

There are 4 basic classes to the Gradebook:

1. Student
2. Rubric
3. Class
4. ClassList

A.) Student

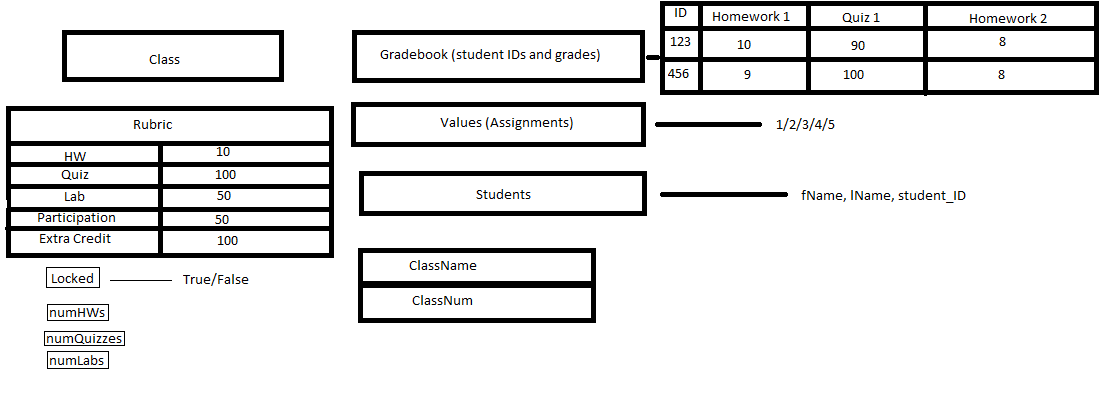
1. Student has 3 data members:
   1. fName - String for a student’s first name
   2. lName - String for a student’s last name
   3. Student\_id - An integer ID number
2. Student has no default constructor, only a constructor that accepts 3 data members
3. Student has 3 functions that are self-explanatory
   1. getFirstName() - Returns fName
   2. getLastName() - Returns lName
   3. getID() - Returns student\_id

B.) Rubric

1. Rubric has 5 data members
   1. HWValue - Value of HW for a specific class
   2. quizValue - Value of quizzes for a specific class
   3. LabValue - Value of labs for a specific class
   4. participationValue - Value of participation for a specific class
   5. extraCreditValue - Value of extra credit for a specific class
2. Rubric has both a default constructor and a constructor that accepts information for all its data members
   1. The default values are, in order with how they were listed: 10, 100, 50, 50, 100
3. Rubric has 6 functions
   1. The first 5 functions return values of the data types
   2. editRubricValues(int HW, int quiz, int lab, int PV, int ECV) - Edits the values of the rubric to the corresponding integer that was passed to the function.

C.) Class

1. The heart and brains of the Gradebook, most of the information is stored here.
2. Has 10 data members
   1. Gradebook - A Vector that contains Vectors that contains integers. A 2D matrix of integers. The [0][x] position is always a student ID, with every other position being a grade
   2. Students - A Vector that contains Student objects
   3. Values - A Vector that contains Assignment types (1 for HW, 2 for quiz, etc.)
   4. numHWs - The number of homework assignments this class has been assigned
   5. numQuizzes - The number of quizzes this class has been assigned
   6. numLabs - The number of labs this class has been assigned
   7. R - The Rubric for this class
   8. className - A String for the name of the class. (e.g. CISC)
   9. classNum - The number for this class. Combines with className to make the file name. (e.g. CISC2200)
   10. Locked - A Boolean variable that either allows or restricts editing
3. Has a default constructor
   1. Since the user names the class, adds the students, and edits the rubric via the GUI there’s no need for a specific constructor
   2. The default constructor creates the data members, makes a default student that occupies a slot which gets overwritten immediately (to avoid an OOB error), sets locked to false (to allow editing), and creates a directory for the class if one does not exist yet
4. Class(String name, int number): Class Constructor which collects information from Files
   1. Uses name and number of a class in order to determine which standard files to grab information from: For example, a class with the className = “CIS” and the classNum “3595” will have a key of “CIS3595” which we will use as an example in the following files.
   2. returnStudentsFile(): function used to fill the Class’ students vector from a saved file known as “CIS3595Students.txt”
   3. returnAssignmentsFile(): function used to fill the Class’ Values vector from a saved file known as “CIS3595Assignments.txt”
   4. returnRubricFile(): function used to fill the Class’ Rubric value from a saved file known as “CIS3595Rubric.txt”
   5. returnLockFile(): function used to determine whether the Class is locked or unlocked. Determines this from a file known as “CIS3595Lock.txt”
   6. Each of these files has a corresponding save function which are used in order to store the information back to the standard files within the Class’ subdirectory. These functions include “createStudentsFile(), createAssignmentsFile(), createRubricFile(), and createLockFile()”
   7. The grades for the class are then read into the 2D Vector variable Gradebook by reading into the data from a file known as “CIS3595grades.txt”. The save function corresponding to this function is createFile().
5. Math functions
   1. getAverageForAssignment(int assignment) - Gets the average for a given assignment
   2. getStanDevForAssignment(int assignment) - Gets the standard deviation for a given assignment
   3. getWeightedAverageForStudent(int ID) - Takes the rubric values for the class, multiplied by the number of assignments in that specific category of assignments (except participation which only has 1 value and extra credit is not included into this yet), and adds up into a variable called “divisor”. Adds up the total points lost into a variable called total. Then the formula is: [(divisor + extra credit - total)/(divisor)]\*100
      1. For example, a class has 3 homeworks, 1 quiz, and 1 lab. A student gets 10 out of 10 on all homeworks, 90 out of 100 on the quiz, and 40 out of 50 on the lab. They have a participation value of 50 out of 50. They earned 5 points extra credit. Divisor = (3\*50) + (1\*100) + (1\*50) + 50 = 350. Total = 20 (20 points lost). Extra credit = 5. [(350 + 5 - 20)/(350))] \* 100 = (335/350) \* 100 = 95.71 weighted average
   4. getClassAverage(): runs getWeightedAverageForStudent() function on each student in the class and then returns the average of the entire class
   5. getClassStdDev(): returns the standard deviation of all the student averages within the class
   6. getClassMax(): returns highest average among the students in the class
   7. getClassMin(): returns lowest average among the students in the class
   8. getMode(): returns most frequent average among the students in the class
   9. getMedian(): returns middle average among the students in the class
6. Backend functions
   1. addAssignment(int type) - Creates a new entry in the Values Vector with the number passed (again, 1 for HW, 2 for quiz, etc.). If the entry is a HW/Quiz/Lab, the respective numXXX data member is incremented by 1. Afterwards, the function adds a value of 0 in this new assignment category in the gradebook for all students.
   2. getAssignmentType(int assignment) - Goes to the position in the Values Vector of the value of assignment and returns the type of assignment in that location.
   3. addStudent(Student kid) - A Student object gets passed to this function. This object is added to the Students vector. A new vector is created that is filled with 0s to be added to the gradebook vector (which is a vector of vectors). The first time a student is manually added to a class, it overwrites the default Student in the gradebook vector. Otherwise, it is appended to the end.
   4. changeGrade(int newGrade, int ID, int assignment) - Sets the location of [ID][assignment] to newGrade. Allows for quick and easy editing.
   5. lockOrUnlock() - Reverses whether the class is locked or unlocked when called. A locked class with be unlocked, and vice versa.
   6. validTable( JTable grades ): checks if the values entered into the JTable by the user are adequate with regard to the ranges set by the Rubric of the class. Returns true if the values entered fit the criteria, otherwise it returns false.



