CIS 422 Project 1:
Ducks on Deck
Programmer Documentation

Ellie Kobak (erk), Kalyn Koyanagi (kek), Liza Richards (ljr), and Kelly Schombert (ks)

January 30, 2022 – v2.0

Table of Contents

1. Programmer Documentation Revision History

Date	Author	Description
1-23-2022	ljr	Created the initial document and wrote first draft
1-23-2022	ljr	Revised section 2
1-24-2022	ljr	Revised section 3, added section 4
1-26-2022	ljr	Made changes to sections 3 and 4
1-28-2022	ljr	Made changes to section 4

2. Introduction

The following sections of this document will explain all the working components of the DucksOnDeck software system. It will explain the different files and all of their respective components including global variables, functions, and how the different functions may interact with other files.

All code is written in python using PyCharm, sublime, and vim. This version of the cold-calling software was written for the 01/30/2022 release of the DucksOnDeck software.

This documentation assumes the reader has prior knowledge of the python programming language in a MacOS environment.

3. Working Component Outline

All the working python files are listed below with a brief description of what each file does and how it works together with other files:

1. fileReader.py: A file that prompts the user for input to ask what file they want to import. Stores the files data into a list to be sent to buildQueue.py.

2. buildQueue.py: A file that initializes/builds the queue from the data store in list by fileReader.py file. Each queue contains all of the student names and their corresponding information.

3. DucksOnDeck.py: This file is designed to control the user interface. When it runs, it displays the Ducks On Deck main menu which contains buttons for the user to choose from. It interacts with buildQueue to import the queue of students into display. It also interacts with fileWriter to record the collected cold-call data.

4. fileWriter.py: Interacts with fileReader.py by initializing a summary performance dictionary for a new roster of students. Interacts with buildQueue.py to update daily log summary and performance dictionaries. Interactions with DucksOnDeck.py to write a summary performance file. Overall job is to create and update a daily log file during usage, and create and update an overall term performance summary file.

4. Working Component Breakdown

This section will individually break down each part of the different working python files.

4.1 buildQueue.py

4.1.1. Queue Class

A class with the purpose of ensuring that the student data is in an organized queue data structure. It's initializer creates the queue from the fileReader.

4.1.2. randomize ()

Function with the purpose of randomizing the queue order to ensure that students are not picked again from the queue too quickly by calling the shuffle function from the random library.

4.1.3. __iter__()

Function with the purpose of making the queue iterable by calling the iter built in function.

4.1.4. _count()

Helper function with the purpose of obtaining the length of the queue which is also the length of the student roster.

4.1.5. onDeck()

Function with the purpose of returning the four top names from the queue to be displayed to the user in the visual window. It indexes into the queue and obtains the first and last names of each of the chosen students.

4.1.6. remove(name, flag)

Function with the purpose of removing student names from the On Deck visual window to be placed back into the back of the queue. It indexes into the queue at the place where the name is found and removes it. Then the student name is appended onto the back of the queue.

4.2 fileReader.py

4.2.1. getFileName()

This function asks the user for input regarding the name of the class roster tab delimited or txt file they want to import into the system. If an invalid file name is entered, the function will perform as an infinite loop and will continue to ask for a file name until a valid name is entered. If the user wants to exit this loop, the user must input "control + c"

4.2.2. filereader()

This function takes a tab-delimited csv or txt file as input and will read all of the file's information into the following format:

<first and last name> <UO ID number> <email>

4.3 fileWriter.py

4.3.1. Globals

a. today
   "today" is a variable of type string which contains the current date formatted as YYYY-MM-DD
b. dailyLog
   dailyLog is a variable of type string which contains the daily log file name and is formamted as YYYY-MM-DD_log.txt
c. timesCalled
   A dictionary tracking how many times each student is cold called
d. numFlags
   A dictionary tracking how many times a student is flagged
e. listDates
   A dictionary storing every date a student is cold called on

4.3.2. makeDailyLog()

The purpose of this function is to create and write to a daily log file with the date specified in its heading. It is called by updateLogs() if the daily log file has not yet been created.

