Programming Assessed Exercise 3

**Paul McHard – 2085227M**

**Final State of Program**

The program provides the full functionality required by the exercise specification, it has been thoroughly tested and I believe it adheres precisely to the specification required and fully satisfies what is asked for in this exercise.

**Assumptions**

* It was assumed during development that the main GUI provided in the set up of the program was sufficient to complete the task and no further work to this interface was required or expected.
* It was assumed that use of ArrayLists was acceptable as an intermediary step in creating or altering the main Fitness Programme array, which I deemed necessary to provide proper program functionality.
* It was assumed that if a fitness class found in the original ClassesIn.txt file and was then deleted and re-added using the functions of the GUI, then the new version of this class, despite having matching ID, name and tutor, would have a blank attendance record, and the program was not expected to re-obtain attendance information under such circumstances.

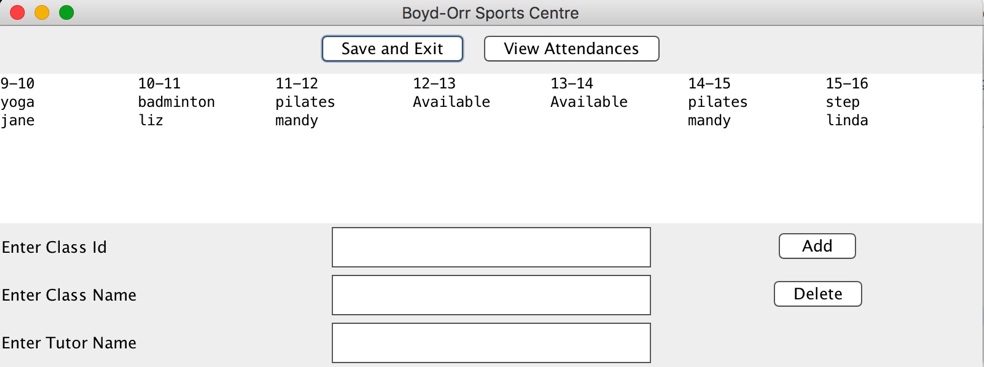
**Deficiencies**

* The specification recommends that method(s) for populating attendance lists are handled within the FitnessProgram Class. In developing the program I found it more convenient to handle this function between the GUI and ReportFrame Classes. While the functionality is delivered, this is not through the process advised by the specification, and I can appreciate how this could be considered a deficiency.
* Secondly, one deficiency that has been noted on program review is that I chose to perform sort methods by writing bubble sort methods, rather than using *Arrays.sort*, as recommended in the specification, which I recognize is a potentially faster solution.
* The code is otherwise as per the specification, and is properly formatted and well commented.