D.) ClassList

1. The highest data structure of our Gradebook program. Primarily used for loading data for the JTable and keeping track of the different classes saved to memory.
2. Contains 4 data members:
   1. List: used to load classes on startup
   2. currClassData: used for storing grade data to be displayed
   3. columnNames: used for display of assignment types
   4. lastEdited: keeps track of which class was most recently edited
3. File Functions
   1. saveLast(): saves the lastEdited variable, or class, value to the lastEdit.txt file found within the Classes subdirectory.
   2. loadLast(): loads the lastEdited variable, or class, value from the lastEdit.txt file found within the Classes subdirectory.
   3. loadClassFile(): loads all classes from ClassList.txt file found within the Classes subdirectory. This allows for display of all Classes that are currently in memory to the main program
   4. createClassFile(): saves all classes from the main program into memory so they can be loaded again during next use. Saves these classes to ClassList.txt found within the Classes subdirectory.
4. loadTable (Class currClass)
   1. Initially, the function loads information into the columnNames from the “Class”Assignments.txt file. This is to ensure that each assignment is properly typed and numbered.
   2. The rest of the function consists of assigning the data from “Class”grades.txt into the currClassData variable for the classList data structure.
   3. These two loaded variables will now be used when information is grabbed from files to load data into the JTable.

The GUI consists of 7 Separate Frames:

1. LogIn
2. Layout (the main frame)
3. AddClass/AddFirstClass
4. AddAssignment
5. AddStudent/AddFirstStudent
6. FinalGrades
7. EditRubric
8. LogIn()
   1. Displayed on startup
   2. Asks user for password, if entered correctly loads Layout
   3. If no classes exist, loads AddFirstClass
9. Layout(JTable grades, ClassList semester)
   1. JPanel navPanel
   2. JTabbedPane tabbedPane
      1. First tab: JPanel gradesPanel: contains all the grades information and buttons
         1. JButton saveGradesButton: saves grades
         2. JButton addAssignmentButton: calls AddAssignment
         3. JButton addStudentButton: calls AddStudent
         4. JButton finalGradesButton: calls FinalGrades
         5. JButton lockClassButton/unlockClassButton: locks/unlocks the class respectively
      2. Second tab: JPanel rubricPanel: contains rubric information and edit button
      3. Third tab: JPanel statsPanel: contains statistics
   3. JSplitPane splitPane: creates a split pane using navPanel and tabbedPane
10. AddClass/AddFirstClass
    1. AddClass(ClassList semester, Class c1, Layout currLayout) & AddFirstClass(ClassList semester, Class c1)
       1. Same function but have different calls based on whether or not any classes already exist
    2. JPanel addClass: asks user for input of class name and number
    3. JButton save: calls AddStudent/AddFirstStudent
11. AddAssignment(ClassList semester, Layout currLayout)
    1. JPanel addAssignment
       1. JComboBox assignmentType: for the user to pick which assignment type they’d like
12. AddStudent/AddFirstStudent
    1. AddStudent(ClassList semester, Layout currLayout) & AddFirstStudent(classList semester)
       1. Same function but have different calls based on whether or not any classes already exist
    2. JPanel addStudent: asks user for input of student first and last name and ID number
    3. JButton save: saves the single student and closes the window
    4. JButton savePlus: saves student and opens another AddStudent frame for input of more students
13. FinalGrades(Class c1)
    1. JPanel finalGrades: displays final grades for the students in the current class
14. EditRubric(Class c1, ClassList semester, JTable table, Layout currLayout)
    1. JPanel editRubric: displays rubric values in editable JTextFields for the user to change them if they wish