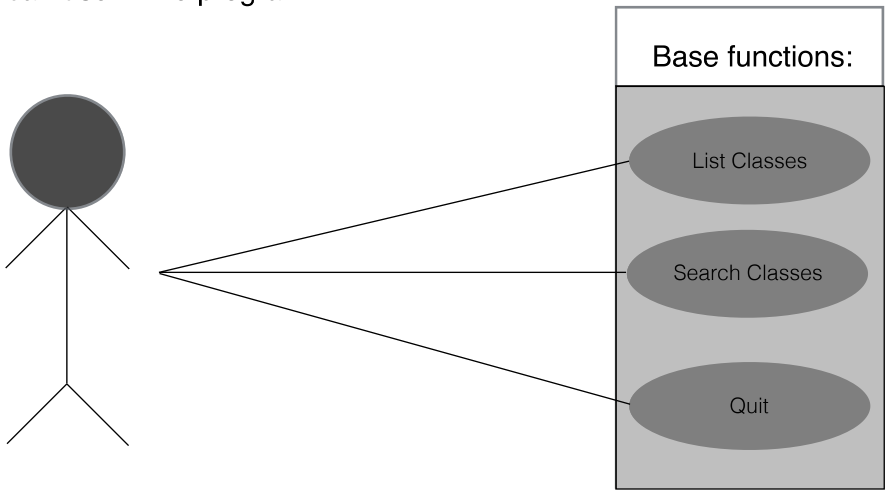
Use-case Diagram:

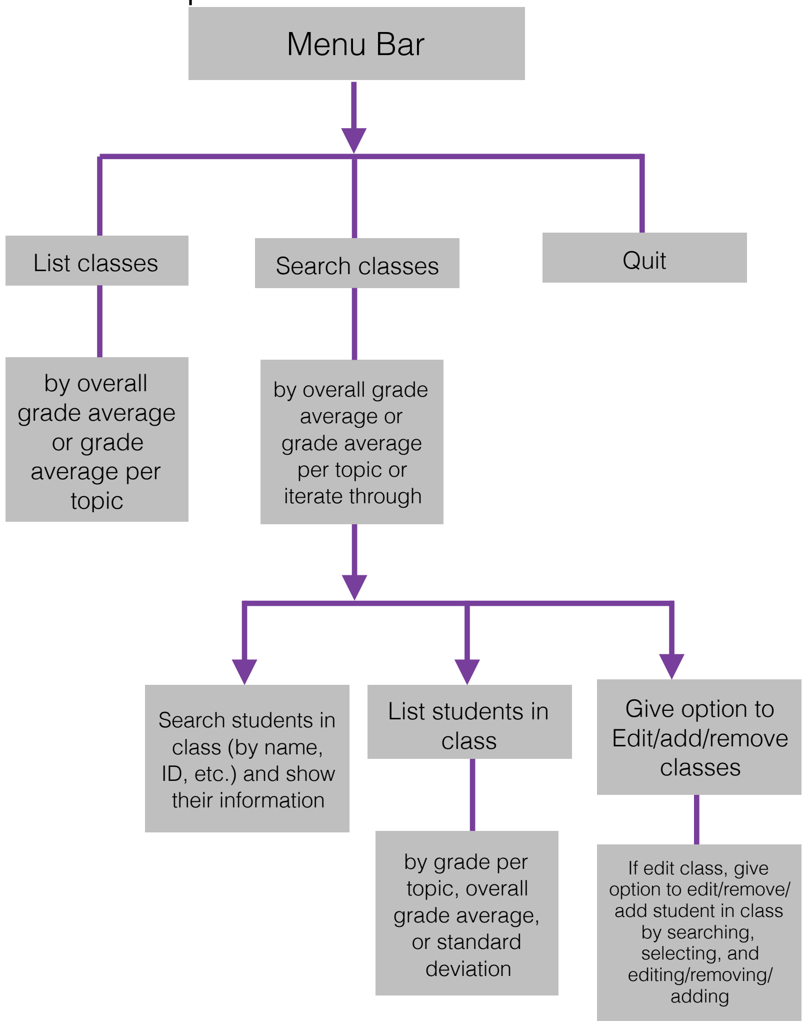
The program only has one user, the client. This diagram shows the different functions that the user can use in the program



