Project Report:

Portfolio Project – Students Manager

Alejandro Ricciardi

Colorado State University Global

CSC372: Programming 2

Professor: Dr. Vanessa Cooper

August 4, 2024

Contents

Project Report:

Portfolio Project – Students Manager

This documentation is part of the Portfolio Project Assignment from CSC372: Programming 2 at Colorado State University Global. This Project Report is an overview of the assignment and the Students Manager program's functionality, testing scenarios, and outputs. The program is coded in Java JDK-21.

The Assignment Directions:

Option #1: **Lessons Learned and Final Program**

Milestones

Milestone 1 (due at the end of Week 4): Java source code (with corrections as required) for programs created in Modules 1-3.

Milestone 2 (due at the end of Week 7): Java source code (with corrections as required) for programs created in Modules 5-6.

Final Portfolio Project

In Week 8, the components you must complete for your Portfolio Project are the Lessons Learned Reflection and the Final Program. Carefully review the requirements below.

Lessons Learned Reflection

Write a summary outlining the lessons you have learned in this programming course. Reflect on how these lessons can be applied to effective coding.

This essay portion of your Portfolio must:

- Be 2-3 pages in length (not including the required title and references pages)
- Be formatted according to APA guidelines in the CSU Global Writing Center.
- Include at least three outside references from the course readings, formatted according to APA requirements.

Final Program

Write a Java program that incorporates a loop that prompts the user for student data. Student data are private fields in a student class including:

- String name
- String address
- double GPA

Each student object is stored in a linked list.

After the user completes the data entry, output the contents of the linked list in ascending sorted order by name to a regular text file that can be opened and viewed using a simple plain-text editor such as notepad.

Validate numeric data for Grade Point Average (GPA).

Compile your Lessons Learned Reflection, source code, screenshots of the application executing, and results into a single document.

Format your document in MS Word, according to APA guidelines in the CSU Global Writing Center, particularly in developing your Lessons Learned Reflection. Support your reflection with a minimum of three references, as noted above. Include both a cover page and a reference page with your Portfolio Project.

⚠️ **Program Notes:**

- I got permission from Dr. Cooper to use the JavaFX library to display the program outputs.
- Added my own icon to the window frame – logo.png.
- Added search functionality (not an assignment requirement).
- Added read file functionality (not an assignment requirement).
- Added the option to add fake data to the file for troubleshooting purposes (not an assignment requirement).
- Implemented my own selection sort and binary search algorithms.
- Created a UML class diagram.
- **For the source code please see the following: Student.java, StudentManager.java, NameComparator.java, GpaComparator.java, SortSearchUtil.java, StudentManagerApp.java**

Project Map:

- **Project Report.pdf**: A pdf file (this file) containing an overview of the assignment and the Students Manager program.
- **README.md**: A markdown file containing information about the project, intended to be viewed on GitHub.
- **Lessons Learned and Reflection.doc:** A Word document containing a summary and reflections on the lessons I have learned in this programming course.
- **Milestone-1:** Directory containing the Portfolio Milestone assignment from Module 4.
- **Milestone-2:** Directory containing the Portfolio Milestone assignment from Module 7.
- **Application:** A folder containing the source code, Java code files for the Students Manager program.

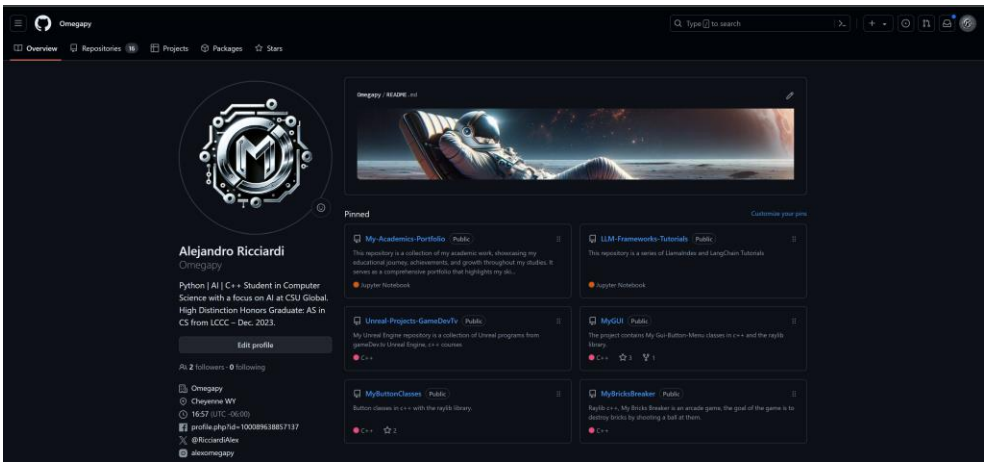
Program Description:

The Students Manager is a small Java application that utilizes JavaFX GUI allowing the user to add, view, search, and sort students data:

- Student data management (name, address, GPA)
- File-based storage
- Sorting by name or GPA
- Search functionality
- Basic data validation

Git Repository

This is a picture of my GitHub page:



I use GitHub as my Distributed Version Control System (DVCS), the following is a link to my GitHub, Omegapy.

