voice mail system uses this class for input and output

Fields:

* Scanner

Methods:

* Get Integer Input: Gets input from the caller and passes it to the voicemail system
* Get String Input: Gets input message of the user to simulate speaking. The information is sent back to the voicemail system.
* Print Output: Prints information sent by the voicemail system

VoiceMailSystem

Responsibilities:

* Simulates pushing a button
* 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 retrieve the messages
* The owner can delete messages
* The admin can activate an extension and set its password

Fiels:

* An array of mailboxes which serve as the extensions

Methods:

* Constructor
* Initiate: Initiates the protocol of connecting the caller by combining the other methods
* Play Greeting: Welcomes the caller and asks for an extension
* Push Button: Simulates pushing a button
* Search Extension: Verifies that the extension is activated and is in range
* Invalid Extension: Plays an error message
* Call: Calls the extension and if the user is present, they talk and hang up
* Speak: Simulates talking. Used when the user picks up or leaves a message.
* Hang up: this method will exit the system
* Voicemail: If the owner is not available (after call), the caller can leave a message or check voicemail. The caller is asked to either enter a password or press ‘#’ to leave a message. Leaving a message utilizes Mailbox methods.
* Voicemail Management: If the password is correct, the caller can listen to messages, delete messages, and manage the greetings. This utilizes Mailbox methods.
* Admin Menu: If the caller dials ‘000’ and enters the password in the package which is ‘987, then he can activate or deactivate mailboxes and set the passwords. This utilizes mailbox methods.

Telephone (Main)

Responsibilities:

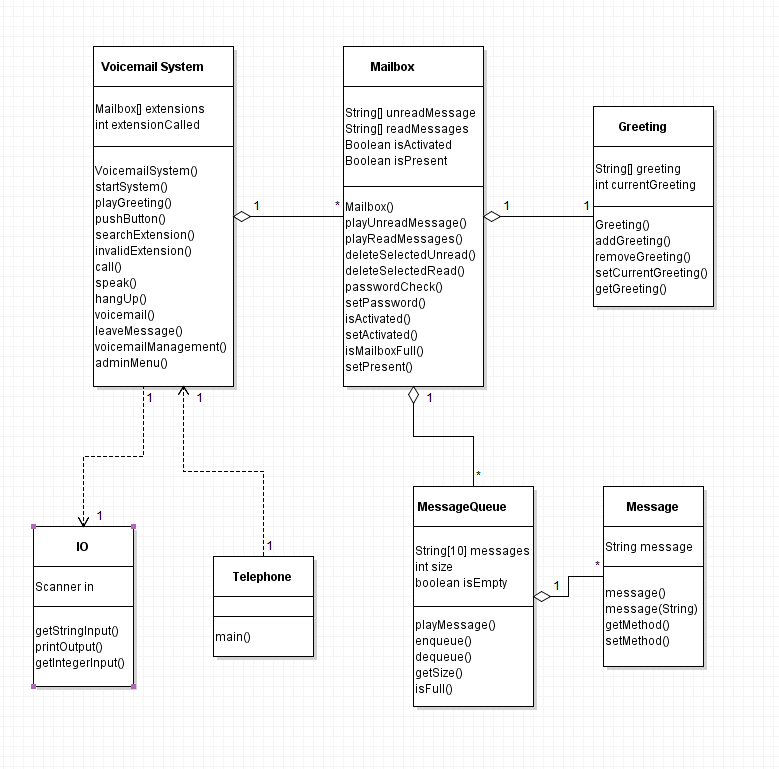
* Creates a VoiceMailSystem and tests the program

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:**

* Telephone **uses** VoicemailSystem
* VoicemailSystem contains 0 to n (**aggregation**) mailboxes Mailbox and **uses** IO
* IO **uses** Scanner and **uses** System.out objects
* Mailbox contains 1 (**aggregation**) Greeting and contains 0 to n(**aggregation**) MessageQueue
* MessageQueue contains 0 to n (**aggregation**) Message type messages

