Documentation of Controller Class

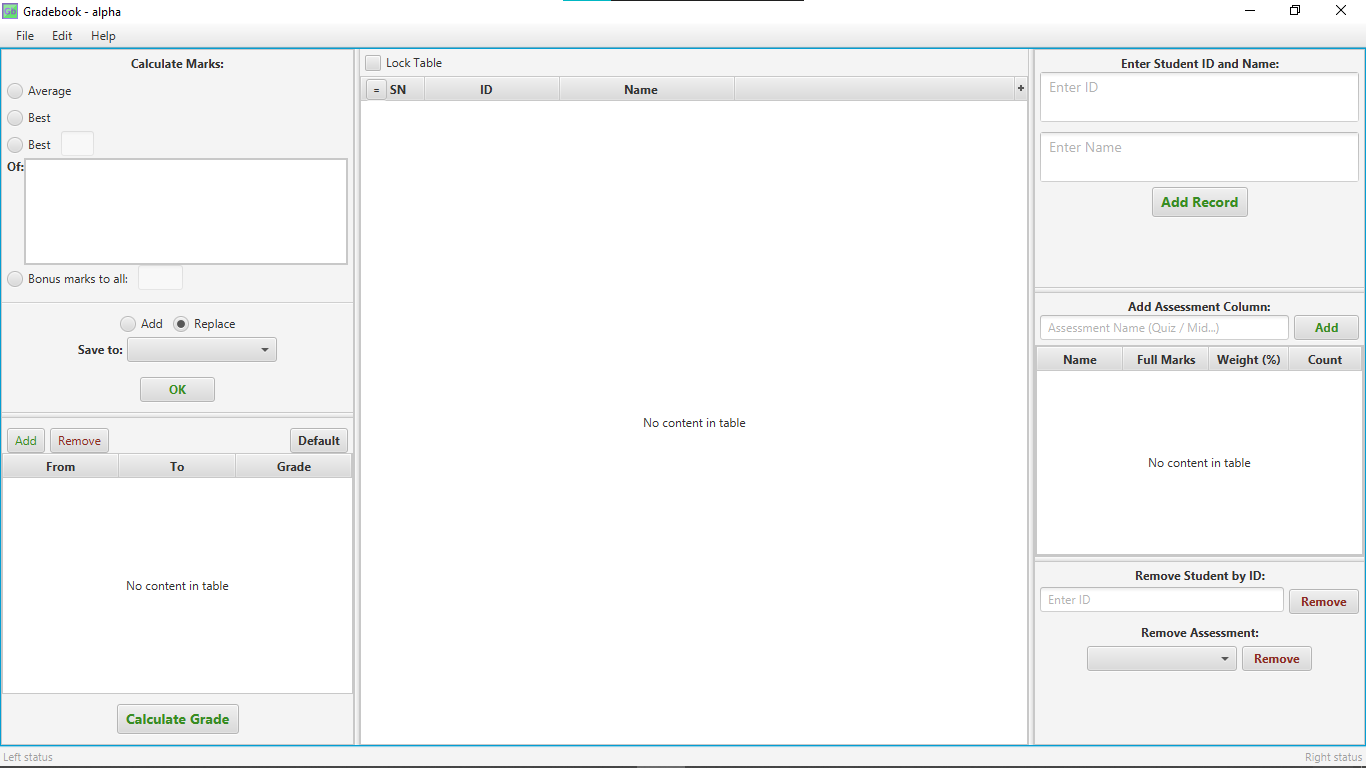
**private ArrayList<String> assessmentNames = new ArrayList<>();**

**private ArrayList<Assessment> assessmentsArrayList = new ArrayList<>();**

These 2 are the global variables in this class and must be kept updated always.