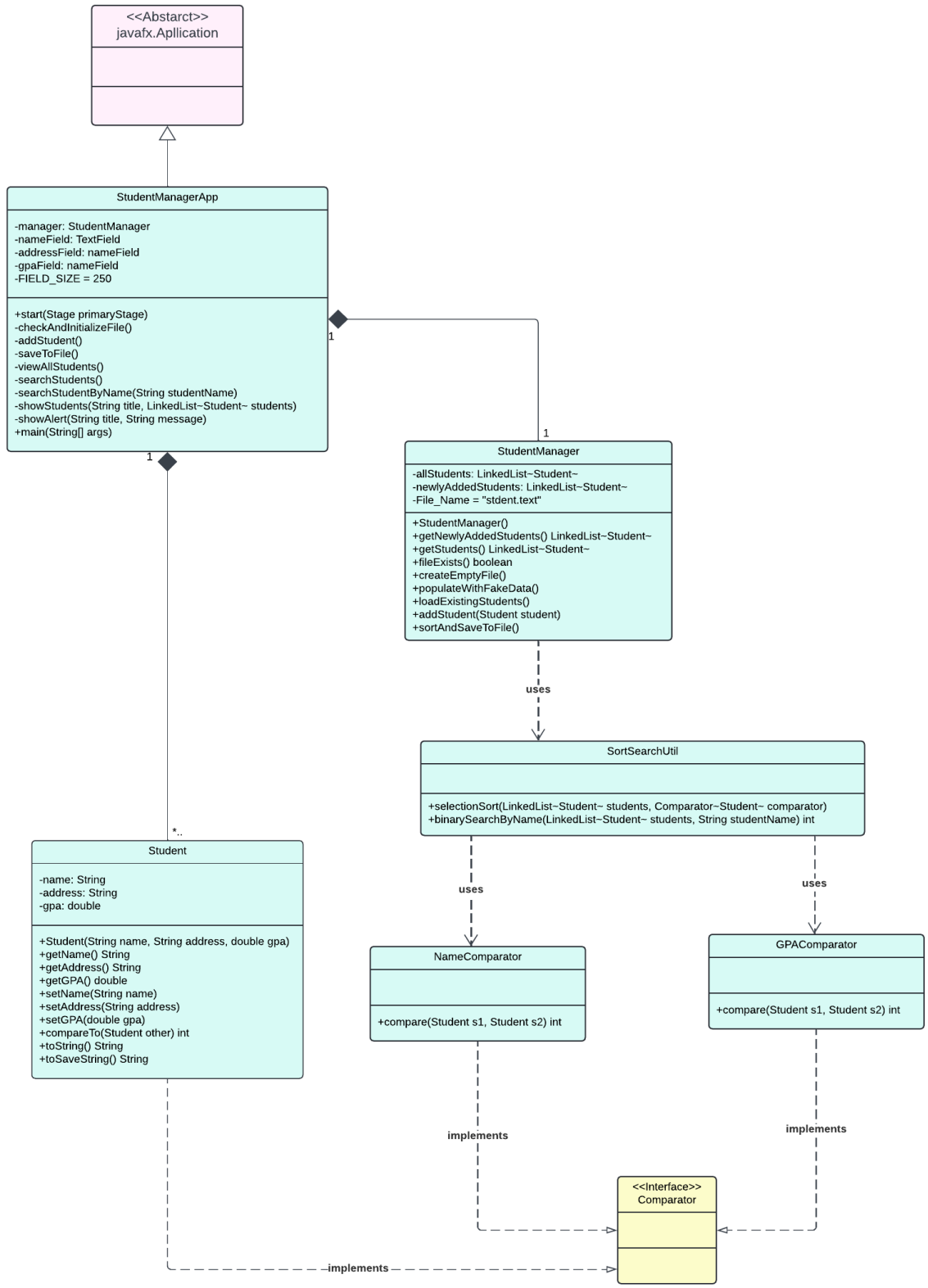
My GitHub repository that is used to store this assignment is named My-Academics-Portfolio and the link to this specific assignment is:

Classes Description:

- **Student Class**
  Creates student objects. This class stores student data and provides methods for data access and validation. It implements the Comparable interface to be used by the SortSearchUtil class to sort and search students
- **StudentManager class**
  Manages student objects and handles file operations. This class is responsible for adding, storing, and retrieving student data, as well as reading from and writing to a file.
- **NameComparator Class**
  Comparator for sorting Student objects based on their names. Implements an ascending order comparison (A to Z), used by the SortSearchUtil class. This class implements the Comparator interface.
- **GpaComparator class**
  Comparator for sorting Student objects based on their names. Implements a descending order comparison (highest to lowest GPA), used by the SortSearchUtil class. This class implements the Comparator interface.
- **SortSearchUtil class**
  Utility class for sorting and searching students. Provides methods for selection sort and binary search on a LinkedList of Student objects.

- **StudentManagerApp class**
  Main application class for the Student Manager program. This class creates the user interface (GUI).