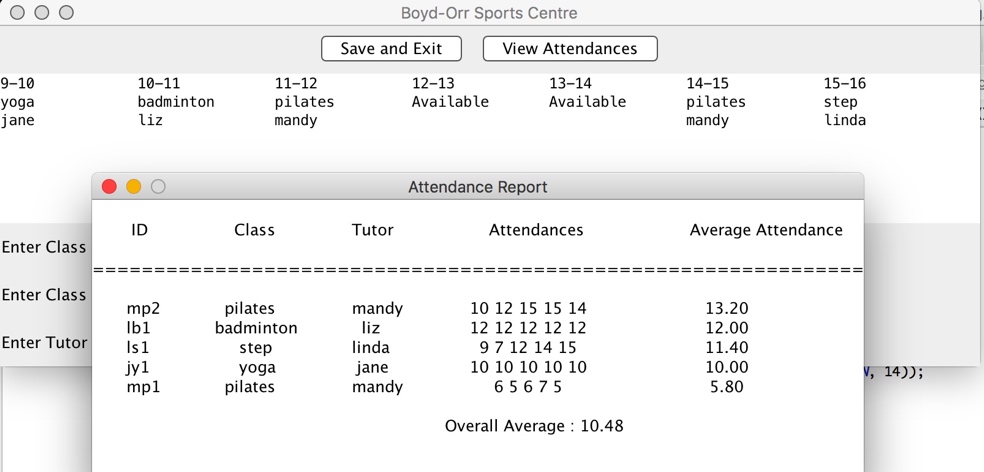
**Test Data**

|  |  |
| --- | --- |
| *Test* | *Proof* |
| 1.Program correctly displays timetable based on input data from unaltered classesIn.txt | Image 1 |
| 2.Program correctly creates an attendance report based on unaltered classesIn.txt and attendancesIn.txt | Image 2 |
| 3.Program successfully outputs classesOut.txt based on unaltered classesIn.txt | Image 3 |
| 4.Program successfully adds a new class into the Fitness Programme at next available time.  **Input:** *ID:* sp1, *Name:* swimming, *Tutor:* paul | Image 4, 5 |
| 5.Program correctly handles invalid input for adding new class (repeated ID: mp1). | Image 6 |
| 6.Program correctly handles invalid input for adding new class (blank input boxes) | Image 7 |
| 7.Program successfully deletes a class from the Fitness Programme. **Input:** *ID: mp1* | Image 8,9 |
| 8.Program correctly handles invalid input for deleting a class (*ID:* ab0 does not exist) | Image 10 |
| 9.Program correctly handles invalid input for deleting a class (*ID* field left empty) | Image 11 |
| 10.Program successfully creates an attendance report after Fitness Programme has been altered to include addition of a class (**Input** as per *Test 4*) and deletion of a class (**Input** as per *Test 7*). | Image 12,13 |
| 11.Program successfully outputs classesOut.txt based on alterations to Fitness Programme as per inputs in *Test 4 & 7.* | Image 14 |
| 12.Program successfully prevents addition of another class once Programme is full (**Inputs:** *Class 1:* as per *Test 4, Class 2: ID:* gg1, *name:* golf*, tutor:* gary*, Class 3: ID:* sp2, *name:* swimming, *tutor:* paul) | Image 15 |

