Cold-Call Assist
Programmer Documentation

Table of Contents

1. Introduction

This document describes each source file used within the Cold-Call Assist program. The functions within each file are described along with their parameters and uses.

2. Cold Call Source Files

2.1 Objects.py

2.1.1 class Student()

- The student class includes an init method which defines the parameters of a student (first name, last name, UO ID, email, phonetic spelling, reveal code, number of times called, flag, number of flags and the dates they have been called on).
- printStudent() displays the students first name, last name, and ID number.
- display() only prints a students first name and last name.
- summaryPerformance() which returns the exact line which is used to create the summary performance log file.
- review() can be called by the output file function feedback to show student information.
- Provides support for a helpful getter called getFlag() which returns a students flag value.
- Also provides support for a helpful setter called setFlag() which sets a students flag value.

2.1.2 class classQueue()

- Includes an init method which defines the queue and a length of the queue
- A method called enqueue(new_student) takes a new student object and pushes them to the queue.
- A function called insertOne(new_student, i) takes a new student object and a position i in which they should be inserted into the student queue.
- A helpful function called dequeue() removes the student at the tail of the queue
- Another function called removeIndex(i) removes whichever student is at index i within the queue.
- A helpful function isEmpty() checks whether there are any students within the queue.
- Initializes a method called combine(new_list) which combines the new queue with the current queue.
- Supports a function called printQ() which prints the contents of the student queue (using printStudent() helper from student class)

2.2 HOME.py

2.2.1 switch_view()

- This is the primary method used for switching from HOME menu and opening the User View secondary window.
- This method inherits from GUI python file.

2.2.2 inputFile()

- This is the primary function which handles input into our program. It reads from a specified input file syntax, and populates our student queue (see objects.py). The function implements error checking to ensure the file is correctly formatted.

2.2.3 writeSummaryPerformanceFile()

- This method is used for populating the summary performance file. The summary performance file is one of two log files supported by our program. Similar to inputFile(), this function uses error checking to ensure that contents from our student queue are added to the log file.

2.2.4 writeLogFile()

- This method is used for populating the daily log file. The daily log file is used to view student participation for the previous day's cold calling. Error checking is implemented to prevent errors within this file.

2.2.5 exports()

- This is a simple helper function which we use to write to both log files during usage. This function is called when a user presses keys associated to flagging a student or in any way adjusting their student data within the queue.

2.2.6 getSettings()

- This method attempts to open config.txt and read in the first line, which represents the path to the data file. If config.txt is opened successfully, the first line contains a path that corresponds to a valid data file, and the global 'ROSTERPATH' variable is set, then the function returns a success code. In all other cases, the program will continue as normal. It is expected that this method will not be successful the first time the program is run, since config.txt does not exist until created by the first call to setSettings().

2.2.7 setSettings()

- This method writes the path of the roster file to config.txt, where it can be read by getSettings(). This function is called after the user selects a valid .txt file from the file selection window, and the file's path is passed as an argument.

2.2.8 openDaily()

- This is another helper function which is coordinated to a button on the user view. Upon clicking the “Daily Log File” this function is called, and opens the daily log file.

2.2.9 openSummary()

- This helper function is similar to openDaily(), it simply opens the summary performance file upon clicking “Performance File” in the User View window.

2.2.10 exitProgram()

- This function is connected to the quit button on the HOME menu. It will close all windows currently open by the program, effectively exiting the cold call system.

2.3 Control.py

This file holds the algorithm responsible for producing a fair and random queue. Within a reasonable class size, greater than 5, this algorithm will guarantee:

Randomness: you cannot guarantee who will be next.

Fairness: the chance of a student being called again right after being called on is 0.01.

A student will not be called continuously (cool down time). The bigger size of the student roster, the longer the cool down time will be.

To test or run this file:

1. Comment out line 19 (“from backend.objects import Student, classQueue”)
2. Uncomment line 18 (“#from objects import Student, classQueue)
3. python3 control.py

2.3.1 N_index(length)

- Given the length of a roster, according to the global variable N (which is 45) it will return index_n of the roster so that the next student will be picked from index 0 to index_n (within the first n% of the roster).

2.3.2 pickOneStudent(onDeck, Roster)

- Given a classQueue onDeck and a classQueue Roster, return the randomly picked student Object. onDeck and Roster will be updated as a side effect. The range to pick the student is indicated by N_index()

2.3.3 initRoster()

- Return a classQueue Roster. This function is used to test the functions in this file and will be replaced by file I/O.