UML Class Diagram

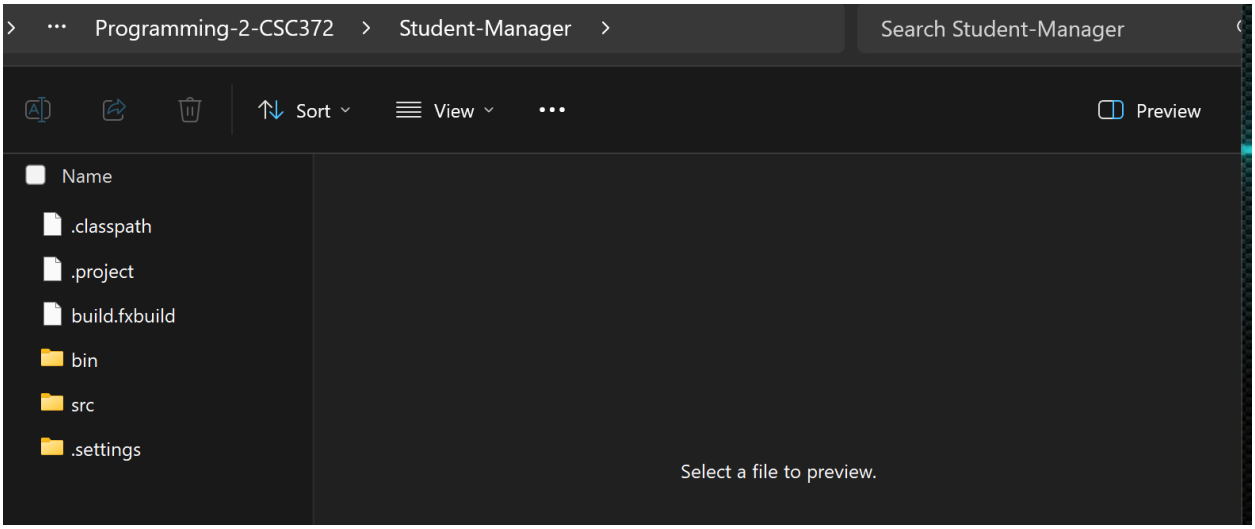


Screenshots

Program Functionality

Figure 1

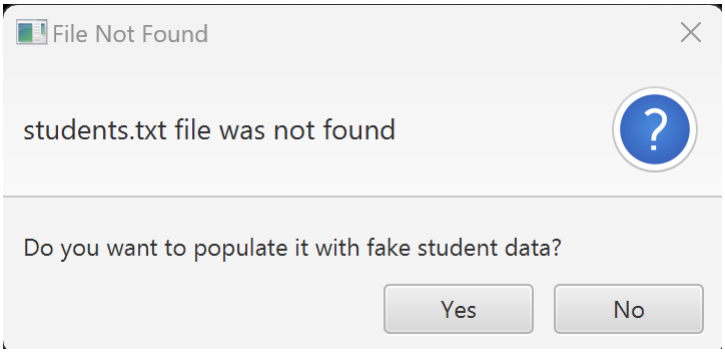
No File in Directory



Note: No students.txt is present in the directory

Figure 2

Initialization

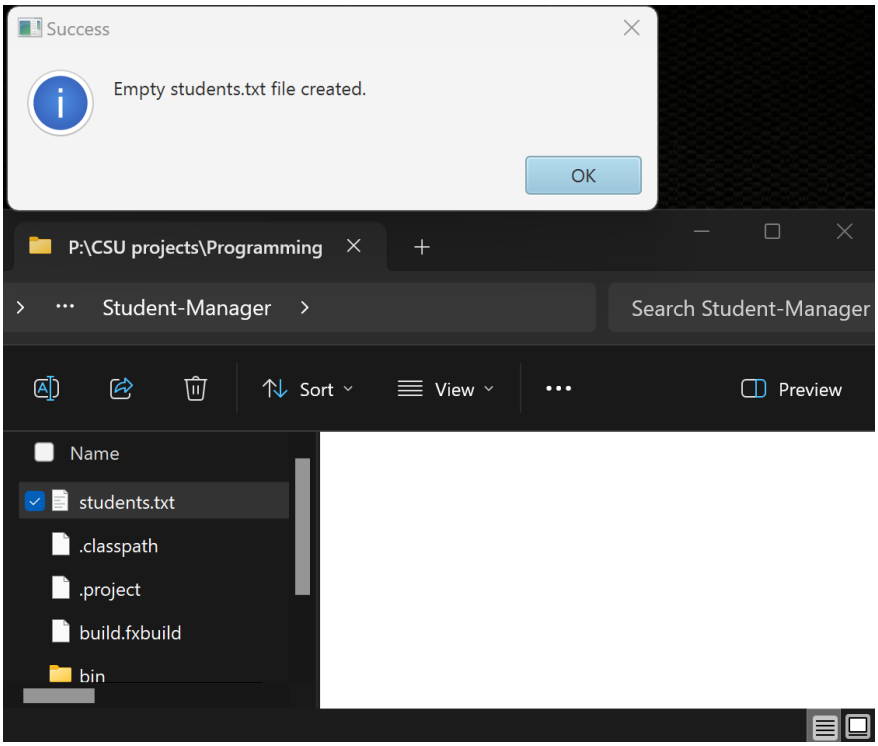


Note: If no students.txt file is found in the directory, the program will ask the user to choose if the user wants to populate the new text file with students' fake data.