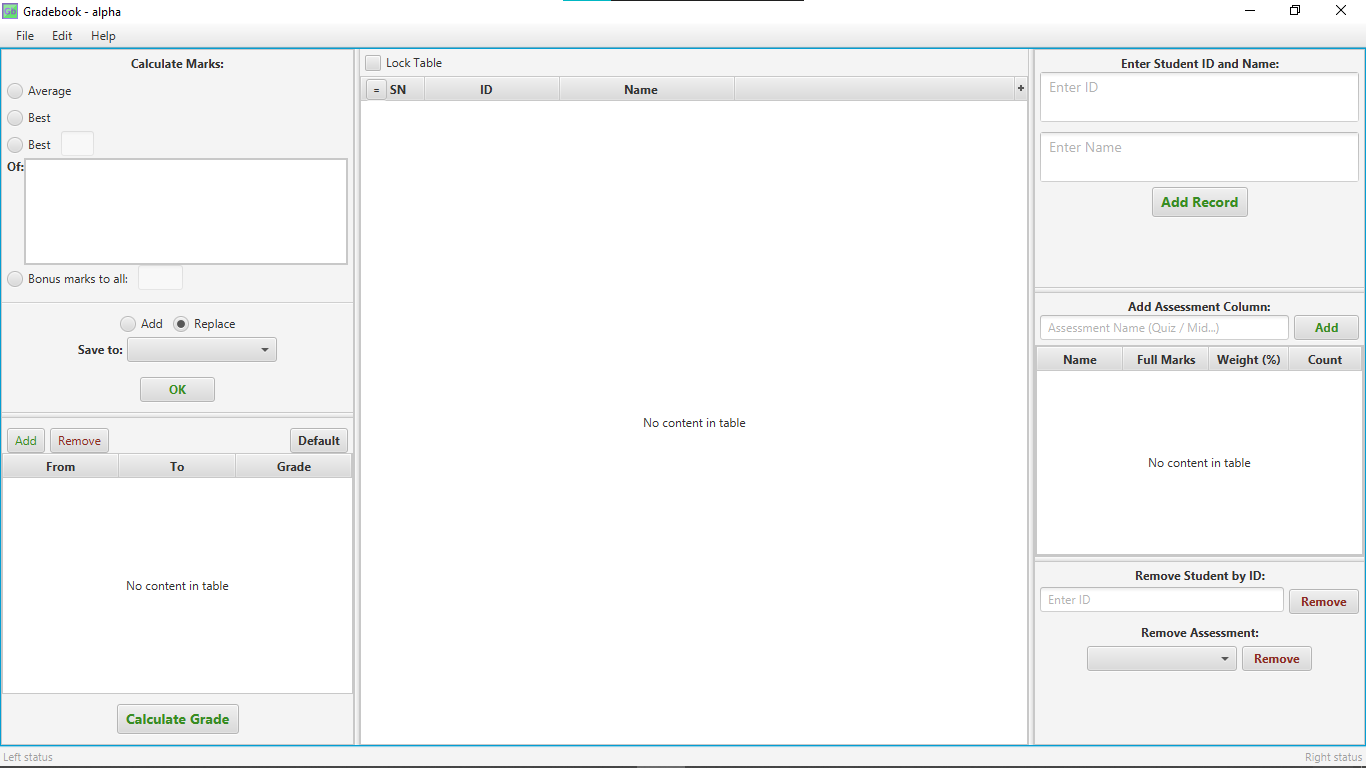
They have no special purpose. I took them just for ease of coding.

**public void addIdName(ActionEvent event):**



When you click the Add Record button shown in the picture it will take ID and Name as input from the above 2 text fields. You can add multiple names and ids at once by separating them with a new-line in the text fields. Input will be saved in 2 string type arrays idStrings and nameString. Now an array of student class will be created. Each element in the idString and nameString will be used to initialize new Student objects and these objects will be saved into the array of students. This array is then used to initialize the observable list of the main table in the middle to show student data.