**Class Diagram**

**Sequence Diagrams**

**Setup Greeting**

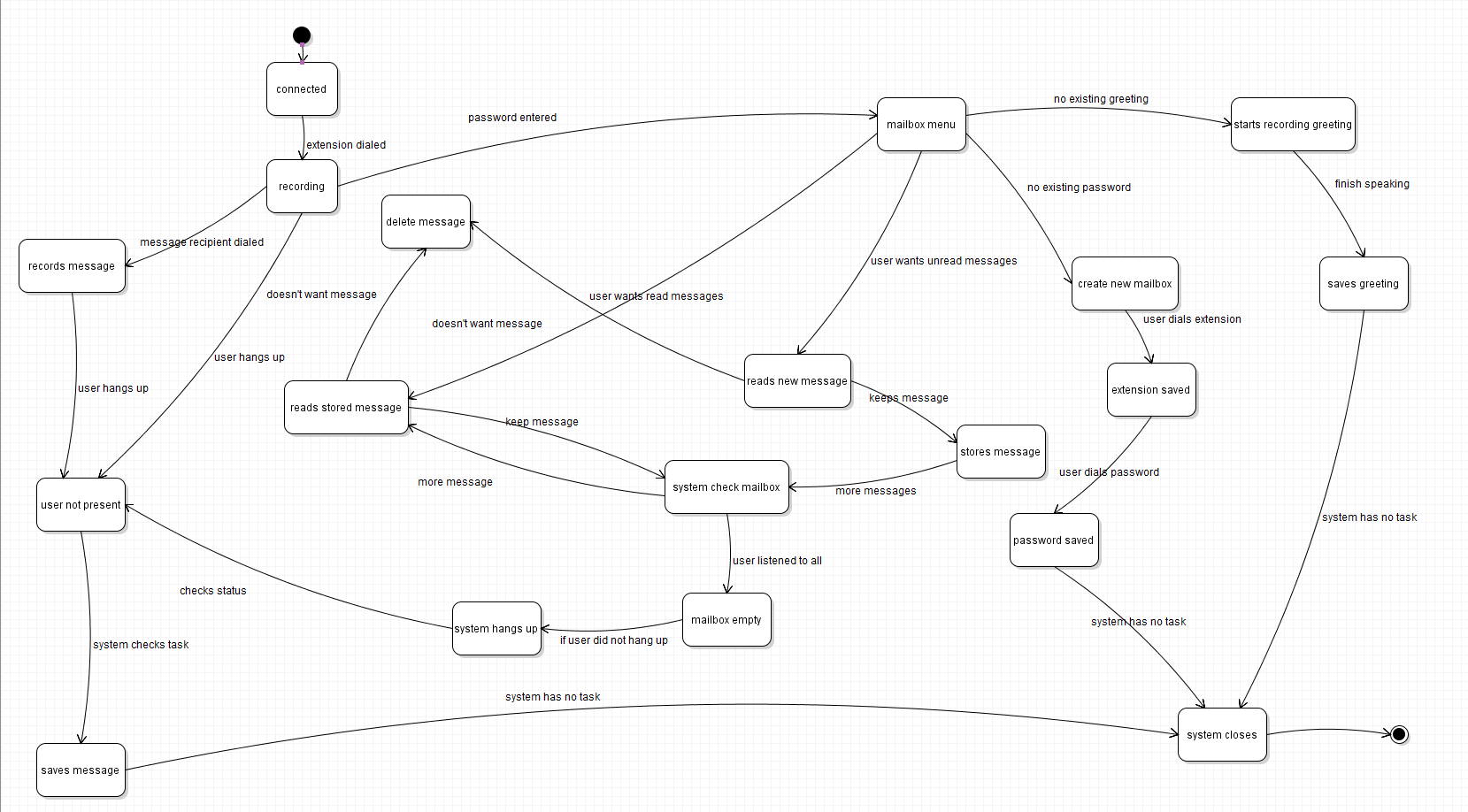
**Playing a Message**

**Playing a Message**

**Leaving a Message**

**Deleting a Message**

**State Diagram**

****