More next page

Figure 3

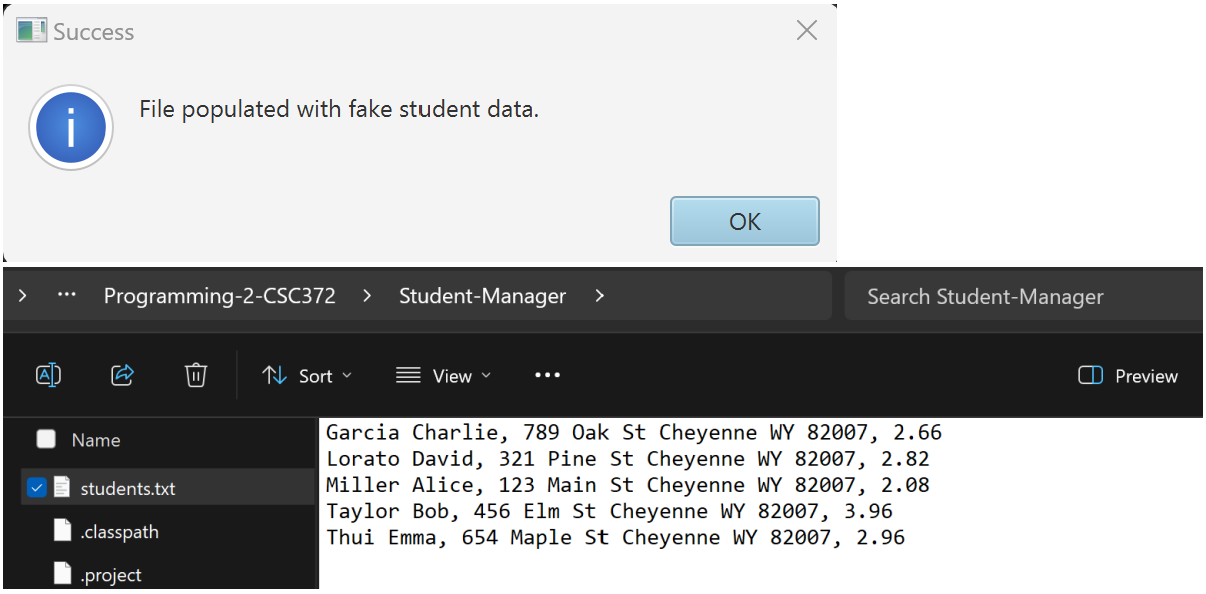
No Fake Student Data



Note: After the user clicked no to populate the text, the program created an empty students.txt file.

Figure 4

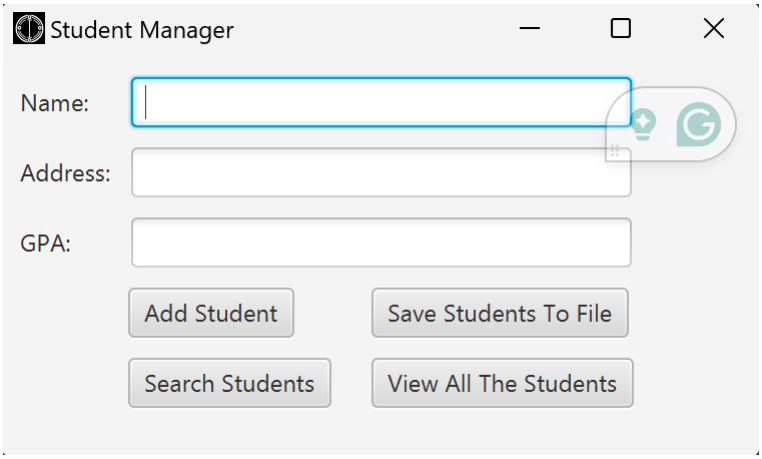
Fake Student Data



Note: After the user clicked yes to populate the text, the program created a students.txt file and populated it with fake student data. Note that the students are sorted by name.

Figure 5

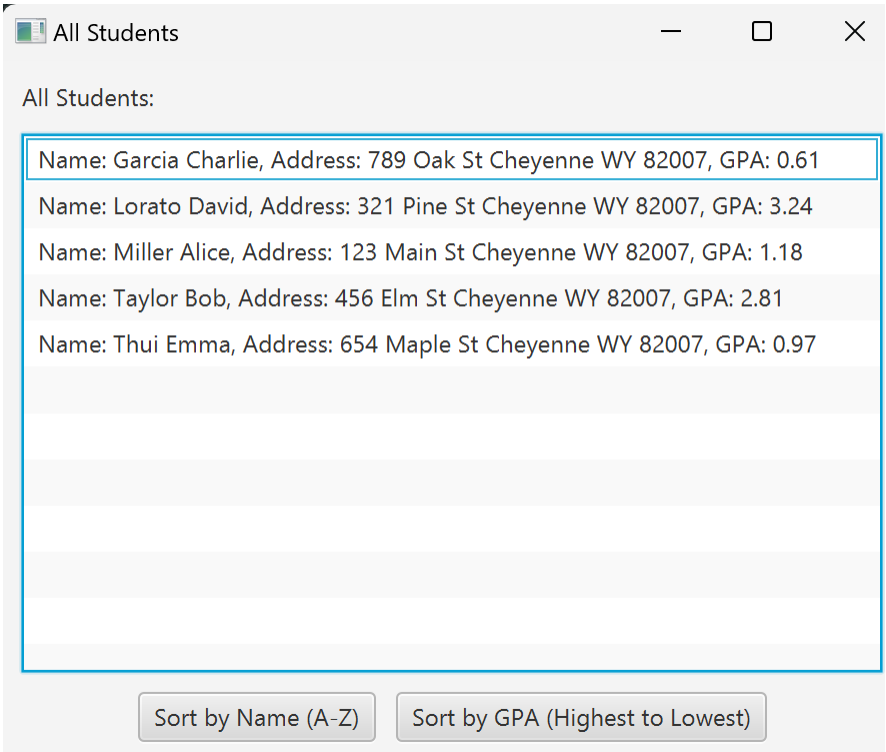
Welcome Window



Note: During the program initialization, if the file already exists in the directory, the program will not create a new file or it will directly the welcome window.

