CSE 271 Project 5 – GUIs, Sorting, and Searching
Spring 2022
Assigned: 4/14/2022
Due: 4/29/2022

Introduction:

In this project, we are going to practice utilizing sorting and searching algorithms in conjunction with a Graphical User Interface (GUI). Your task is to develop a program with the GUI shown in the example figure below. Specifically, you will be developing two classes: a class **Student.java** for storing student records, and a class **StudentRecord.java** for GUI functionality. The program reads student records from a file, sorts them according to student grades, and finds students by a student ID. Note for projects 1-5, you must submit work to Canvas once during the first 7 days of when the project was assigned to show progress towards completion (please see the rubric).

Class Student:

Create a class **Student.java** with private instance properties that have appropriate data types: **id**, **firstName**, **LastName**, and **grade**. This class implements the interface **Comparable<Student>**. It also contains the following public interface:

- **Workhorse constructor**
- **public int compareTo(Student s) –** This method compares the grade instance property only. You do **not** need to check the other instance properties.
- **public boolean equals(Object o) –** This method compares the id instance property only. You do **not** need to check the other instance properties.
- **public static Student[] readFromFile(String filePath) –** This method reads the data from a file and returns an array of student records. Note that it is easier to deal with an ArrayList while reading from a file. After reading all records, convert the ArrayList to an array and return it.
- **public String toString() –** This method returns a String in the following format: "id, firstName, lastName, grade", e.g., "1, Julita, Annell, 80".
- **public static void sort(Student[] students) –** Implement any sorting algorithm that you prefer except Arrays.sort(). The Students are sorted by grade from lowest to highest grade. Note that here you are going to use the **compareTo()** method to compare the objects instead of the "==" operator.
- **public static int search(Student[] students, Student s) –** Implement the linear or binary search algorithm to find if an object is present in the array. Use the **equals()** method instead of the "==" operator.

Class StudentRecord:

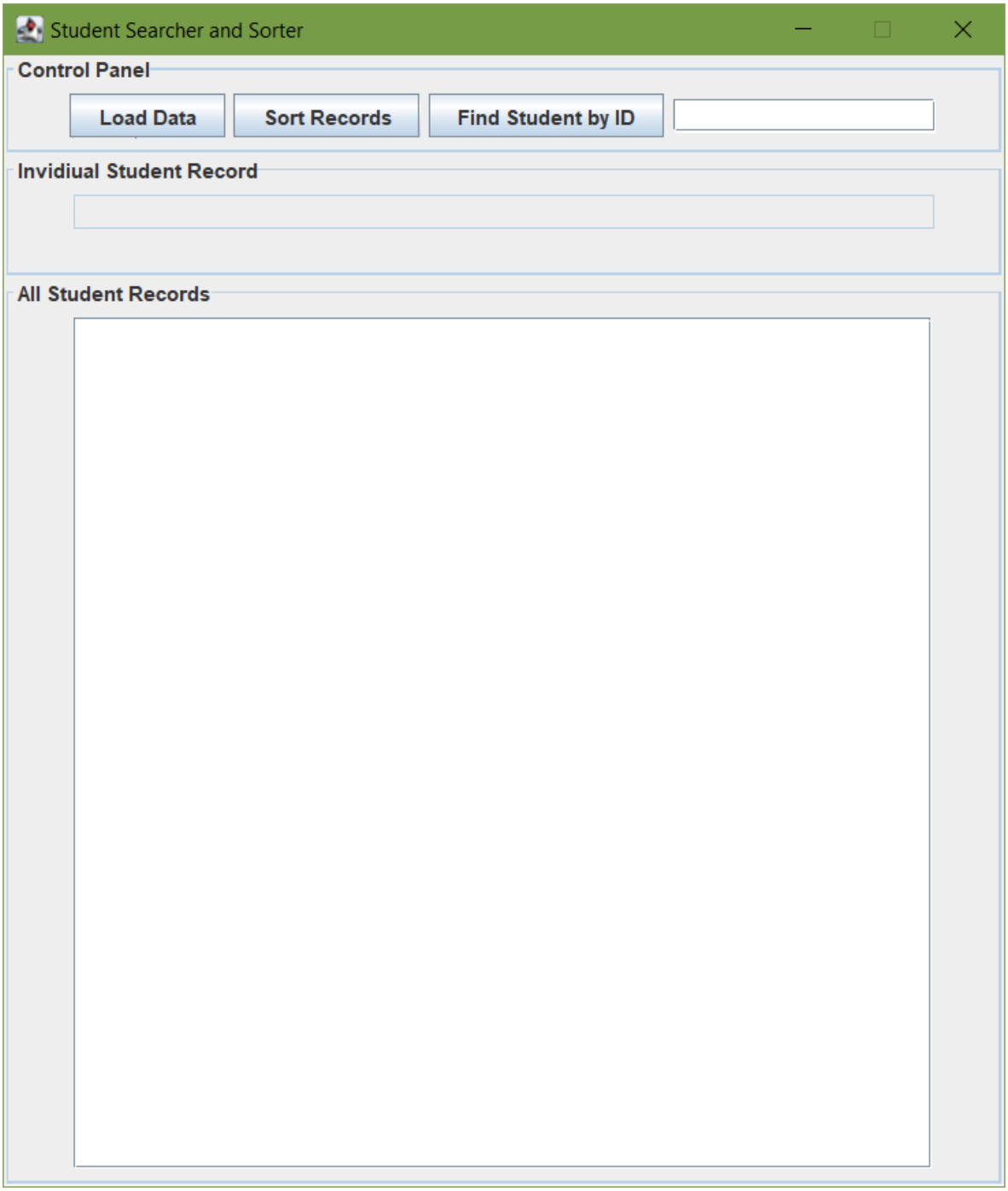
Create a class **StudentRecord.java** that inherits JFrame and contains the main() method. It contains an instance property named *studentArray* that stores an array of Student records. The frame consists of three JPanels as described below and has a suggested size of 600 pixels wide and 700 pixels tall. All the panels should have a titled border as well as shown in the example figure.