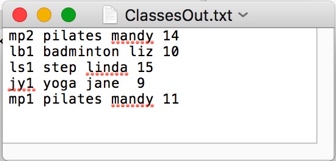
**Screenshots**

**

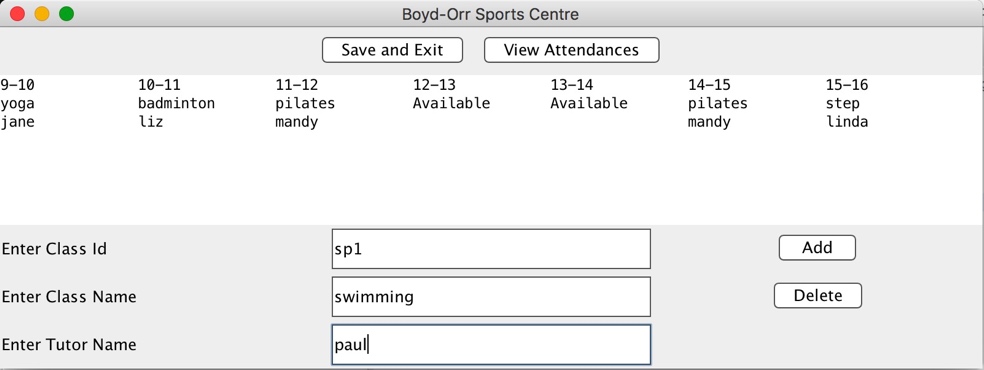
*Figure 1*

**

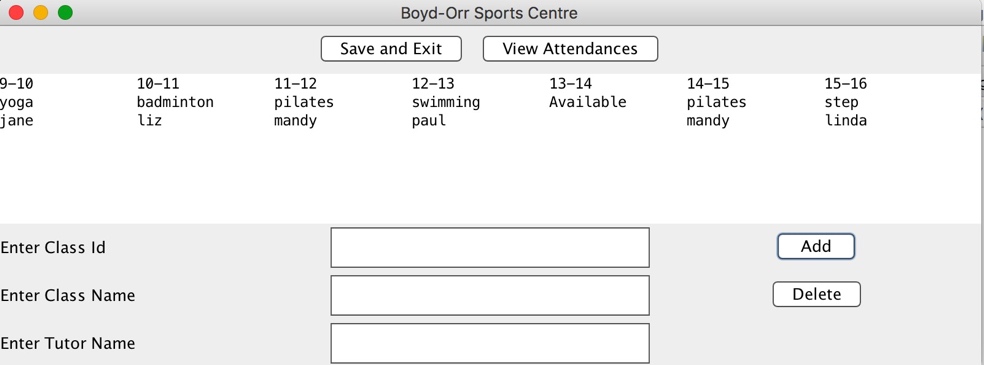
*Figure 2*

**

*Figure 3*

**

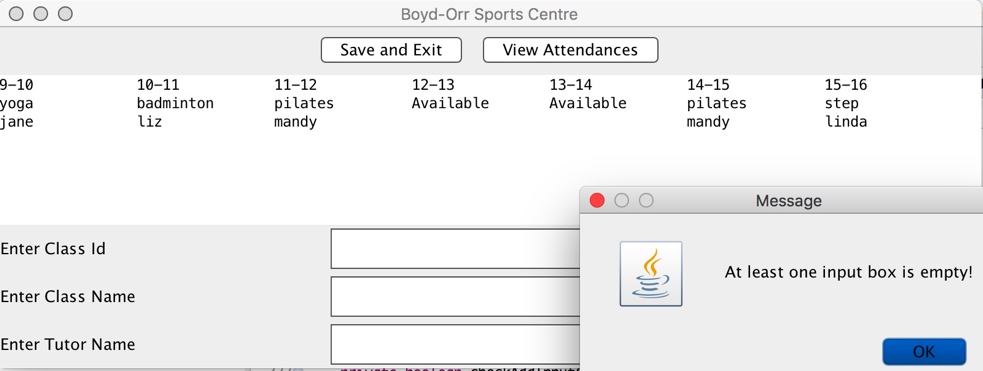
*Figure 4*