Figure 6

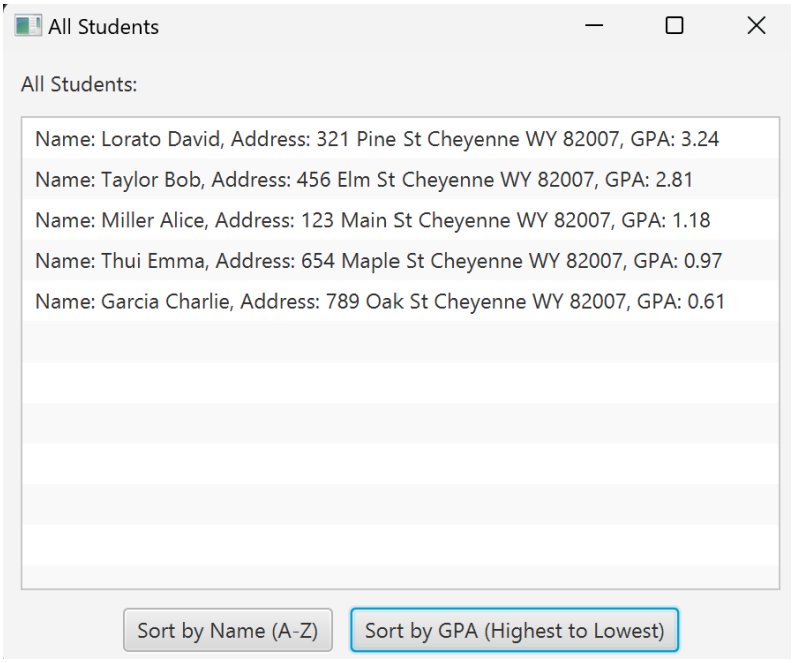
View All The Students



Note: After the user clicks “View All The Students,” the program displays all the students saved in the students.txt file. These students are from the program initialization fake data option. Note that they are sorted by name.

Figure 7

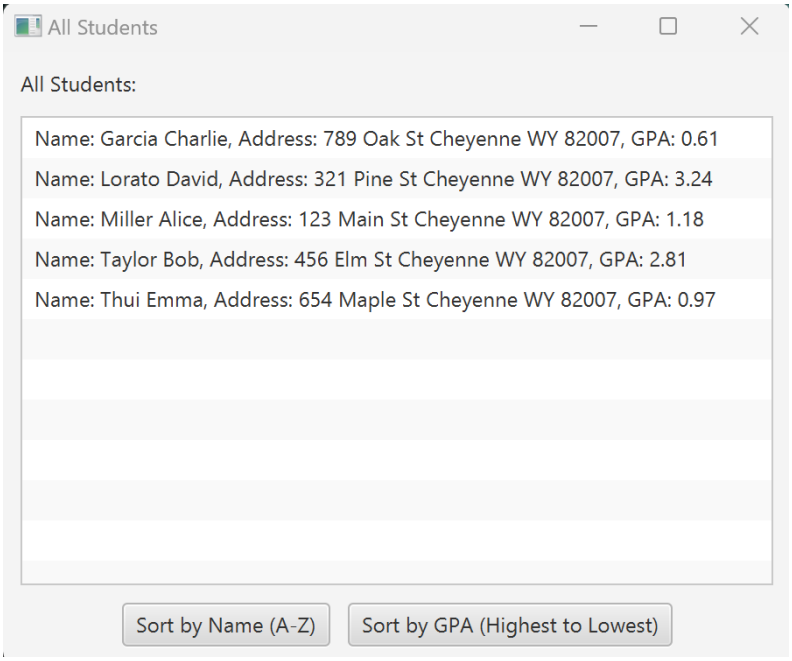
Sort By GPA



Note: After the user clicks "Sort by GPA (Highest to Lowest)," the program displays all the students saved in the students.txt sorted by GPA (Highest to Lowest).