**public void addAssessment(ActionEvent event):**



When you click the Add button pointed in the picture the text field left to it will be used for taking input. Input will be saved as a string. The string will be added in the  **assessmentNames** global arraylist**.**

**assessmentsArrayList** will also be updated.

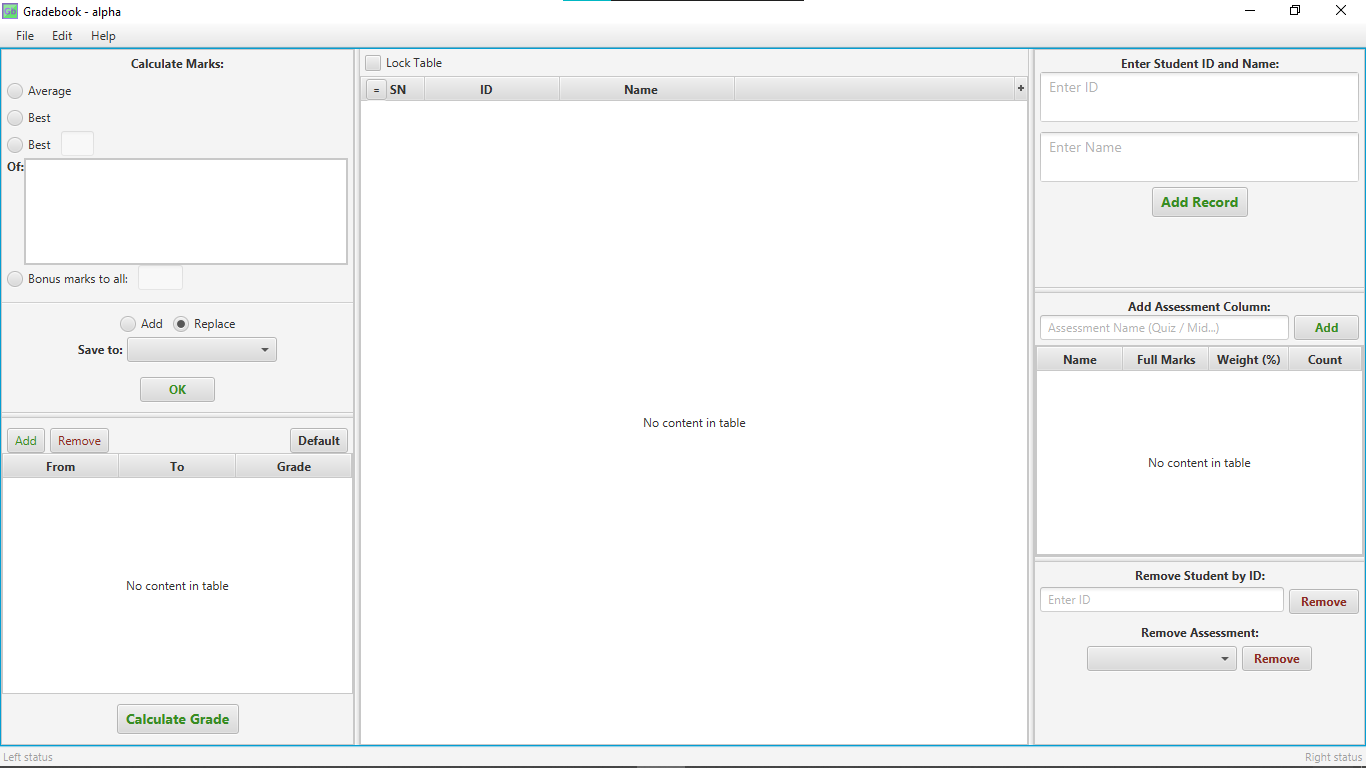
A new tableColumn will be created for adding the assessment column (quiz / mid) in the main table.

Then the **assessmentNames** arrayList will be passed to each Student object in the main table to update the Student class’s Array list.

The remaining code in this method is for making the newly added column cell editable by double clicking and also for initializing the mini table in the “Add Assessment Column” section.