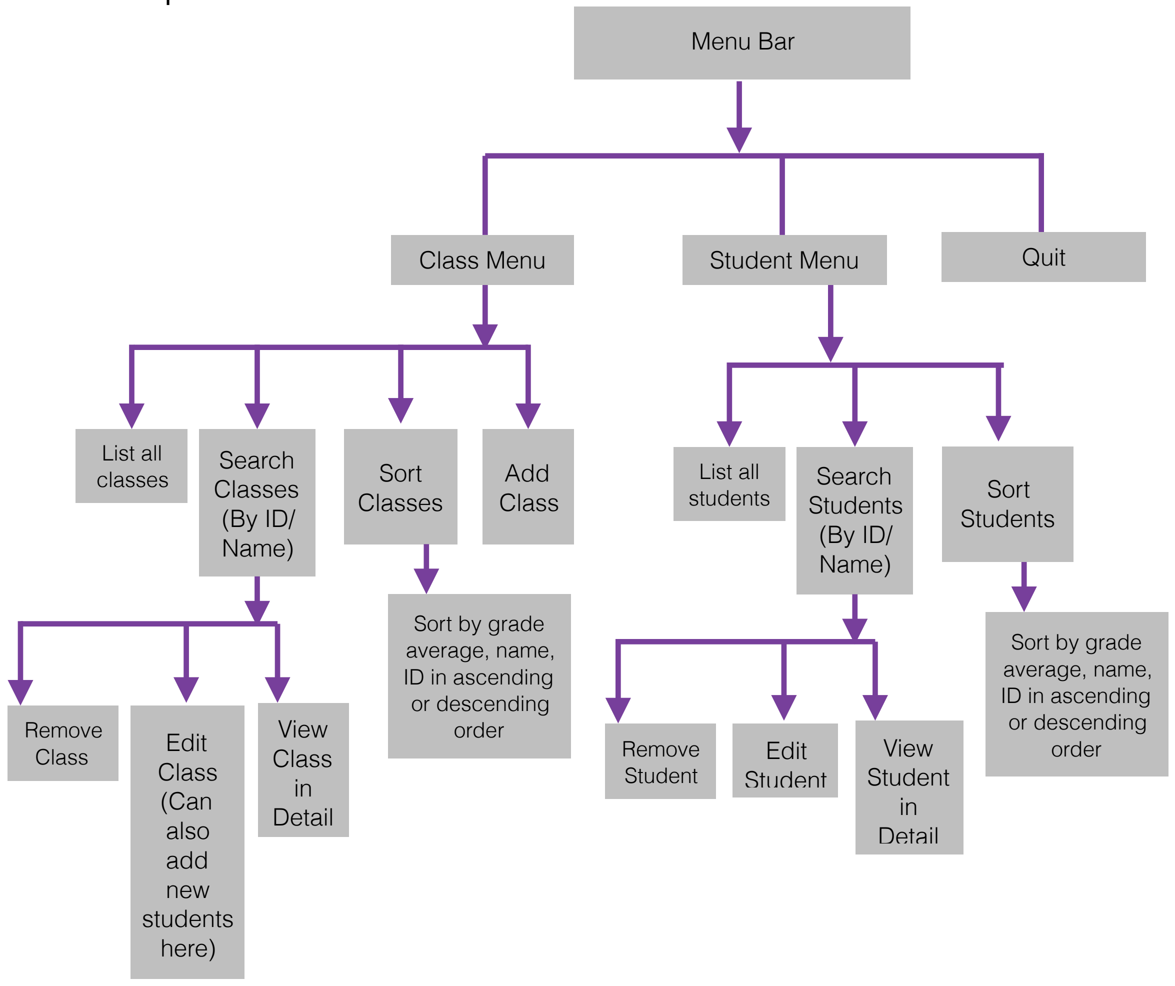
Top-down diagram:

This diagram shows the progression through the program that the user can choose. It's a breakdown of the options that the user can reach from the main menu.