Figure 8

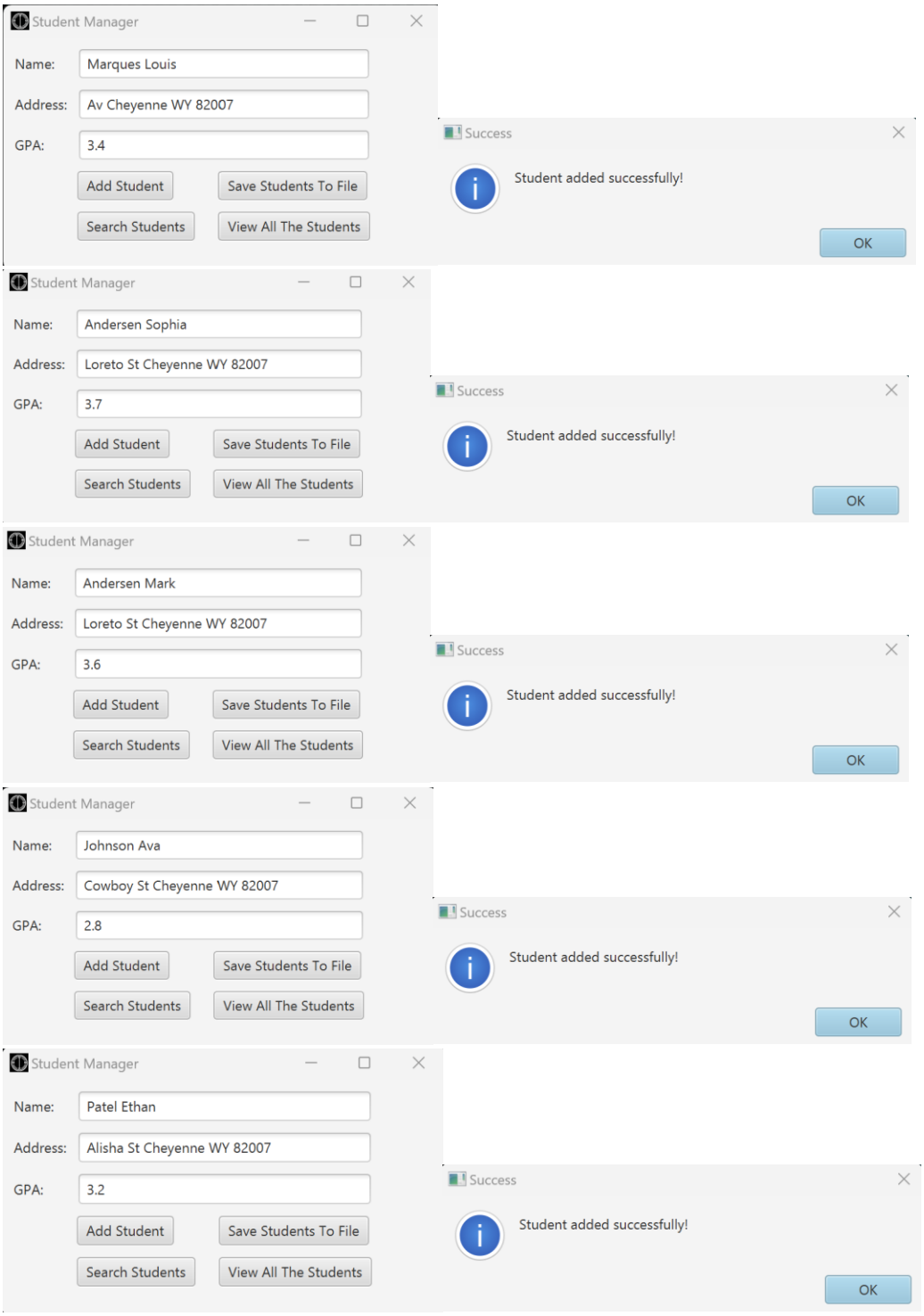
Sort By GPA



Note: After the user clicks "Sort by Name (A-Z)," the program displays all the students saved in the students.txt file sorted by name (A-Z).

Figure 9

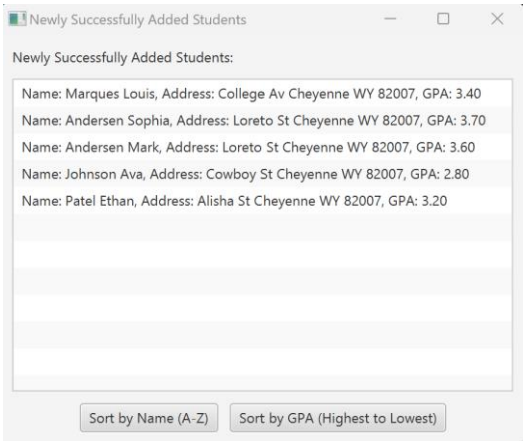
Add Students

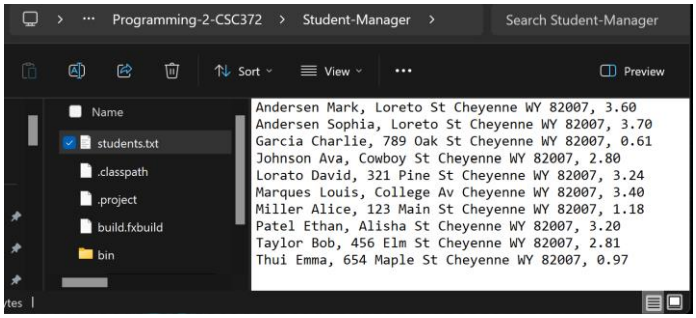


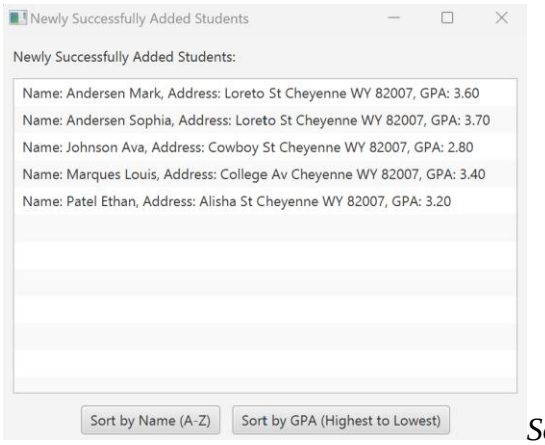
Note: After the user enters a student's data and clicks "Add Student," the program adds the student's data to a LindedList. Note that the student's data is not saved into the students.txt file, for that the user after adding the student(s) needs to click "Save Students To File."