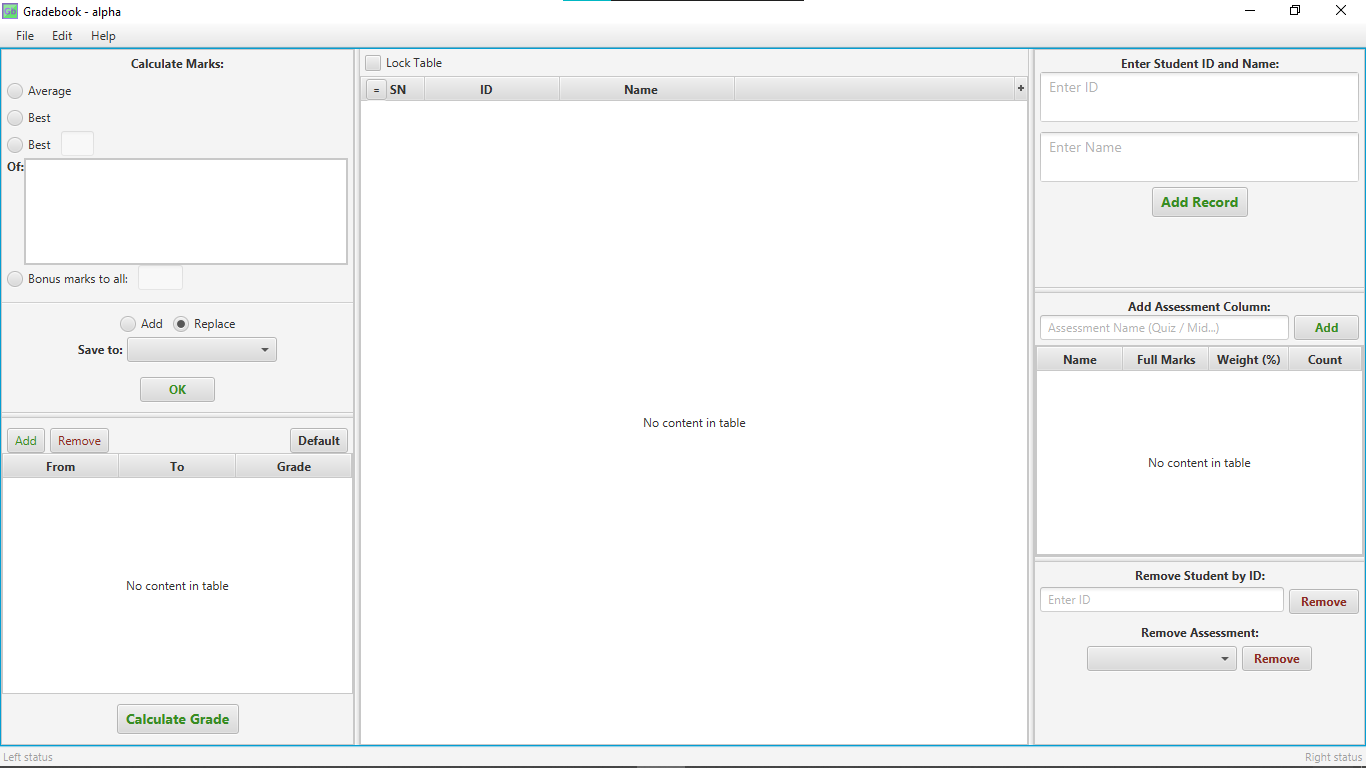
The added Assessment’s name will also be stored in the check boxes and list view shown in red arrows.

**public void recalculateSerialNumber(ActionEvent event):**



If you sort the rows by clicking on their heading the serial number will also be changed. Clicking this button will recalculate the serial number.