- The upper JPanel contains three JButton objects and one JTextField object (with column width of 15). The text field is used for the user to input the ID of a student.

- The middle JPanel contains one JTextField (with column width of 50). Make this JTextFiled uneditable. This text field displays individual student records when a user clicks the "Find Student by ID" button.
- The lower JPanel contains one JTextArea (with 31 rows and 50 columns). Make the JTextArea uneditable. The text area displays the loaded student records. Each student record is displayed on a new line. Finally, make sure the JTextArea is contained within a JScrollPane.

The functionality of the buttons are as follows:

- **Load Data:** Upon clicking this button, the program loads the data from MOCK_DATA.csv (which you will download from the corresponding Canvas project page) and stores the data of Students in the *studentArray* array. Display the content of the array in the JTextArea of the lower JPanel. To accomplish this, use the **readFromFile()** method of the **Student.java** class. You should write a helper method called **private void displayAllRecords()** that displays all the student records stored in the *studentArray* instance property. The helper method should use the **toString()** method of the **Student.java** class to display individual student records.
- **Sort Records:** Upon clicking this button, the *studentArray* array is sorted using the **sort()** method of the **Student.java** class. Then, the JTextArea of the lower panel is cleared and displays the sorted array using the **displayAllRecords()** method.
- **Find Student by ID:** The user enters the Student ID in the JTextFIeld of the upper JPanel and then clicks this button. When clicked, the program should find the student record in the *studentArray* using the **search()** method of the **Student.java** class. The entire student record, along with the index of the object in the array, should be printed to the middle JPanel's JTextField. You should use the **toString()** method of the **Student.java** class to display the record, e.g., "1, Julita, Annell, 80, Index = 1".



JavaDoc Style Comments:

You are required to make JavaDoc comments for all classes and methods including parameters and return descriptions.

Important Note:

Please make sure the file names are correct and the code is well commented!

Submission Instructions:

Submit your .java files (**Student.java and StudentRecord.java**) zipped to the corresponding Project 5 Canvas folder.

Rubric:

Task	Grade
Student	
Correctly implemented the workhorse constructor	4
Correctly implemented the compareTo() method	10
Correctly implemented the equals() method	5
Correctly implemented the readFromFile() method	10
Correctly implemented the toString() method	5
Correctly implemented the sort() method	8
Correctly implemented the search() method	8
StudentRecord	
The Load Data button is implemented correctly	10
The Sort Records button is implemented correctly	10
The Find Student by ID button is implemented correctly	10
The GUI matches the example GUI given in the project manual	10
JavaDoc is present for all methods and both classes and followed the Miami University coding guidelines	5
Provided an intermediate submission with noticeable progress.	5
Total	**100**