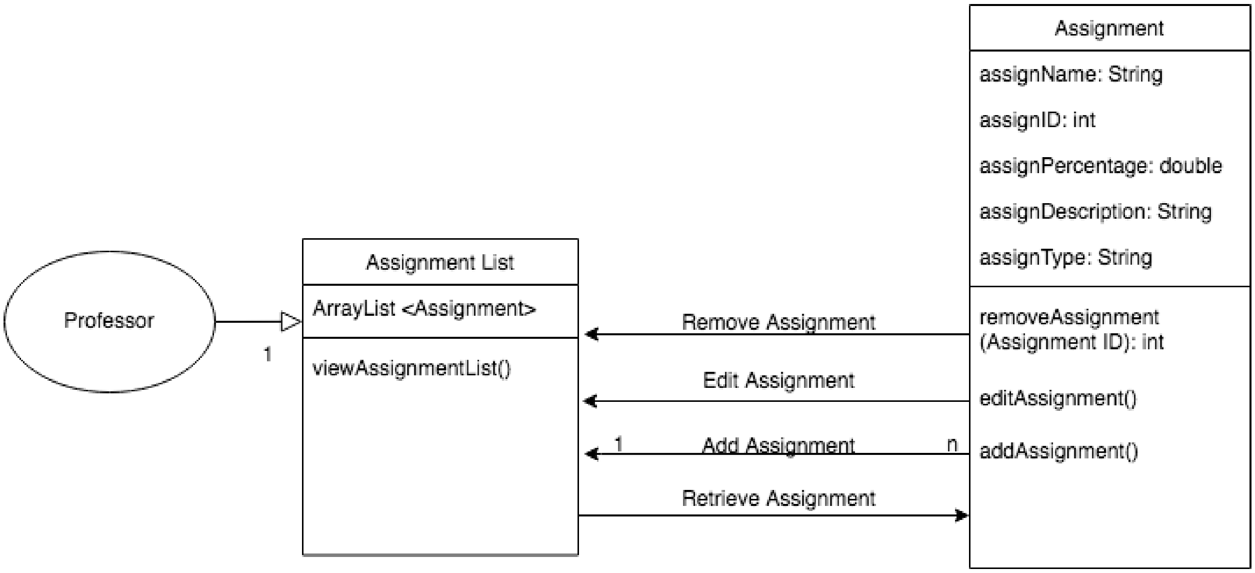
Class Gradebook Software Design 2.0

Created By: Brandon Reiley, Bhavya Desai, and Erik Torres

Software Architecture Overview

Introduction: The purpose of this software design specification is to display and discuss the continued development of the class gradebook software system. The document will begin by detailing the class structure and architecture that was implemented into the system. The document will then discuss updates from the original design specification, planned verification and validation methods, and a planned data management strategy.



The diagram above is a UML (Unified modeling language) class diagram. Each class has a name attached to them and contains attributes which are specified by instance variables. Furthermore, if needed, a class will contain operations or functions. The overall purpose of the class diagram is to display the structure of a Gradebook software system. It consists of 2 main classes: the Assignment List class and the Assignment class.

Class Descriptions:

- Assignment List class: This class is designed to hold and allow access to the assignment objects.

- Assignment: This class is used to create assignments that are essentially stored in the assignment list.
-

Attributes of each class:

- Assignment List: The attribute ArrayList<Assignment> holds the assignments created in an ArrayList.

- Assignment: The assignName attribute holds the name of an assignment and is of data type String. The assignID attribute holds the attached Identification number of an assignment and is of datatype int. The assignPercentage attribute is used to hold the percentage an assignment has for the total grade and is of datatype double. The assignDescription attribute is used to hold a description of an assignment; it uses a String datatype. The assignType attribute is of datatype String and holds information on the assignment type.

Operations of each class:

- Assignment List: The viewAssignmentList() operation allows the user to view the list of existing assignments.

- Assignment: The removeAssignment() operation removes an assignment from a gradebook object with the only parameter being the assignment name. The addAssignment() operation allows for the creation of an assignment. The editAssignment() operation allows for the contents of an assignment to be modified. Any attributes that an assignment may have can be changed as needed.