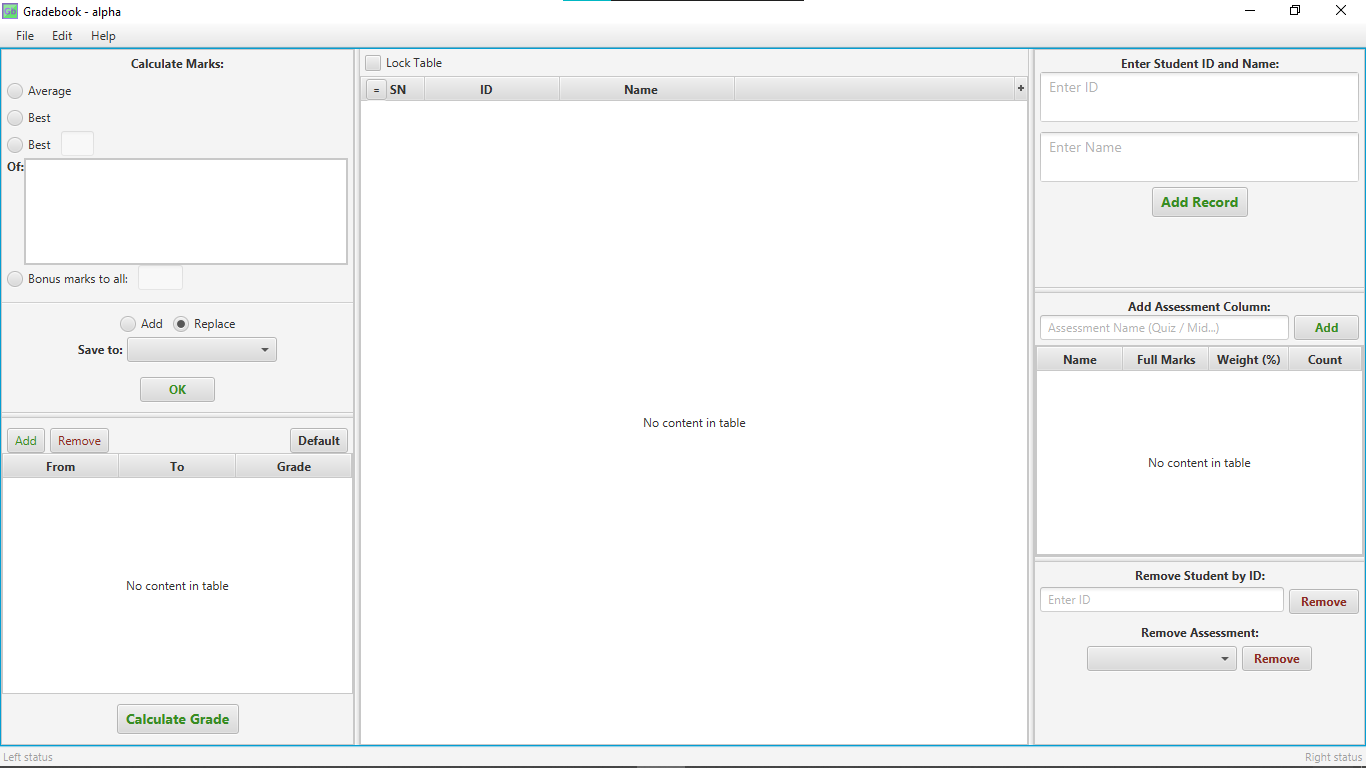
**public void lockTable(ActionEvent event):**



Clicking the Lock Table ckeck box will lock the table and nothing can be added removed then.

