Lab Exercise Four - CIS*2430 (Fall 2021)

DUE DATES

Monday labs	*- Nov 15th by 11:59 pm*
Tuesday labs	*- Nov 16th by 11:59 pm*
Wednesday labs	*- Nov 10th by 11:59 pm*
Thursday labs	*- Nov 11th by 11:59 pm*
Friday labs	*- Nov 12th by 11:59 pm*

Extend the student class system you created in Lab Exercise Three with a HashMap and ensure that the program is robust by implementing proper exception handling.

- Add the lastName attribute to the Student class. This should be an additional input for the user to enter since it is **mandatory** for each student.
- Due to a forecasted increase in student enrolment, your program must handle the search more efficiently at a bigger scale. Instead of checking the ArrayList sequentially, we can create a HashMap index for all Students so that the search can be done more efficiently. To create a HashMap, we need to associate each Student with a unique key value, which consists of the program, the year, and the last name, all of which are concatenated into one string and normalized into lower cases.
- Exceptions are to be thrown in the constructor(s) of the Student class and caught in a method that calls these constructors. Please refer to the example in the lab slides for more information. To create a valid Student object, we should always provide the program, year, and last name. In addition, if other information is available, an average grade should be between 0 and 100 (inclusive). Any violations to these conditions should throw an exception in the related constructor(s).
- Every time we get the Student information from the user or from the input file, we should always check if the corresponding key value is already on the HashMap. If so, simply display a warning message and reject the input. Otherwise, create a Student object and add it to both the ArrayList and the HashMap. In other words, the HashMap is not only useful to speed up the search but also helpful for the duplicate check of the key values.

- The menu should now look like the following:

  (1) Enter information for a new Student.
  (2) Enter information for a new GraduateStudent
  (3) Show all student information with each attribute on a separate line
  (4) Print the average of the average grades for all students as well as the total number of students
  (5) Enter a specific program and show all student information for that program
  (6) Load student information from an input file.
  (7) Save all student information to an output file
  (8) Lookup via HashMap with program, year, and lastName.
  (9) End Program

Example Usage and output:

Program displays menu.

User selects option (6).

Please enter the name of the input file:

students.txt

*****Reads and parses file*****

Program displays menu.

The input file should be a text file with the following format:

Program Year avgGrade supervisor isPhD undergraduateSchool lastName

where program, supervisor, undergraduateSchool, and lastName are all made of single words and isPhD is either 1 or 0

Examples:

CompSci 4 85.4 Song 1 Guelph Smith

Psych 2 92.1 Brown

Biology 1 75.0 Wineberg 0 McGill MacDonald

You can assume that the input file is perfectly formatted since it is likely saved previously by your program. However, you cannot assume that the input file will exist, which must be error checked.

MARKING RUBRIC (10 marks in total)

3 Marks for HashMap search in option (8).
3 Marks for HashMap creation/update and duplicate check in options (1), (2) and (6).
3 Marks for exception handling in options (1), (2), and (6).
1 Mark for the command loop, and options (3), (4), (5), (7), and (9).