**Final Project**

**Programming 2 – COMP 123 - SEC. 012**

**Centennial College**

**Group 3**

Amanda Nunes Sardella #301288044

Carl Kevin Gasal #

Selene Chhangur #300742304

Arpit Singh

**April 16, 2023**

Contents

[Introduction 2](#_Toc132062799)

[Class Student 2](#_Toc132062800)

[Code: 2](#_Toc132062801)

[Student list 3](#_Toc132062802)

[Code: 3](#_Toc132062803)

[DisplayMenu Method() 3](#_Toc132062804)

[Code: 3](#_Toc132062805)

[Methods 4](#_Toc132062806)

[AddStudent() 4](#_Toc132062807)

[Code: 4](#_Toc132062808)

[DisplayAllStudents() 5](#_Toc132062809)

[Code: 5](#_Toc132062810)

[SearchStudent() 6](#_Toc132062811)

[Code: 6](#_Toc132062812)

[DeleteStudent() 7](#_Toc132062813)

[Code: 7](#_Toc132062814)

[DisplaySummary() 8](#_Toc132062815)

[Code: 8](#_Toc132062816)

[Additional Method: 9](#_Toc132062817)

[SaveStudentsToFile() 9](#_Toc132062818)

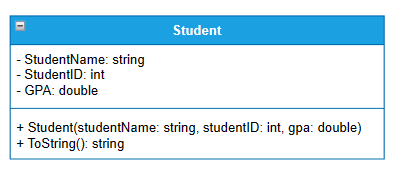
[Code: 9](#_Toc132062819)

[Conclusion: 9](#_Toc132062820)

# Introduction

In our project, we created a console application that manages student records. This application will allow users to add new students, display all students, search for students by name or ID, delete students, and display summary statistics. The student records will be stored in a text file for which we chose JSON.

## Class Student

****

### Code:

class Student

{

// Properties

public string StudentName { get; set; }

public int StudentID { get; set; }

public double GPA { get; set; }

// Constructor

public Student(string studentName, int studentID, double gpa)

{

StudentName = studentName;

StudentID = studentID;

GPA = gpa;

}

public override string ToString()

{

return $"{StudentName,-10} ID# {StudentID,-10} GPA:{GPA,-10:F2}";

}

}

We create this class in a separate file so it will be more accessible to find it. It is used to store the student information which is their name, the student’s ID and their GPA. The class Student contains three properties:

* **StudentName:** is a string representing the student's name.
* **StudentID:** an integer representing the student's ID.
* **GPA:** a double representing the student's grade point average.

The three properties are publicly accessible and have get and set accessors so that the main program can access and modify the values of each property. Encapsulating the data makes for easier use of the class throughout the program.

The constructor public Student(string studentName, int studentID, double gpa) initializes the properties of the object Student. The method public override string ToString() returns the properties in a string format that is formatted in a clean and readable manner.

## Student list

### Code:

static List<Student> student = new List<Student>();

static string filePath = "students.json";

In the main program we are creating a new list that will hold all the data for the students and will be written to/read from a JSON file named students.json. We chose JSON because it is most common amongst developers and was an important learning aspect of Programming 2 this semester.

## DisplayMenu Method()

### Code:

//main menu user will interact with

public static void DisplayMenu()

{

Console.WriteLine("=== Student Record Management System ===\n");

Console.WriteLine("1.) Add New Student");

Console.WriteLine("2.) Display All Students");

Console.WriteLine("3.) Search Student");

Console.WriteLine("4.) Delete Student");

Console.WriteLine("5.) Display Summary");

Console.WriteLine("6.) Exit");

}

This method shows the menu options that the user can choose from. We decided to create a method for the menu so that the code is cleaner and more readable. This method is called in our main and is incorporated with a switch case to read the user’s input.

do

{

Console.Clear();

DisplayMenu();

Console.Write("\nEnter the number of your choice: ");

choice = Console.ReadLine();

switch (choice)

{

case "1":