Before development:


Menu Bar
List classes
Search classes
Quit
by overall grade average or grade average per topic
by overall grade average or grade average per topic or iterate through
Search students in class (by name, ID, etc.) and show their information
List students in class
Give option to Edit/add/remove classes
by grade per topic, overall grade average, or standard deviation
If edit class, give option to edit/remove/add student in class by searching, selecting, and editing/removing/adding

After development:


Menu Bar
Class Menu
Student Menu
Quit
List all classes
Search Classes (By ID/ Name)
Sort Classes
Add Class
Sort by grade average, name, ID in ascending or descending order
Remove Class
Edit Class (Can also add new students here)
View Class in Detail
List all students
Search Students (By ID/ Name)
Sort Students
Sort by grade average, name, ID in ascending or descending order
Remove Student
Edit Student
View Student in Detail

Test Plan:
The following is the test plan(s) for the program to be implemented for my client

When the user enters the date (DD/MM/YYYY) to remind for a new reminder, there would be a type restriction

Data Type	Type of Validation	Comment	Example	Test Pass/Fail
Normal	- Character Type Check - Presence Check - Range Checks - Length Checks	Clearly within the limits	04/11/2017, 21/7/2018	✓
Abnormal		Outisde the limits	45/14/208, -1/56/1	✓
Extreme		On the boundaries of the limits	31/12/2017, 01/01/2000	✓

When the user enters the neatness grade of a student, it has to be between 0 and 12 (0 represents no grade).

Data Type	Type of Validation	Comment	Example	Test Pass/Fail
Normal	- Range Check - Presence Check - Type check (Must be an integer, not a decimal - float or double)	Clearly within the limits	2, 10	✓
Abnormal		Outisde the limits	-1, 13, 205	✓
Extreme		On the boundaries of the limits	0, 12	✓

When the user enters the ID of a student, only numbers between 00000 and 99999 can be accepted, with 5 digits always having to be inputted

Data Type	Type of Validation	Comment	Example	Test Pass/Fail
Normal	- Range Check - Presence Check - Type check (Must be an integer, not a decimal - float or double)	Clearly within the limits	11111, 50876	✓
Abnormal		Outisde the limits	-1, 2098, 09876	✓
Extreme		On the boundaries of the limits	99999, 11111	✓

When the user enters a letter to select menu items, only letters relevant to the menu can be inputted (ignores case of letters - capital letter or not).

Data Type	Type of Validation	Comment	Example	Test Pass/Fail
Normal	- Character type check - Range check	Clearly within the limits	If menu is accepting input between a and d, then c, C, b	✓
Abnormal		Outisde the limits	1, 0 ,2	✓
Extreme		On the boundaries of the limits	If menu is accepting input between a and d, then a, d	✓

Defining Diagram:
This diagram breaks down the program in terms of the input, processing, and output defined by the client

Input	Processing	Output
Name of Student	- Search by student name - Sort by student name	- List student with neatness grade information, name, topics studied, etc. - List students by student name
ID of Student	- Search by Student ID - Sort by student ID	- List student with neatness grade information, name, topics studied, etc. - List students by student ID
Student neatness grade	- Sort by student neatness grade - Search by student neatness grade - Calculate topic and class neatness grade averages from indivdual student neatness grades	- List students by student neatness grade - List student with neatness grade information, name, topics studied, etc. - Set topic and class neatness grade averages
Name of topic	- Search topic by topic name - Sort by topic name	- List students taking specific topic - List classes taking specific topic - List topic neatness grade average - List topics by topic name

Data dictionary:

The data dictionary is an initial attempt at figuring out which data has to be captured

Data	Data type	Description
Student name	String	Must be entered
Student ID	int	Must be entered, 5 digits
Student grade in each topic	ArrayList<int>	Each is between 0 and 12 (integer 1 increments). 0 refers to not-applicable, student not having a grade.
Student average grade across topics	float	Between 0.0 and 12.0
Student average grade across topics (text)	String	"unsatisfactory", "adequate", "good", or "excellent"
General Remarks for each student	String	Optional input
Average grade for class	float	Between 0.0 and 12.0
Average grade for topic	float	Between 0.0 and 12.0
Extra information on class	String	Optional input
Name of topic	String	Must be entered
Extra notes for topic	String	Optional input
Class ID	int	Automatically created by program as classes are added
Topic ID	int	Automatically created by program as classes are added

File structure:

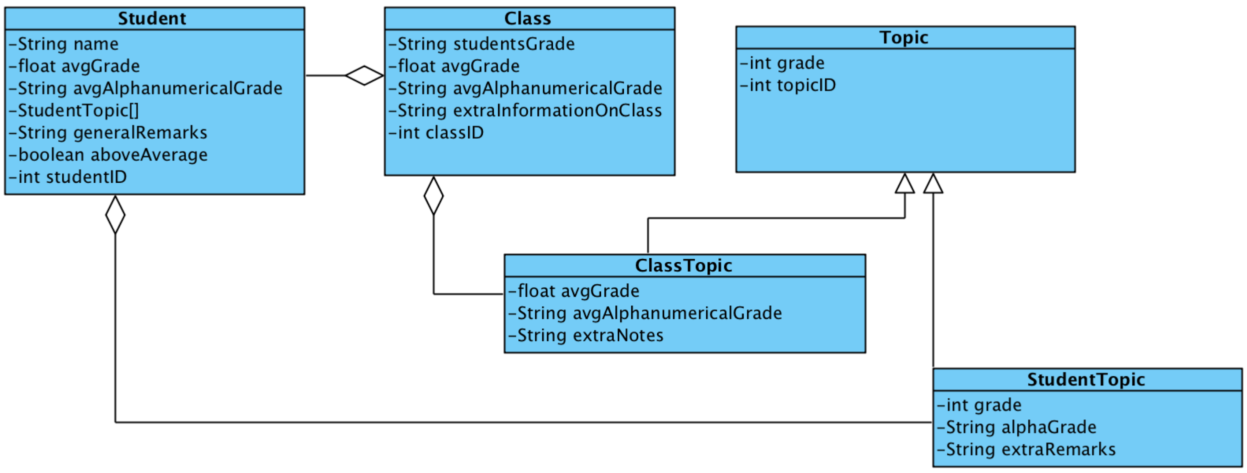
The file structure diagram is an initial attempt at figuring out the files required to save the data captured by the program. These will be stored in plain sequential text files (.txt files), not random access files.

Student	Class	Topic
Student Name: String	Students in class: ArrayList<int> (Use ID of each student to store)	Topic Name: String
Student ID: int	Class ID: int	Topic ID: int
Student grades in each topic: ArrayList<int>	Topics in class: ArrayList<int> (Use topic ID to store)	
General Remarks for each student	Extra information on class: String	

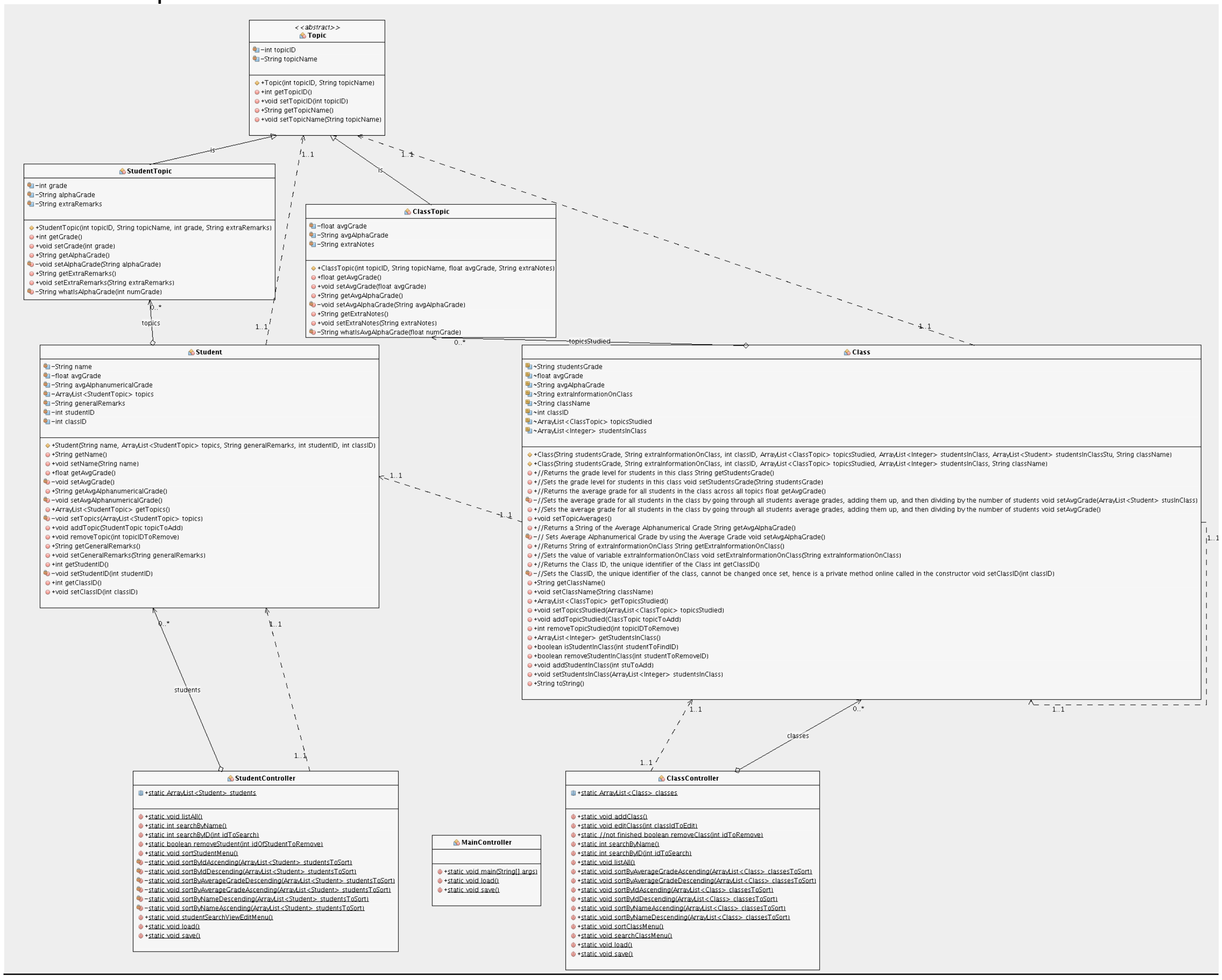
Class Diagram:

This diagram is a preliminary design for all the entity classes to be used in this program

Before development:



After Development:


<<abstract>>
Topic
-int topicID
-String topicName
+Topic(int topicID, String topicName)
+int getTopicID()
+void setTopicID(int topicID)
+String getTopicName()
+void setTopicName(String topicName)
is
is
1..1
1..1
StudentTopic
-int grade
-String alphaGrade
-String extraRemarks
+StudentTopic(int topicID, String topicName, int grade, String extraRemarks)
+int getGrade()
+void setGrade(int grade)
+String getAlphaGrade()
-void setAlphaGrade(String alphaGrade)
+String getExtraRemarks()
+void setExtraRemarks(String extraRemarks)
-String whatIsAlphaGrade(int numGrade)
ClassTopic
-float avgGrade
-String avgAlphaGrade
-String extraNotes
+ClassTopic(int topicID, String topicName, float avgGrade, String extraNotes)
+float getAvgGrade()
+void setAvgGrade(float avgGrade)
+String getAvgAlphaGrade()
-void setAvgAlphaGrade(String avgAlphaGrade)
+String getExtraNotes()
+void setExtraNotes(String extraNotes)
-String whatIsAvgAlphaGrade(float numGrade)
0..*
topics
1..1
0..*
topicsStudied
1..1
Student
-String name
-float avgGrade
-String avgAlphanumericalGrade
-ArrayList<StudentTopic> topics
-String generalRemarks
-int studentID
-int classID
+Student(String name, ArrayList<StudentTopic> topics, String generalRemarks, int studentID, int classID)
+String getName()
+void setName(String name)
+float getAvgGrade()
-void setAvgGrade()
+String getAvgAlphanumericalGrade()
-void setAvgAlphanumericalGrade()
+ArrayList<StudentTopic> getTopics()
-void setTopics(ArrayList<StudentTopic> topics)
+void addTopic(StudentTopic topicToAdd)
+void removeTopic(int topicIDToRemove)
+String getGeneralRemarks()
+void setGeneralRemarks(String generalRemarks)
+int getStudentID()
-void setStudentID(int studentID)
+int getClassID()
+void setClassID(int classID)
1..1
1..1
Class
~String studentsGrade
~float avgGrade
~String avgAlphaGrade
~String extraInformationOnClass
~String className
~int classID
~ArrayList<ClassTopic> topicsStudied
~ArrayList<Integer> studentsInClass
+Class(String studentsGrade, String extraInformationOnClass, int classID, ArrayList<ClassTopic> topicsStudied, ArrayList<Integer> studentsInClass, ArrayList<Student> studentsInClassStu, String className)
+Class(String studentsGrade, String extraInformationOnClass, int classID, ArrayList<ClassTopic> topicsStudied, ArrayList<Integer> studentsInClass, String className)
+//Returns the grade level for students in this class String getStudentsGrade()
+//Sets the grade level for students in this class void setStudentsGrade(String studentsGrade)
+//Returns the average grade for all students in the class across all topics float getAvgGrade()
-//Sets the average grade for all students in the class by going through all students average grades, adding them up, and then dividing by the number of students void setAvgGrade(ArrayList<Student> stusInClass)
+//Sets the average grade for all students in the class by going through all students average grades, adding them up, and then dividing by the number of students void setAvgGrade()
+void setTopicAverages()
+//Returns a String of the Average Alphanumerical Grade String getAvgAlphaGrade()
-// Sets Average Alphanumerical Grade by using the Average Grade void setAvgAlphaGrade()
+//Returns String of extraInformationOnClass String getExtraInformationOnClass()
+//Sets the value of variable extraInformationOnClass void setExtraInformationOnClass(String extraInformationOnClass)