Figure 10

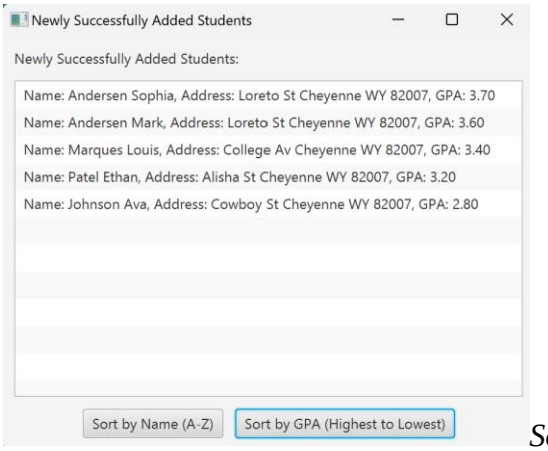
Save Students To File







Sorted by name

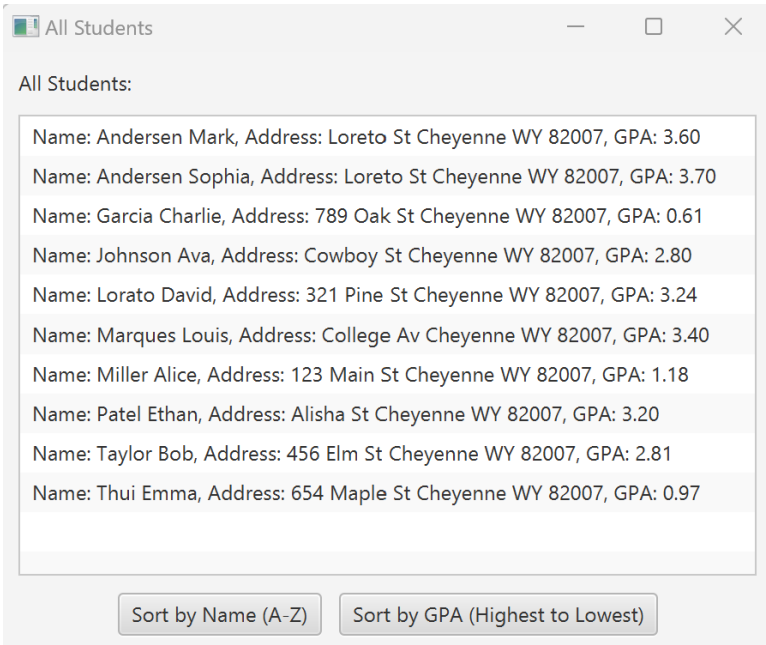


Sorted by GDP

Note: After adding the students, the user needs to click "Save Students To File." The program saves the newly added students into the students.txt file soted by name (A-Z) (output the contents of the linked list in ascending sorted order by name to a regular text file). Additionally, the program displays the newly added students. The newly added students are displayed in chronological order of entry. Furthermore, the list can be sorted by GPA and by name. Moreover, in the text file, the newly added students and the existing ones are combined and sorted by name (A-Z).