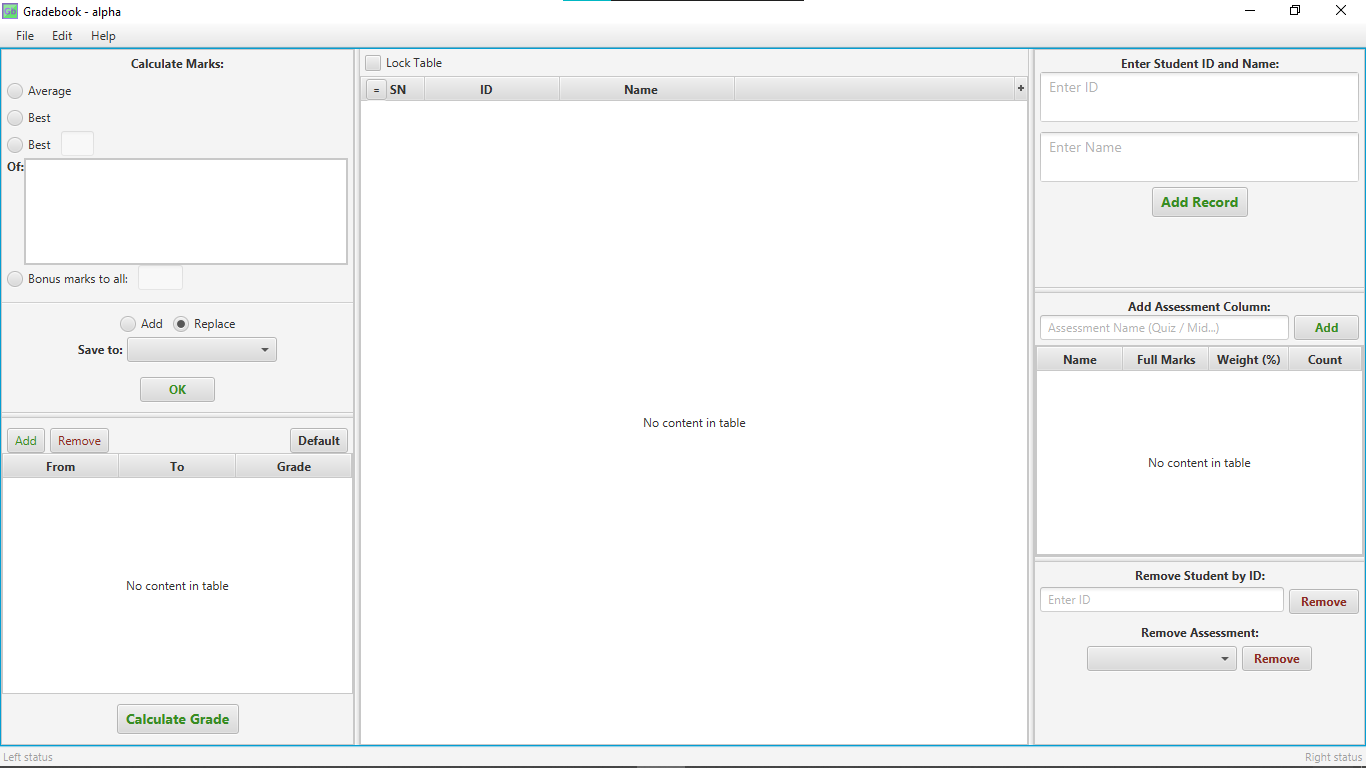
It will alter a Boolean value which will be checked at first of every method. If table is locked no method do anything.

**public void markCalculationProcess(ActionEvent event):**



When any of these 4 radio buttons are clicked this method will be fired. Notice that the last 2 buttons has a text box with them. By default these 2 text boxes will be disabled. This method is will enable them only when the corresponding button is selected.

**public void calculateMark(ActionEvent event):**



Depending on which of the 4 radio buttons you selected earlier the mark calculation will be done. If you choose “Add” then the calculated result will be added to the numbers in the “Save to” column. You have to choose the “save to” column before you click OK. If you choose replace then it will be replaces with the existing value.

You will need to add another column from the “Add Assessment Column” if you want to store the calculated result in a new column.

