Timetabling

1. Introduction

This assignment is to exercise your skills in programming in Java to parse a file containing timetable information, to store the data in a suitable data structure, and to program a GUI to query that list.

Completion of this task requires a number of technical Java requirements:

- reading and parsing of a file with data in tab-separated format,
- creation and description of a suitable object model in which to store the data (using the information below),
- construction of the GUI to browse and amend the data (e.g. based on that from the practicals),
- writing an HTML file to allow the data to be displayed in a web browser.

Work which is submitted for assessment must be your own work. You are permitted to adapt the code you worked on for the Java practicals. The penalties for plagiarism can be severe. The University has a formal policy on plagiarism which can be found at:

http://www.stir.ac.uk/academicpolicy/handbook/assessmentincludingacademicmisconduct/

2. Part One: Input

The input is a tab-separated text file. Each line of the file has an entry for a scheduled class at the University of Stirling. There are two file formats: a simple one and a more complex one which you can attempt for a first class grade. All entries comprise:

name, day, start time, end time, week pattern, location, room size, class size, staff, department

In the simple version the name is just a module code plus class type descriptor. In the complex version there can be multiple module codes here. You can assume that all fields are variable length strings, enclosed in quotes. You should not store the quotes.

3. Part One: Object Modelling and Data Structures

Entries in the timetable can be categorised as lecture, computer lab, or seminar. Each category is associated with the same data as above, but you will be required to handle them differently in your program. You should formulate a suitable object model for this data, making use of appropriate (abstract) superclasses, and subclasses. (This is an exercise in inheritance, so although the timetable is rather simple, you should create an object model with inheritance.) You are expected to document your choice of object model in a short text file *README.txt.* The BlueJ environment will show the class diagram resulting from your object model, but you may hand write this and your documentation of the model and submit a scan if it's helpful. You should use a suitable data structure from the Collections framework to hold the timetable entries.

4. Part One: System Requirements

This part of the exercise is worth 85% of the overall grade for the assignment. Your goal is to create a Java program to allow the timetable to be browsed easily, and to manipulate the entries. The program should implement the following functionality:

Up to 40/85:

- Store the entries in a suitable object model as described above. You may consider simply hardwiring some initial entries for the timetable to allow you to test your program.
- Allow the user to browse the directory by searching for **module name.** The user inputs a text string and the program returns all entries matching that string. Partial matches are permitted (e.g. CSC will return all computing modules. The match need not be an initial match: that is, U9 returns all undergraduate modules). Matching should be case insensitive. The output should

be appropriately formatted for cases of zero, one, or more results.

- Display suitable information for each entry returned, depending on category. Although the information stored for each type of class is the same, you should present it as follows: lectures should appear in ALL CAPITALS with all data fields; computer labs should be in lower case with all data fields with a message if the class size exceeds the room size; seminars should be in lower case with a message if the class size either is less than half of the room size or more than 10% above room size. The object model you have created should take care of these differences.

Up to 55/85:

- Read the source directory as specified above. The name of the source file should be entered as a command line parameter. You may assume the file is correctly formatted (simple version).
- Allow new entries to be made. You should carry out suitable error checking to ensure that entries have the correct format. You should not allow duplicate entries. Entries are uniquely identified by name, day, start time, end time, week pattern.

Up to 85/85:

- Correct parsing of the more complex name format.
- Allow the user to save a timetable. For basic credit, save the whole timetable file in exactly the same format as given (tab separated text file). For extra credit save a selected portion of the timetable (as specified by name search) as an HTML file, using the same file name specified on the command line. You can be creative here: to display the output appropriately you will want to organise the data so that a weekly timetable can be presented. You may also limit the user to a certain number of entries.
- The system should be robust to errors in input, returning suitable messages for invalid search terms, and invalid file entries.

At all points you are expected to carry out suitable tests to ensure your code functions correctly but these are not part of the submission.

5. Part Two: Documentation

This part of the exercise is worth 15% of the overall grade for the assignment. You must document your process in a *README.txt* file that briefly describes:

1. A brief summary of the completed functionality of your program.
2. Your object model and the decisions you made in constructing this.

6. Submission

This assignment is worth 40% of the final marks for the module. As well as technical accuracy (e.g. correct reading and parsing of the input file, correct writing of output file, and correct implementation of the required functionality for searching), good programming style will also be taken into account (e.g. appropriate use of object oriented design, appropriate use of Java constructs, effective use of comment text, consistency, legibility and tidiness of program layout, suitably informative choices of variable and method names.)

You will need to submit your work on Canvas as a zipped file bearing your university username (3 letters + 5 digits, e.g., xyz00001.zip). This zip file should contain a folder also named after your username. Place all the files created during the assignment in this folder.