4.3.3. updateLogs(flag: bool, name: str, email: str)

The purpose of this function is to update the daily log text file. If the daily log file does not yet exist, it will create one by calling makeDailyLog(), and then appends relevant student data that is passed to the function. It is called by buildQueue.py when a keystroke is pressed and records which action was taken.

4.3.4. exportSumPerf(roster: list)

The purpose of this function is to export the summary performance data collected during the system runtime. It asks the user for the summary performance file name, and writes the heading and student data to this summary performance file. This function is also called by DucksOnDeck.py when the user chooses to export summary performance data.

4.3.4. initSumPerf(roster: list)

The purpose of this function is to initialize summary performance dictionaries and corresponding csv files. It is called by the fileReader file with the import of a new roster of students.

4.3.4. updateSumPerf(flag: bool, name: str)

The purpose of this function is to update student data in the dictionary, and write new data to txt files. It is called by DucksOnDeck.py anytime that a keystroke is pressed and a student is cold called.

4.4 DucksOnDeck.py

4.4.1. GraphicalUserInterface Class

Class that holds all the user interface functions. Displays the different visual windows when prompted, keeps track of students in the on deck window. Interacts with buildQueue by using the student data from the queue structure. Keystrokes modify the exported performance summary files by interacting with the fileWriter.

4.4.2. RightKeystroke(event)

Changes made if the right keyboard key is pressed, moves to the right between students in the On Deck list.

4.4.3. LeftKeystroke(event)

Changes made if the left keyboard key is pressed, moves to the left between students in the On Deck list.

4.4.4. UpperKeystroke(event)

Changes made if the up keyboard key is pressed. Flags and removes the student from the On Deck list.

4.4.5. LowerKeystroke(event)

Changes made if the down keyboard key is pressed. Removes students from the On Deck list.

4.4.6. toggleHighlight(highlight_index, old_pos)

Changes the highlight between names as the user moves through the student names using the left and right keystrokes.

4.4.7. presView()

The purpose of this function is to create the On Deck window with four randomly chosen student names from the queue. This window responds to user keystrokes.

4.4.8. ondeck()

The purpose of this function is to create the list of four student names from the queue that is to be displayed in the On Deck window.

4.4.9. UserExportTerm

This function’s purpose is to export the summary performance file with recorded cold-call data using the exportSumPerf function from fileWriter.py

4.4.10. MenuDisplay

Displays the DucksOnDeck main menu. It establishes the window size and has three buttons which are “Export Performance Summary”, “Exit” and “New Session”. Each button correlates to a new action.

4.4.11. exit()

The purpose of this function is to exit the program when the exit button is selected by the user on the Ducks On Deck main menu window.

4.4.12. UpdateDeck(current_ducks)

Once a student name is removed, this function adds a new student from the queue into the On Deck list.

5. Known Bugs

5.1 Exporting Performance Summary Bug

There is a bug present when the user chooses to export the performance summary log, gives the desired log file name, and types “ctrl + d” to submit it. After pressing enter to submit this line, the program runs in an infinite loop and the user will not be able to access the command line. In order to work around this bug, the user must return to the Ducks On Deck main menu display window and select the “Exit” button. This button will terminate the system and the user will gain access back to the terminal command line.

5.2 Exporting Performance Summary Overwriting Bug

There is a bug present where the exported files are being overwritten as opposed to expanded upon. The original intention of our design was for when the user is prompted to input the name of the file for the export, if the same export file name was given as a file that already exists, then the old file would be updated/expanded upon using the new sessions cold-calling data. For example, the previous session’s “Total times called” and “Number of flags” values for each student would be expanded upon by adding these values from the new sessions data. To fix this bug, we believe that the acceptable imported file format should be modified to have the same structure and format that the exported file is designed to have. This means that the imported file should include the two columns “Total times called” and “Number of flags” that are present in the exported files, except the values should all be initialized to zero.