Assignment

CSCU9T4

Managing information

Read me

Designing and creating a

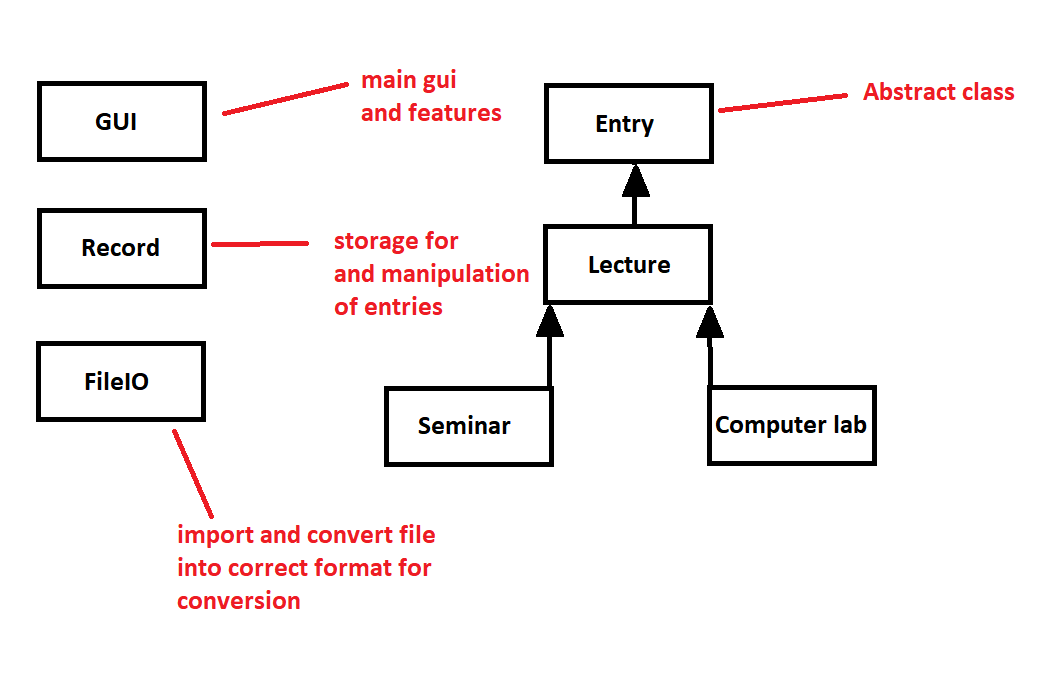
Timetabling program

Student ID 2628030

Spw00004

07/03/2019

For this assignment I created the following object model:



By using the above model I handled most GUI related issues and parsing in the GUI class, which is the main class of this project. The record class as shown above is used in searches, adding removing and counting of all the entries that are stored in the program. I stored all of the entries as an arraylist over a linked list as although we are able to remove entries from the list the main feature of this program was to search for and then output data from a text file imported at the program start. For this reason, the arraylist offers a better performance of O(1) compared to O(N) in the linked list. The above object model is constructed in the following way with entry as the main abstract superclass, relating to all methods that are to appear in lecture as specified in the assignment outline. Then lecture is fleshed out, seminar and computer lab inherit from lecture as they use the same main features with some extras that are unique to them. Therefore they both inherit from lecture but not from each other. All the limits for class size versus room size are outlined separately in each as is the output given depending on which record type you access. With the entries I split them into three groups lecture, seminar and computer lab as it seems that this is the type of handling expected in the assignment outline. All are then added to the same arraylist.

The fileIO class is used in its entirety to start formatting the input file for our use later. Once fileIO is done the file is sent to importOutput in the GUI class, where the file is parsed. Note: Since I have added an import tab, I made sure that either the simple or complex file can both be formatted correctly here, also I saved a string of the name of the file imported so that the user cannot attempt to import the same file again, furthermore if the user does import a file here, the original data is cleared with the removeAll function in record then the new data is imported. In the first tab as the user opens the program suitable information is outputted in the text box telling the user that the file is imported. Also should the user not enter a string at command line the program will then automatically import the complex file at start-up in the internal command line argument in the program. Effectively this means that as soon as the user enters all the entries of the complex file are ready to use.

There are three tabs in the program, import as previously talked about, search and export, and add/remove. Search and export provides all data fields for what is defined as as a unique entry, so that the user can find the data specified. I split the name into three parts, module code, ID (I.e seminar 1: 1 being the idea) and the number I.e (seminar 1: 10 where 10 is the number) all three of the fields can be used to uniquely search by name. For the search features I included a search by name exact, for precise searches of name including id and number, search by name partial which allows the user to simply enter U9 to get all classes, and search exact which requires all boxes defined as unique in the assignment outline to receive a more precise match. Both partial search and exact search are also included in the add/remove tab as are all the data fields. One additional data field in search/export is the html export limit, which allows the user to define how many entries maximum they wish to export. Export to file and export to html are of course both used here as buttons to allow the user to export the data shown in the text area bellow, also both output the name of the file created in the text area.

Finally the add/remove tab allow the user to add, remove or remove all. All of these features are as simple as they sound, add an entry remove one or remove all the entries in the arraylist. Personally I felt these features might be the most useful to any potential user. For the exporting part of the program there was specified a tab separated format or personalized timetable, as I couldn’t find the timetable html file that is referred to in the assignment outline I assumed that this meant either tab separated or your own style. For mine I outputted all the information in the same way that it appears in the text area so depending on how many records are shown this is the amount outputted, with the html file I also added a limit with the maximum value of 100. As specified in the outline the html should have a limit on how many records can be outputted.

For all of the functionality of the program I have included suitable notes so that any other programmer should be able to look through the code and understand what it is doing at any given point. I designed the program in a GUI style program as I thought that in this way it seems more comprehensive and allows more freedom to the user after the program has started. Even allowing the Import of different files similar to the simple and complex files we already have to work with.

All in all I think that the functionality of the program is robust in the fact that any blank boxes or incorrect data entry are handled with try catch exceptions, I have tested this rigorously and believe that the program should be completely without errors. Any errors that it may encountered should be handled by the try catch statements and output given to the user as to what is incorrect is sent to the text area relating to where the error may have occurred and how to fix it.