**

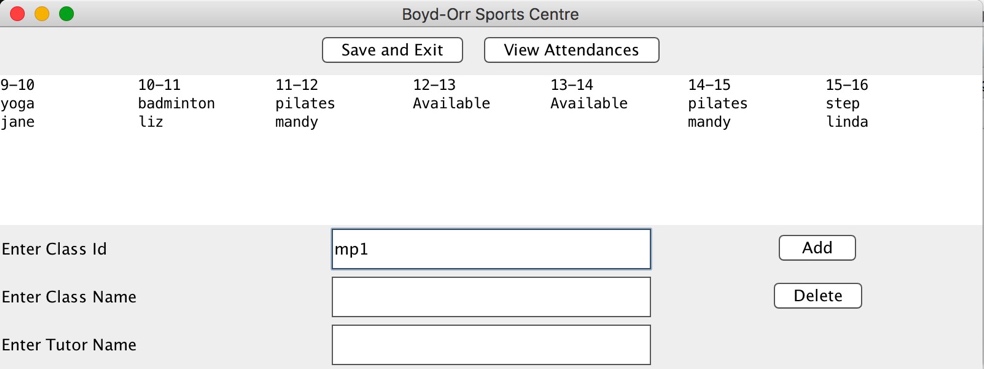
*Figure 5*

**

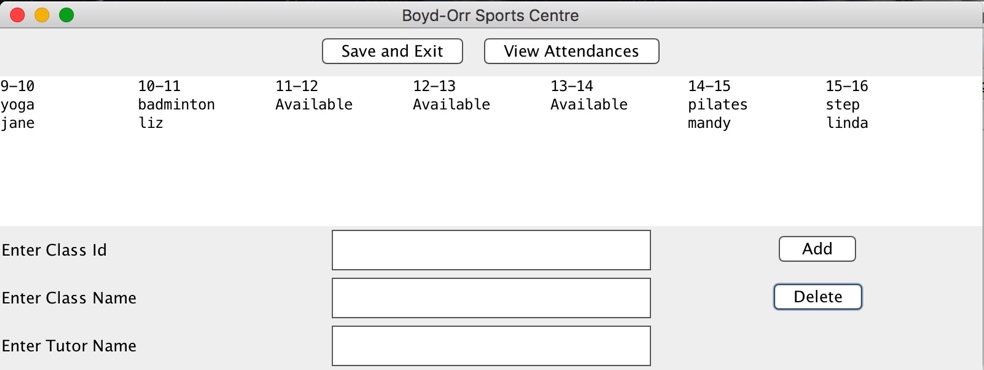
*Figure 6*

**

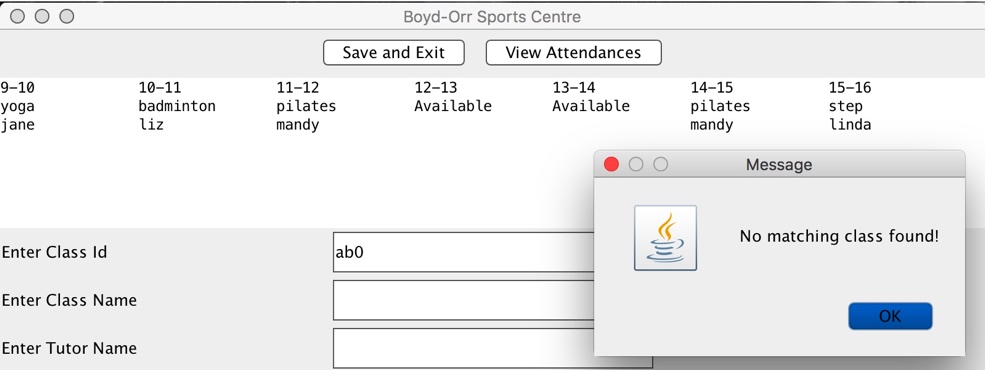
*Figure 7*

**

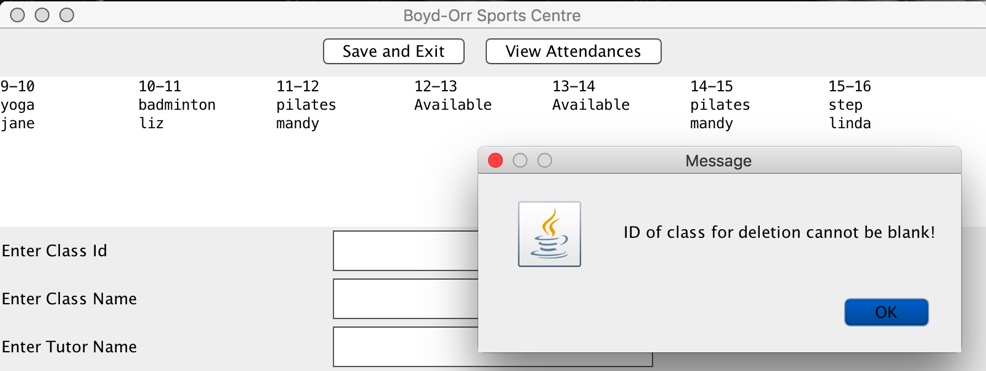
*Figure 8*

**

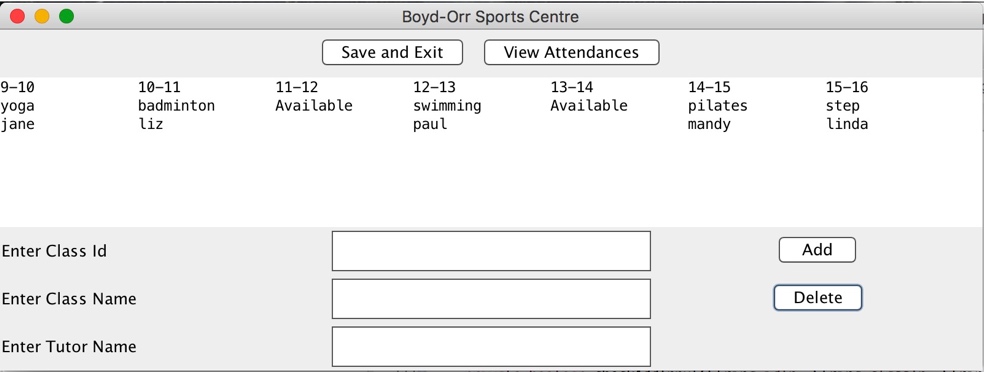