AddStudent();

break;

case "2":

Console.Clear();

DisplayAllStudents();

break;

case "3":

SearchStudent();

break;

case "4":

DeleteStudent();

break;

case "5":

DisplaySummary();

break;

case "6":

Console.WriteLine("\nThank you! Have a nice day!\nPress any key to exit...");

Console.ReadKey();

break;

default:

Console.WriteLine("\nInvalid input.\nEnter from 1 - 5, 6 for exit.");

Console.WriteLine("\nPress any key to return to choice options...");

Console.ReadKey();

break;

}

} while (choice != "6");

We decided that using a do while loop would best benefit handling exceptions errors. We have 6 options that the user can choose from, and entering any other character will prompt an error message that tells the user they must enter in the correct options. When the user does choose a valid option, it will take them to that menu. Each case there is a method that relates to the option. The 6 options are: add new student, display all students, search student, delete student, display summary and the last one will be the option to exit the menu.

# Methods

## AddStudent()

### Code:

static void AddStudent()

{

Console.Clear();

Console.WriteLine("+++ Add Student: +++\n");

Console.Write("Enter Student Name: ");

string studentName = Console.ReadLine();

Console.Write("Enter Student ID: ");

int studentID = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Student GPA: ");

double studentGPA = Convert.ToDouble(Console.ReadLine());

Student newStudent = new Student(studentName, studentID, studentGPA);

student.Add(newStudent);

Console.WriteLine($"\nStudent Added: {newStudent}\nPress any key to continue...");

Console.ReadKey();

// Serialize the list of students to JSON format

string json = JsonConvert.SerializeObject(student);

// Write the JSON data to the file

File.WriteAllText(filePath, json);

}

This method will allow the user to add a student in the list. They will need to insert the Student Name, Student ID and Student GPA. After, it will show a confirmation message that the student was added successfully. Then, the list of students is serialized to JSON format using the JsonConvert.SerializeObject(student) method and writes the JSON data to the file.

## DisplayAllStudents()

### Code:

static void DisplayAllStudents()

{

Console.WriteLine("+++ All Students: +++\n");

if (student.Count == 0)

{

Console.WriteLine("No students found.\nPress any key to continue...");

}

else

{

foreach (var stud in student)

{

Console.WriteLine(stud);

}

Console.WriteLine("Press any key to continue...");

}

Console.ReadKey();

}

This method will display a list of all the students in the list. First, it will check if the number of students in the list is equal to zero, if so, it will print a message saying that there are no students found. Else, if there are students in the list, the method loops through each student in the list using a foreach loop and prints the information to the screen.

## SearchStudent()

### Code:

static void SearchStudent()

{

Console.Clear();

Console.WriteLine("+++ Search Student: +++\n");

Console.Write("Search by\n1.) ID\n2.) Name\n3.) Back to Main\nEnter Option: ");

string searchOption = Console.ReadLine();

//created switch case for searching student with ID or Name

do

{

switch (searchOption)

{

case "1":

Console.Write("Enter Student ID: ");

int studentID = Convert.ToInt32(Console.ReadLine());

Student studentFound = null;

foreach (var id in student)

{

if (id.StudentID == studentID)

{

studentFound = id;

break;

}

}

if (studentFound == null)

{

Console.WriteLine("\nStudent not found.\nPress any key to continue...");

}

else

{

Console.WriteLine($"\nStudent Found: {studentFound}\nPress any key to continue...");

}

Console.ReadKey();

break;

case "2":

Console.Write("Enter Student Name: ");

string studentName = Console.ReadLine();

studentName.ToLower();

Student studentNameFound = null;

foreach (var id in student)

{

if (id.StudentName.ToLower() == studentName.ToLower())

{

studentNameFound = id;

break;

}

}

if (studentNameFound == null)

{

Console.WriteLine("\nStudent not found.\nPress any key to continue...");

}

else

{

Console.WriteLine($"\nStudent Found: {studentNameFound}\nPress any key to continue...");