+//Returns the Class ID, the unique identifier of the Class int getClassID()
-//Sets the ClassID, the unique identifier of the class, cannot be changed once set, hence is a private method online called in the constructor void setClassID(int classID)
+String getClassName()
+void setClassName(String className)
+ArrayList<ClassTopic> getTopicsStudied()
+void setTopicsStudied(ArrayList<ClassTopic> topicsStudied)
+void addTopicStudied(ClassTopic topicToAdd)
+int removeTopicStudied(int topicIDToRemove)
+ArrayList<Integer> getStudentsInClass()
+boolean isStudentInClass(int studentToFindID)
+boolean removeStudentInClass(int studentToRemoveID)
+void addStudentInClass(int stuToAdd)
+void setStudentsInClass(ArrayList<Integer> studentsInClass)
+String toString()
1..1
1..1
0..*
students
1..1
1..1
1..1
0..*
classes
1..1
1..1
StudentController
+static ArrayList<Student> students
+static void listAll()
+static int searchByName()
+static int searchByID(int idToSearch)
+static boolean removeStudent(int idOfStudentToRemove)
+static void sortStudentMenu()
-static void sortByIdAscending(ArrayList<Student> studentsToSort)
-static void sortByIdDescending(ArrayList<Student> studentsToSort)
-static void sortByAverageGradeDescending(ArrayList<Student> studentsToSort)
-static void sortByAverageGradeAscending(ArrayList<Student> studentsToSort)
-static void sortByNameDescending(ArrayList<Student> studentsToSort)
-static void sortByNameAscending(ArrayList<Student> studentsToSort)
+static void studentSearchViewEditMenu()
+static void load()
+static void save()
MainController
+static void main(String[] args)
+static void load()
+static void save()
ClassController
+static ArrayList<Class> classes
+static void addClass()
+static void editClass(int classIdToEdit)
+static //not finished boolean removeClass(int idToRemove)
+static int searchByName()
+static int searchByID(int idToSearch)
+static void listAll()
+static void sortByAverageGradeAscending(ArrayList<Class> classesToSort)
+static void sortByAverageGradeDescending(ArrayList<Class> classesToSort)
+static void sortByIdAscending(ArrayList<Class> classesToSort)
+static void sortByIdDescending(ArrayList<Class> classesToSort)
+static void sortByNameAscending(ArrayList<Class> classesToSort)
+static void sortByNameDescending(ArrayList<Class> classesToSort)
+static void sortClassMenu()
+static void searchClassMenu()
+static void load()
+static void save()

Algorithms to be used:
All the algorithms shown here were modified from Simple Program Design: A Step-by-step Approach by Thomson.

The following sorting algorithms will help meet success criteria L as the students and classes can be sorted by comparing various variables (average grade, name, ID, etc.)

Pseudocode to sort the students in a class by average grade:

```
1   I = NUMBER_OF_STUDENTS_IN_CLASS
2   ELEMENTS_SWAPPED = true
3   loop while (ELEMENTS_ SWAPPED)
4       J = 1
5       ELEMENTS_SWAPPED = false
6       loop while J <= I – 1
7           if STUDENTS_LIST(J).AVGGRADE > STUDENTS_LIST(J + 1).AVGGRADE then
8               TEMP = STUDENTS_LIST(J)
9               STUDENTS_LIST(J) = STUDENTS_LIST(J + 1)
10              STUDENTS_LIST(J + 1) = TEMP
11              ELEMENTS_ SWAPPED = true
12          end if
13          J = J + 1
14      end loop
15      I = I – 1
16  end loop
```

To sort by name replace line 7 with
"if STUDENTS_LIST(J).NAME > STUDENTS_LIST(J + 1).NAME then "