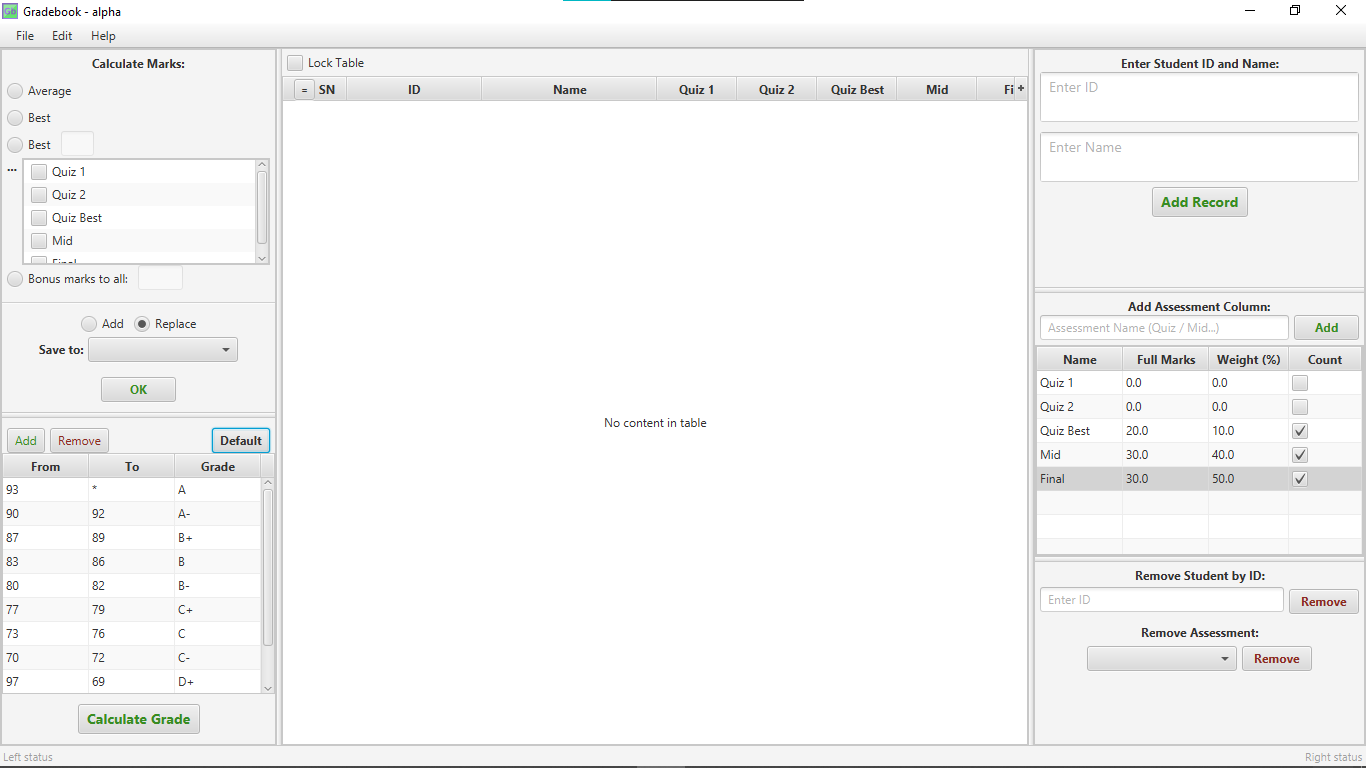
**@Override**

**public void initialize(URL arg0, ResourceBundle arg1):**

This method is for setting the cell values of the all the tables during program launch so that they take input from their corresponding classes.

For example the “SN” column is paired with the “int sn” in the student class. This means int sn will be used as the data in SN column.

**public void calculateGrade();**

****

You will have to click the default button first if you want to get default grading system. Alternatively you can use the Add and Remove button for creating your custom grade.

Then press Calculate Grade.

Only the assessments that you check on the right side will be used for grade calculation.

The Score of one assessment will be counted based on this formula:

Score = Score of one quiz \* Its weight / Its Full mark

Now scores of all the assessment will be added together using loop to get the finalScore of each and every student. The grading table will be used for determining the grade(A/A-/B+).

For grade the formula is:

Math.*ceil*(finalScore) >= min && finalScore <= max

Math.ceil is taken because, if filnalScore is 92.45 for example then it doesn’t satisfy any grading condition as 90-92 is A- and =>93 is A.