}

Console.ReadKey();

break;

case "3":

Console.WriteLine("\nAny key to go back...");

Console.ReadKey();

break;

default:

Console.WriteLine("\nInvalid input.\nPlease enter a valid option.\n1.) Search by ID \n2.) Search by name \n3.) Exit.");

Console.WriteLine("\nPress any key to return to menu options...");

Console.ReadKey();

break;

}

} while (searchOption != "3");

}

This method adds a search function to find a particular student. The user will be prompted on how to search for the student: option 1 by student ID, option 2 by the student’s name, or option 3 to exit. We use the switch again to filter those options. If the input is wrong, the student will not be found, and a message will be displayed indicating that and the user is prompted to insert the right information again. We use a do while so that the search is repeated until the user chooses to exit from the search menu. Additionally, option 2 is set toLower() so it will disregard case sensitivity when finding for the student’s name.

## DeleteStudent()

### Code:

static void DeleteStudent()

{

Console.Clear();

Console.WriteLine("+++ Delete Student: +++\n");

Console.Write("Enter Student ID: ");

int studentID = Convert.ToInt32(Console.ReadLine());

Student studentFound = null;

foreach (var id in student)

{

if (id.StudentID == studentID)

{

studentFound = id;

break;

}

}

if (studentFound == null)

{

Console.WriteLine("Student not found.\nPress any key to continue...");

}

else

{

student.Remove(studentFound);

Console.WriteLine($"\nStudent Deleted: {studentFound}\nPress any key to continue...");

}

// Serialize the list of students to JSON format

string json = JsonConvert.SerializeObject(student);

// Write the JSON data to the file

File.WriteAllText(filePath, json);

Console.ReadKey();

}

This method will allow the user to remove a student from the list. The method Console.Clear() clears the screen of the previous text. It will prompt the user to enter a student ID. If the student exists, it will prompt that the student was deleted with all the student’s information. A foreach loop is used to search the list of students to see if the entered student ID matches any of the students' IDs in the list and it assigns to the variable studentFound. After this, an if else statement is used to check if studentFound is still null or if it contains a valid student’s information. In the else statement, it will remove the student from the list “student” and display a message that the student was deleted. We then update the list of students by serializing the list to JSON format and then write the JSON data to the file so if the user wants to display all students in the list it will be updated.

## DisplaySummary()

### Code:

static void DisplaySummary()

{

Console.Clear();

Console.WriteLine("+++ Summary of Students: +++\n");

Console.WriteLine($"Total number of students: {student.Count}");

Console.WriteLine($"Average GPA: {student.Average(student => student.GPA):F2}");

Console.WriteLine($"Highest GPA: {student.Max(student => student.GPA):F2}");

Console.WriteLine($"Lowest GPA: {student.Min(student => student.GPA):F2}");

Console.WriteLine("\nPress any key to continue...");

// Serialize the list of students to JSON format

string json = JsonConvert.SerializeObject(student);

// Write the JSON data to the file

File.WriteAllText(filePath, json);

Console.ReadKey();

Console.ReadKey();

}

This method will display a list of all the students in the list, the total number of students, the highest GPA, and the lowest GPA. We add the average option as well for our own personal preference. We use the methodConsole.Clear() to clear the screen of the previous text for a cleaner display. Using in-built methods Average(), Max(), and Min() we find the respective highest, lowest, and average GPA of all students to display.

# Additional Method:

## SaveStudentsToFile()

### Code:

static void SaveStudentsToFile()

{

string json = JsonConvert.SerializeObject(student);

File.WriteAllText(filePath, json);

Console.WriteLine($"\nStudent records saved to file: \n{filePath}");

}

This function is called upon exit or in option 6 of the Main Menu, Saving the file into .txt format and saved it to the default path inside the folder.

# Conclusion:

We really enjoyed working on this project with the team. We had a good rapport and we communicated clearly and frequently. We also made sure to review each other's code and test the program thoroughly. We think this project was a success because we worked well together and paid attention to the details.