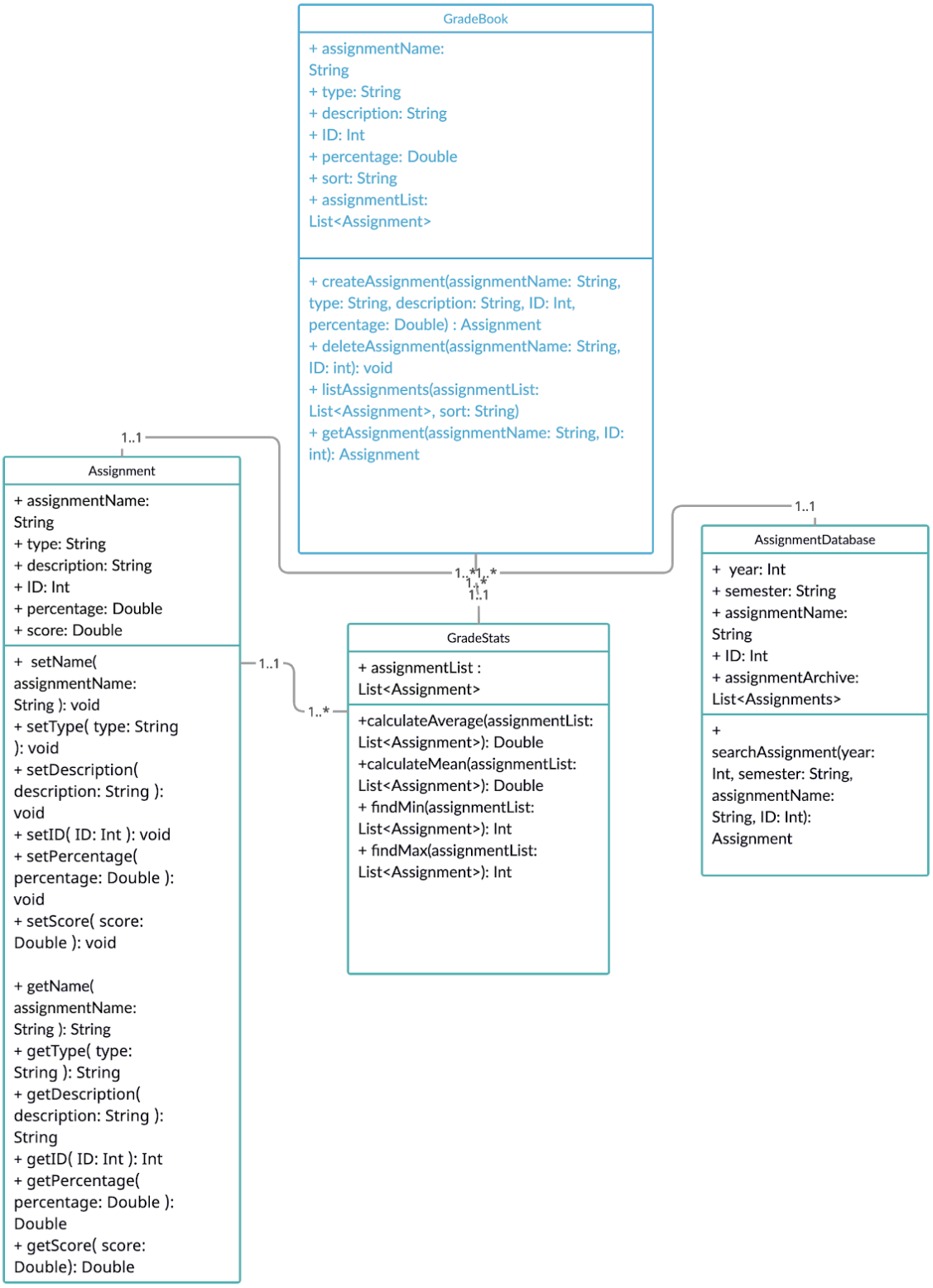
CS496 Gradebook Software

Hashim Abdirahim

Alberto Guadiana

Jason Songvilay

Software Architecture Overview



Classes

The implementation for the Gradebook software will consist of a Gradebook Class, a class for Assignments themselves, GradeStats, and an AssignmentDatabase. The Gradebook class will be the primary class for our implementations which include creating assignments with specific listed attributes, deleting assignments, and sorting assignments by a selection of attributes.

From the Gradebook class, we will include an Assignment class which will house the assignment itself featuring the name of the assignment, it's type, description, ID number, percentage and score. This class will feature a set of getters and setters to allow alterations and access to information when necessary. The Assignment class will be part of a list of Assignment objects to allow multiple assignments to be displayed at once.

The next class, GradeStats, will feature multiple functions to calculate meaningful statistics for assignments such as the average, minimum, and maximum scores from a list of assignments.

Lastly, we will have an AssignmentDatabase class that will store a list of Assignments by year and semester. The lists in this class may be accessed by functions that take in these parameters and sort them based on their values.