To sort by the grade of a specific topic grade replace line 7 with
"if STUDENTS_LIST(J).ID > STUDENTS_LIST(J + 1).ID then"

To sort by the grade of a specific topic grade replace line 7 with
"if STUDENTS_LIST(J).TOPICGRADE > STUDENTS_LIST(J + 1).TOPICGRADE then"

To sort all students (regardless of class) by average grade, replace line 1 with
"I = TOTAL_NUMBER_OF_STUDENTS"

To sort all students (regardless of class) by ID number, replace line 1 with
"I = TOTAL_NUMBER_OF_STUDENTS"
and line 7 with
"if STUDENTS_LIST(J).ID > STUDENTS_LIST(J + 1).ID then "

To sort all students (regardless of class) by name, replace line 1 with
"I = TOTAL_NUMBER_OF_STUDENTS"
and line 7 with
"if STUDENTS_LIST(J).NAME > STUDENTS_LIST(J + 1).NAME then "

Pseudocode to sort classes by average grade:

```
1   I = NUMBER_OF_CLASSES
2   ELEMENTS_SWAPPED = true
3   loop while (ELEMENTS_ SWAPPED)
4       J = 1
5       ELEMENTS_SWAPPED = false
6       loop while J <= I – 1
7           if CLASSES_LIST(J).AVGGRADE > CLASSES_LIST(J + 1).AVGGRADE then
8               TEMP = CLASSES_LIST(J)
9               CLASSES_LIST(J) = CLASSES_LIST(J + 1)
10              CLASSES_LIST(J + 1) = TEMP
11              ELEMENTS_ SWAPPED = true
12          end if
13          J = J + 1
```

```
14      end loop
15      I = I – 1
16  end loop
```

To sort by Class ID number, replace line 7 with
"if CLASSES_LIST(J).ID > CLASSES_LIST(J + 1).ID then"

To sort by average grade in a specific topic, replace line 7 with
"if CLASSES_LIST(J).TOPICAVGGRADE > CLASSES_LIST(J + 1).TOPICAVGGRADE then"

Following algorithm will help meet Success Criteria A which asks for students to be searched by Student ID number

Pseudocode to search for students by ID (List must be sorted according to ID in ascending order):

```
1   function search(STUDENTSARRAY, NAMETOFIND) : integer
2       set FOUND to false
3       set INDEX to 0
4       set LOCATION to -1
5       loop while not FOUND and INDEX < LENGTH
6           if STUDENTSARRAY[INDEX].NAME = NAMETOFIND then
7               set FOUND to true
8               set LOCATION = INDEX
9           end if
10          set INDEX to INDEX + 1
11      end loop
12      return LOCATION
13  end function
```

Following algorithm will help meet Success Criteria B which asks for classes to be searchable by Class ID number

Pseudocode to search for class by ID (returns location of student in ArrayList):

```
1   function search(CLASSARRAY, IDTOFIND) : integer
2       set FOUND to false
3       set INDEX to 0
4       set LOCATION to -1
5       loop while not FOUND and INDEX < LENGTH
6           if CLASSARRAY[INDEX].ID = IDTOFIND then
7               set FOUND to true
8               set LOCATION = INDEX
9           end if
10          set INDEX to INDEX + 1
11      end loop
12      return LOCATION
13  end function
```

Following algorithm will help meet Success Criteria A which asks for students to be searchable by name

Pseudocode to search for students by name (returns location of student in ArrayList):

```
1   function search(STUDENTSARRAY, NAMETOFIND) : integer
2       set FOUND to false
3       set INDEX to 0
4       set LOCATION to -1
5       loop while not FOUND and INDEX < LENGTH
6           if STUDENTSARRAY[INDEX].NAME = NAMETOFIND then
7               set FOUND to true
8               set LOCATION = INDEX
9           end if
10          set INDEX to INDEX + 1
11      end loop
12      return LOCATION
13  end function
```

Following algorithm will help meet Success Criteria B which asks for classes to be searchable by name

Pseudocode to search for students by name (returns location of student in ArrayList):

```
function search(CLASSARRAY, NAMETOFIND) : integer
    set FOUND to false
    set INDEX to 0
    set LOCATION to -1
    loop while not FOUND and INDEX < LENGTH
        if CLASSARRAY[INDEX].NAME = NAMETOFIND then
            set FOUND to true
            set LOCATION = INDEX
        end if
        set INDEX to INDEX + 1
    end loop
    return LOCATION
end function
```