2.3.4 initDeck(Roster)

- Given a classQueue Roster, return a classQueue onDeck which contains the four student names that are on deck.

2.3.5 left(cur_index, onDeck, Roster)

- Given a classQueue onDeck and a classQueue Roster and a current index return an updated current index. This function will do the corresponding updating of the onDeck and Roster when the left key is pressed.

2.3.6 right(cur_index, onDeck, Roster)

- Given a classQueue onDeck and a classQueue Roster and a current index return an updated current index. This function will do the corresponding updating of the onDeck and Roster when the right key is pressed.

2.3.7 up(cur_index, onDeck, Roster, flagQ)

- Given a classQueue onDeck and a classQueue Roster and a current index return an updated current index. This function will do the corresponding updating of the onDeck and Roster when the up key is pressed.

2.3.8 down(cur_index, onDeck, Roster)

- Given a classQueue onDeck and a classQueue Roster and a current index return an updated current index. This function will do the corresponding updating of the onDeck and Roster when the down key is pressed.

2.3.9 importFun()

- The purpose of this function is to check if control.py is connected to GUI.py.

2.3.10 testRandomness()

- This function will call pickOneStudent() 10000 times to see if the chance of every student being called is similar. The following is one of the reports. Notice the report will be different due to the randomness of pickOneStudent()

```
############ Report ###################


Queue at index 0 has Nation 839 chance been called:  0.0839
Queue at index 1 has Yin 836 chance been called:  0.0836
Queue at index 2 has Haihan 832 chance been called:  0.0832
Queue at index 3 has Jimmy 830 chance been called:  0.083
Queue at index 4 has Quan 781 chance been called:  0.0781
Queue at index 5 has Lucas 837 chance been called:  0.0837
Queue at index 6 has Maura 840 chance been called:  0.084
Queue at index 7 has Harry 840 chance been called:  0.084
Queue at index 8 has Noah 845 chance been called:  0.0845
Queue at index 9 has Anthony 817 chance been called:  0.0817
Queue at index 10 has Jessie 875 chance been called:  0.0875
Queue at index 11 has Ann 828 chance been called:  0.0828

Range for chance been called: 0.0094
```

Figure 1: testRandomness() report

2.4 GUI.py

2.4.1 Class GUI

- __init__()
  - Initializes the User View screen to display four names for the instructor to call on.
- CloseWindow()
  - Closes the User View window. This will be used by the exitProgram() function in HOME.py
- leftKey()
  - Calls left() from control.py and updates the user interface.
  - This method is called whenever a left arrow key is pressed. It moves the highlighting one student to the left of the current student. If the student is already at the far left, the highlighting wraps around to the far right student.
- rightKey()
  - Calls right() from control.py, and updates the user interface.
  - This method is called whenever a right arrow key is pressed. It moves the highlighting one student to the right of the current student. If the student is already at the far right, the highlighting wraps around to the far left student.
- upKey()
  - Calls up() from control.py, and updates the user interface and database.
  - This method is called whenever an up arrow key is pressed. This removes the student from the names being displayed, and the name is saved in the log files as flagged.
- downKey()
  - Calls down() from control.py, and updates the user interface and database.
  - This method is called whenever a down arrow key is pressed. This removes the student from the names being displayed, and the name is saved in the log files as unflagged.
- Update()
  - Updates the names displayed and/or the name being highlighted

2.4.2 displayMessage()

- Displays an apple notification on the top right corner with a message.

2.4.3 userViewOpen()

- Used to keep track of whether or not a user view window is already open to prevent two windows from opening simultaneously. Returns 1 if a window is already open, and returns 0 if there is no open window.

2.4.4 getUserViewWindow()

- Returns the current User View window object. Used in HOME.py to force the window to close if the home menu window closes.

2.4.5 testArrowKeys()

- A test function to make sure the window could register key presses.

2.4.6 testScreenUpdate()

- A test function to make sure the highlighting could move positions as well as making sure names could be updated.

2.4.7 testcontrol()

- The main function for running the User View Window.

2.4.8 Backend:

- classQueue Roster (all the students in the class).
- classQueue onDeck (four students on deck).
- current_index (which name is highlighted).
- classQueue flagQ (a list of students that had been flagged that day).
- self.path = “” (path to database).

2.4.9 overwriteRosterFile():

- This method overwrites the data held in the roster file. The first line of the file is preserved, while the subsequent lines are replaced by the data held in the queue. This method is called by both the up and down arrow keys, and is used to preserve data between sessions.