Updates from Original Specification

The primary changes to the Design Specification are the UML diagram and information regarding the diagram. The changes were made to reflect our initial build for the Class gradebook system rather than the planned development. Our initial build included one assignment list, which was initially called a gradebook, in which the user can add, remove, and edit assignments as needed. The intended functionality that was planned for the system did not change once the system was built. Information regarding the class descriptions and operations were also updated.

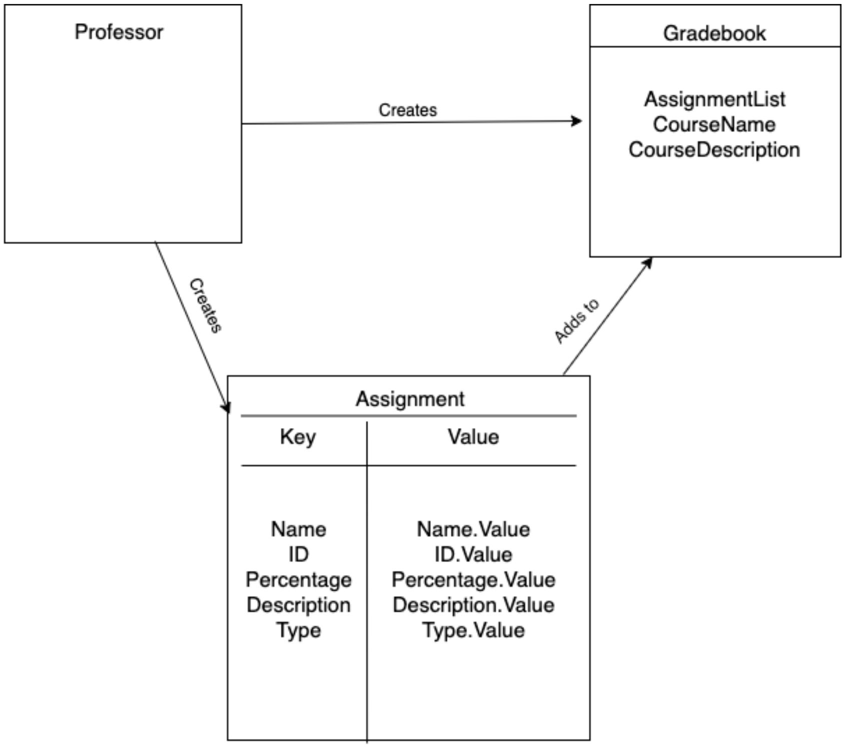
Verification and Validation

Verification: In order to verify the system, testing will be implemented throughout the system. First, there are planned tests for the unit granularity, particularly with each attribute of an assignment. This would begin by testing for errors with input, such as with inputting a String when an integer is expected and such. After this, plans to test if each unit's information is being stored correctly will be done. Following this would be functional testing through black-box testing. This will apply to each function of the system. Planned testing for the the addAssignment function includes testing to see if an assignment was successfully added to the Assignment list and if the information of an assignment was maintained. Similar testing will occur for the editAssignment function, particularly to see if modifications remain after the function is called. For the removeAssignment function, planned testing includes ensuring that an assignment was successfully removed from the assignment list and that no other assignments were affected. Lastly, for the viewAssignmentList function, planned testing includes checking to see if correct assignment information is being displayed, and if assignments are displayed in the correct order. Displaying modified assignments will als be tested.

Validation: To validate whether the system constructed meets the requirements, we plan on receiving and implementing feedback from the client. Test plans may change if the current system does not meet the requirements.

Data Management Strategy

We will be using a NoSQL database system for securing our class gradebook data. NoSQL has many uses that line up perfectly with our needs, such as following software development practices, delivering prototypes, object-oriented programming, and the use of local data transactions. We will be using a key value pair based database to store our information. It is a lot like a hash table where each key is unique, and we will be able to easily access the data by searching for an assignment's name, id, percentage, description, or type. Below is a simple diagram of how the NoSQL database will work, with the professor creating a gradebook for a specific course, as well as adding each assignment into that gradebook.


Professor
Creates
Gradebook
AssignmentList
CourseName
CourseDescription
Creates
Adds to
Assignment
Key
Value
Name
ID
Percentage
Description
Type
Name.Value
ID.Value
Percentage.Value
Description.Value
Type.Value