Attributes

The attributes of the Gradebook class will consist of an assignmentName (String), type (String), description (String), ID (Int), percentage (Double), sort (String), and a List<Assignment> of assignments named as assignmentList. These attributes are the primary pieces of information required by our Software Requirements Specification document in order to create an assignment. These variables will be utilized by our functions to allow the user to create, delete, sort, and get assignments.

The attributes of the Assignment class will include the name of the assignment, assignmentName (String), type (String), description (String), ID (Int), percentage (Double), and score (Double). These attributes will be passed from the Gradebook class when an assignment is formally created and be customizable through basic getters and setters found within this class.

The attributes of the GradeStats class will be a list of Assignments, assignmentList (List<Assignment>). An assignment can be calculated using the getters mentioned previously in order to compute the statistics necessary.

Lastly, the attributes of the AssignmentDatabase will be the desired year (Int), semester (String), ID (Int), and an assignmentArchive (List<Assignments>). In this class, the attributes will be primarily used to look up and sort through the list of assignments, as the information of assignments must persist throughout the years.

Operations

The functions of the Gradebook class will outline the premise of the SRS document. In this class it will have a function, createAssignment(assignmentName: String, type: String, description: String, ID: Int, percentage: Double), taking in the attributes outlined before returning a new Assignment before appending it to our list of assignments. The next function, deleteAssignment(assignmentName: String, ID: int), taking in descriptive parameters listed and returning void, will delete the assignment specified and remove it from the list of assignments (unless it cannot be found, which will be handled). After, we will have a function, listAssignments(assignmentList: List<Assignment>, sort: String), to list out assignments based off of a sort (String) attribute of either percentages or names.

The functions of the Assignment class will consist of basic getters and setters to allow for manipulation of the attributes outlined in the Gradebook class. Each parameter will have a set method to alter parameters, as well as a get method to return the desired attribute.

The functions of the GradeStats class will primarily deal with an assignmentList parameter. The calculateAverage(assignmentList: List<Assignment>) function will calculate the average score of the Assignments based on their score and return their percentages as a Double. The next functions, findMin(assignmentList: List<Assignment>) and findMax(assignmentList: List<Assignment>) will also return Doubles of the lowest and highest scores from an assignmentList, implemented using the appropriate sorting algorithms for maximum lookup efficiency.

The lone function for the AssignmentDatabase class, searchAssignment(year: Int, semester: String, assignmentName: String, ID: Int), will return an Assignment object based on the inputted search parameters for this function. However, the implementation for this class is subject to change as we conduct tests to figure out the greatest efficiency for assignment lookups.

Development plan and timeline

- Partitioning of tasks
    - Tasks will be divided for the following: development, implementation, and testing.
    - Discussion of how the software will be efficient is to take place at first. This will ensure that the logic of the code is efficient; hence avoiding unnecessary code that may hinder the efficacy of the software.
        - It is important to create a responsive and accurate software that is not clunky and/or slow.
    - The software is to be collaboratively developed with each member of the development contributing to the code. There is no single person that should do all of the work.
        - Each member will be assigned to develop one or more functions that will be present within the software.

        - Members are to frequently pull the updated document and frequently push their changes. Every change should be noted by adding comments when pushing. A github repository is to be created in order to keep track of progress made.
    - When code is complete testing should take place immediately. This would be the implementation and testing of the software taking place at the same time.
        - Note, this will be done by creating automated scripts. Software will not be distributed to potential customers during this time.
        - This phase of the project will serve to address any bugs that are present in the software. It gives the developers the ability to determine if a more efficient algorithm is to be used and what hinders performance of the software.
    - Once implementation and testing is complete, it would be beneficial but not required to have a professor test the software for feedback on its features and functionality.
        - Note, the software has been polished at this point in time. This only serves to further improve the software and give it more functions if there is a request for any. Again, this is only an option.
        - Making the software the best it can be while maintaining efficiency and performance is critical for its success.
    - Finally, the client's software is ready for distribution.

- Team member responsibilities

    - Developers: Alberto, Jason, Hashim
        - The developers will implement the task at hand into a software for the client
        - Each developer is responsible to make their assigned functions work with the one's from their team members
        - All developers must be pushing changes and pulling through github in order to keep an updated and steady workflow
        - Developers are to be in frequent communication with one another describing what they have changed, added, and/or removed.

    - Testers: Alberto, Jason, Hashim
        - The testers will be in charge of making sure each function works properly by created automated scripts
        - Automated scripts will ensure that the software performs as it should and is not slow
        - Testers are to keep in mind that the software should perform all functions accurately without computation errors and in a quick

manner. The faster grades are calculated and organized the better for the client

Timeline

3/15-3/17 - Complete code skeleton, implement all basic functions

3/17-3/19 - Combine classes, test to ensure functionality is correct

3/19-3/22 - Continue testing, refactor code if possible

3/22-3/24 - Complete code for Software release 1.0