Figure 11

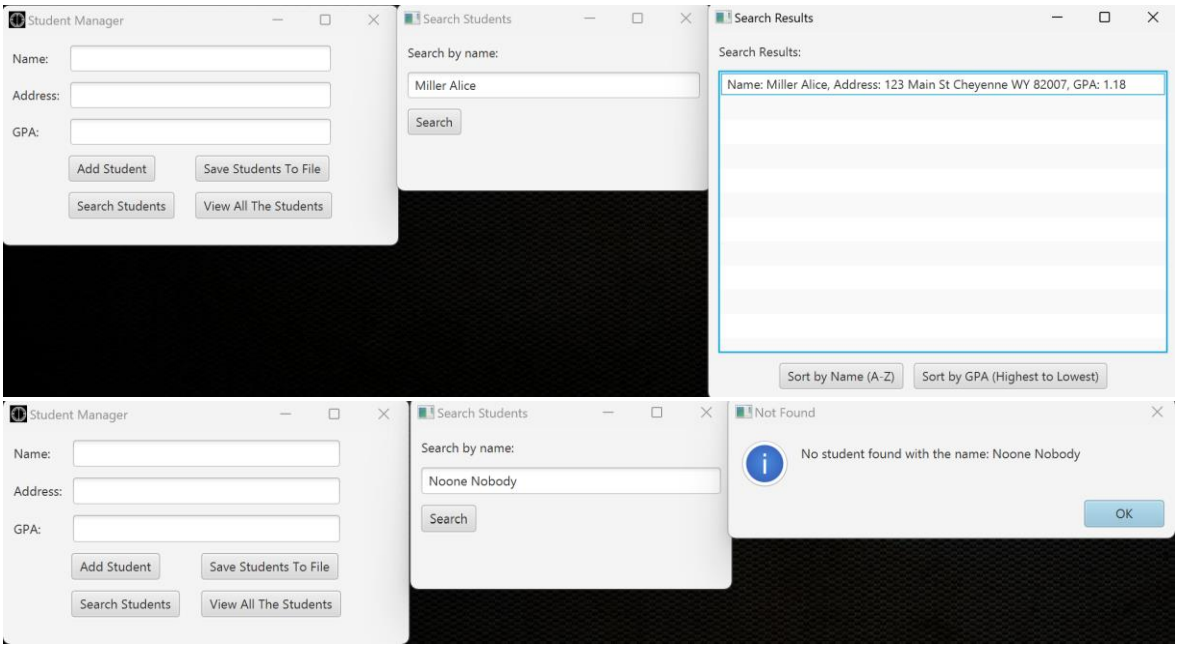
Display All Students



Note: After the user clicks "View All The Students," the program displays newly added students and the existing ones sorted by name.

Figure 12

Search Students

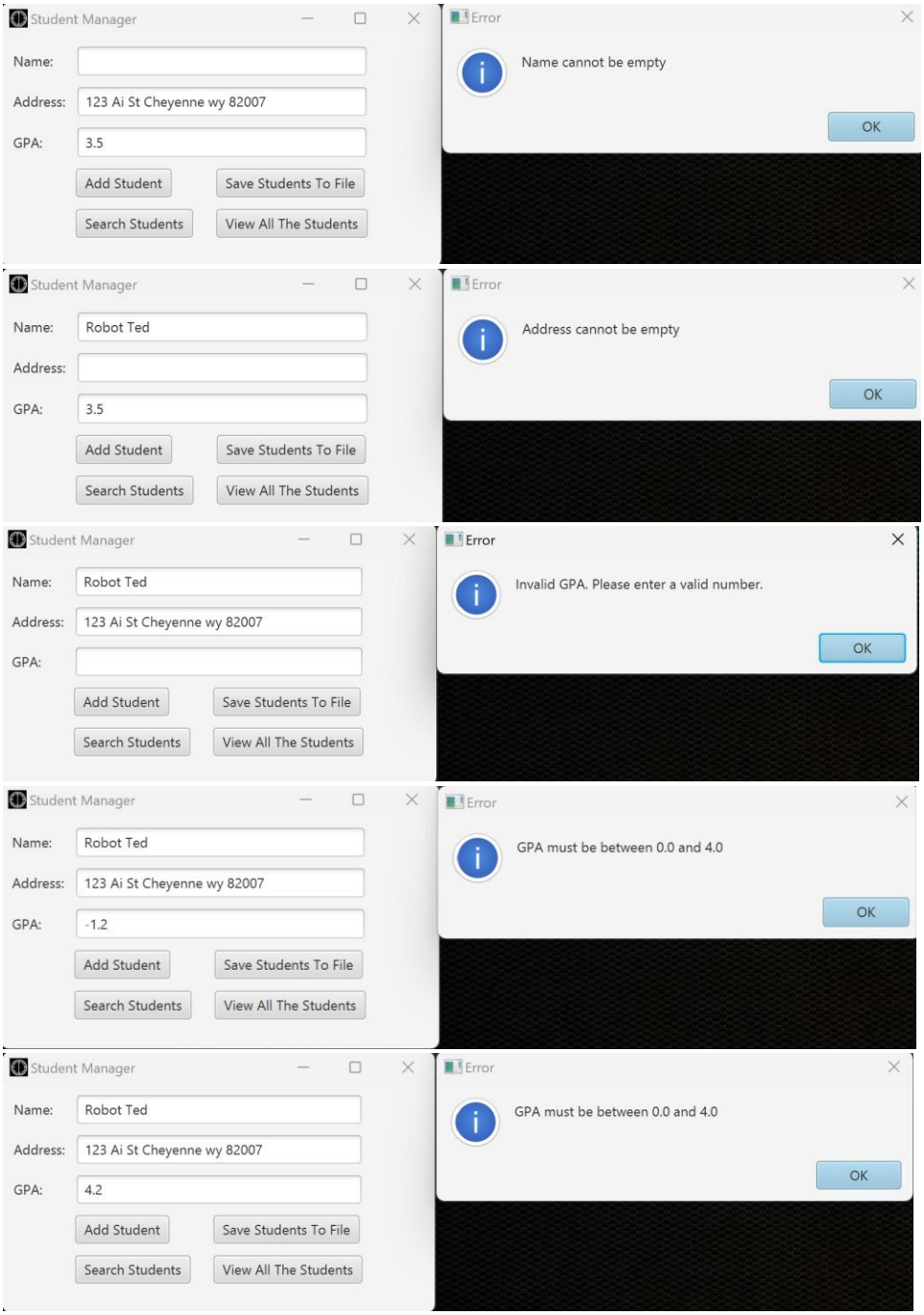


Note: After the user clicks "Search Student," the user enters the name to search and the program displays all the student data (if found).

User Input Validation

Figure 12

User Input Validation



As shown in Figure 1 through Figure 12 the program runs without any issues displaying the correct outputs as expected.