CSCI 2540 Assignment 6

100 points

Due date: Tuesday, Oct. 25 by 11:59pm

In this assignment, you will implement an application that manages the information of a list of students. You will need to use **ArrayList** (not array) to store the list of students.

Information about a student includes id, name, standing (such as junior, senior, etc.), and major. The roster of students is store in a text file **before** and **after** the program is executed. Use "**assg6_data.txt**" for the file name.

You will need to write the code for one interface (**StudentListInterface**) and three classes (**Student**, **StudentList**, and **StudentMIS**), for which the details are given below.

Interface

- **StudentListInterface** - this interface should include the following methods:
  - *loadData* – this method should have a file name as the parameter. The method loads the data containing all the students from a given file.
  - *displayRoster* – this method displays the complete information of all the students on the roster. It does not have any parameter and returns nothing.
  - *searchForStudent* – this method should have an id (of String type) as the parameter. It should return the Student object if found, or null if not found.
  - *addStudent* – this method is used to add a new Student. It should have four parameters that represent the id, name, standing, and major. If the id is already in the student roster, then a message should be printed informing the user that the student already exists. This method returns a boolean value. If the student is added, it returns true; otherwise it returns false.
  - *removeStudent* – this method should have an id (of String type) as the parameter. It should remove the Student from the roster if the id is found. Otherwise it should print a message. This method returns a boolean value. If the student is removed, it returns true; otherwise it returns false.
  - *getStudentsByMajor* – this method should have a major as the parameter. It should return an ArrayList object with all the students with the given major. If there is no student with the given major, the size of the ArrayList will be zero.
  - *Sort* – this method has no parameter and returns nothing. It should sort all the students by their id.

  - *Save* – this method has no parameter. It should write the student roster back to the file if the data has been changed (the same file is used for both reading and writing).

Please include detailed comments (using Javadoc commenting) for all the methods in the interface (as well as in the class that implements the interface).

Classes

- **StudentMIS** - The class with the main method. You will need to read the data from input file, display menu options, and perform various tasks that user selects. You will need to create an object of the StudentList class that represents a roster.

- **StudentList** - The class for a list of students. The information of the students must be stored in an **ArrayList** object. You will need to write the code for all the methods specified in the **StudentListInterface**.

  You may add additional methods as needed to this class, however ALL additional methods you add MUST be declared as **private.**

- **Student** - The class containing all of the information and methods for one student.
  - **The information to be stored includes id, name, standing, and major.** You should include one constructor with four parameters with given information for a student.
  - You should include get method for each data field.
  - You should include set method for major. Other fields cannot be changed.
  - You need to include the *toString* method.
  - You need to include the *equals* method. This method should have one parameter of Object type. Two student objects are considered equal if they have the same id.
  - You need to include the *compareTo* method. This method should have one parameter of Student type. The order of the students is based on the order of the id. In order to have this method, **the Student class should implement the Comparable<Student> interface**.

Program Requirements

- As your program begins, it should read previously saved data from a file. The name of the file should be **assg6_data.txt**. Do not change the file name! The data in the file should be in the following format: one student per line and the attributes are separated by "," (see the sample input file at the end of this document). A sample input file is also posted on Canvas.

- The program should then display a menu of options to the user and allow the user to make choices from the menu. This continues until the user selects the exit option from the menu. Upon exiting, the program should save the student roster back to the file **if any changes (such as add, remove, or change of order) has been made**.

Menu Options

The program should have a menu with the following options:

1. Display the roster
2. Search for a student by id
3. Add a new student
4. Remove a student
5. Search for students by major
6. Sort and save to file
7. Save to file
8. Exit

After the results are displayed for a given selection, ask user to press Enter to continue.

For the menu “Display the roster”, display the complete information of all the students, in the same format as in the input file.

For the menu “Search for a student by id”, display the complete information of the student with the given id. If not found, display a message.

For the menu “Add a new student”, first ask user to enter an id. If the id is already in the roster, then display a message and no new student will be added. Otherwise, ask user to enter the rest of the information and add the student to the roster.

For the menu “Remove a student”, first ask user to enter an id. If the id is found, the student with the given id is removed. Otherwise display a message.

For the menu “Search for students by major”, display the id and name of all the students (if any) by the given major. If there are no such student, display a message.

For the menu “Sort and save to file”, you can simply call **Collections.sort** method to sort all the students in the list. Make sure you implement *compareTo* method correctly in the Student class.

For the “Save to file” menu, if any change has been made to the roster, it will save the updated roster to the file.

For the “Exit” menu, upon exiting, the program should check if any changes have been made. If so, the modified data should be saved and written back to the file.

Sample Input File:

```
100126, John Smith, junior, CS
265812, Emily Washington, freshman, Math
145228, James Park, senior, Engineering
526045, Sandra King, senior, CS
366010, Mark Baker, junior, Math
166908, Adam Wong, freshman, CS
```

Please add more data when you test your programs. Your program should also be able to run correctly if there are blank line(s) at the end of the input file. You may use the **StringTokenizer** class to parse the input data or the **split** method in the String class.

Javadoc comments are required for this assignment and all future assignments.

Submission instructions:

To submit your programs, you need to submit your programs electronically on Canvas.

Please use a named package for each of your assignment. For example, for assignment 6, create a new package and name your package as assg6_yourPirateId (use lower case for your pirate id), such as assg6_smithj19. You also need to include a statement such as "package assg6_smithj19;" at the beginning of each of your .java file. Please follow this naming convention exactly for all future assignments. You will be deducted points for not doing so.

When you submit your files to Canvas, please submit a zip file with your package folder inside the zip file. The package folder should include only .java files (make sure you include .java files, not .class files). The name of the folder should match with your package name. You do not need to include input file in your package.