*Figure 9*

**

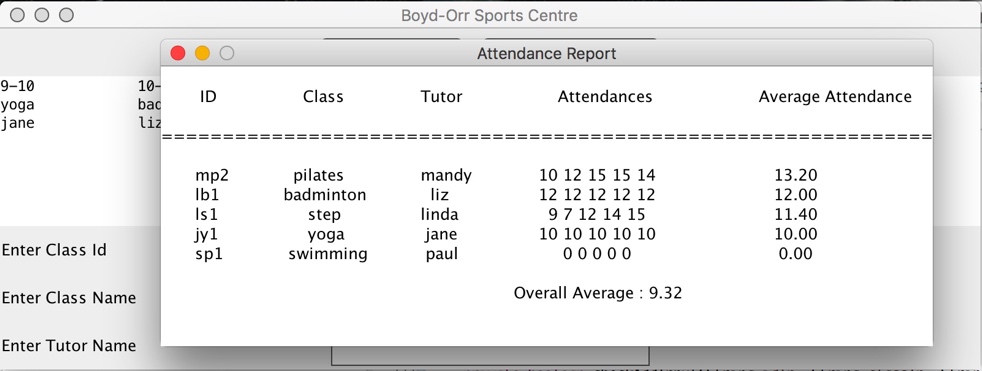
*Figure 10*

**

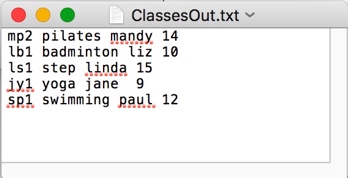
*Figure 11*

**

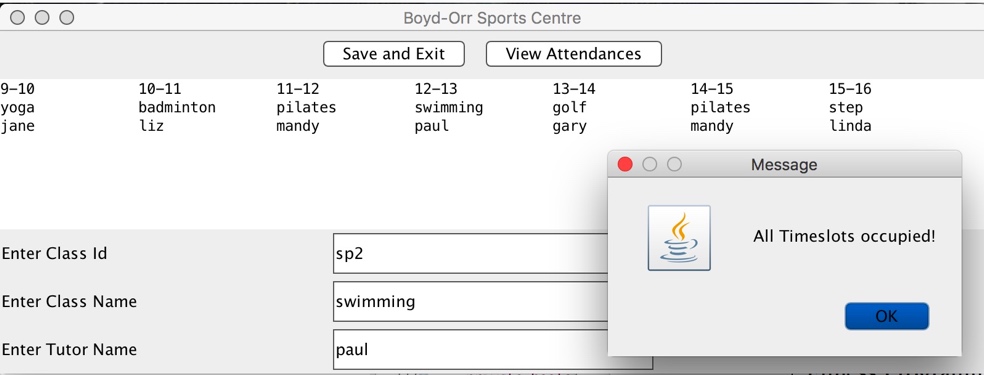
*Figure 12*

**

*Figure 13*

**

*Figure 14*

**

*Figure 15*