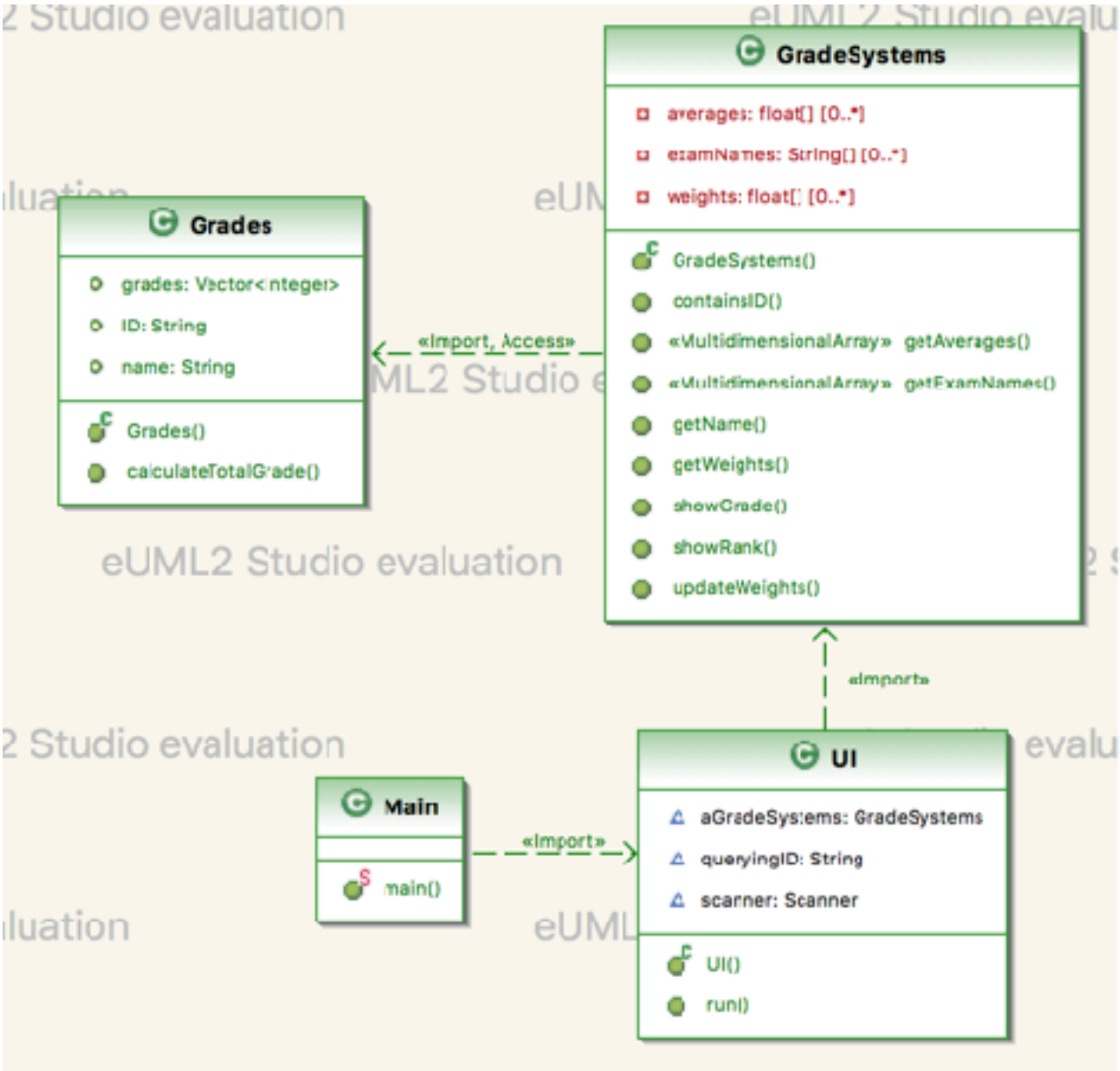
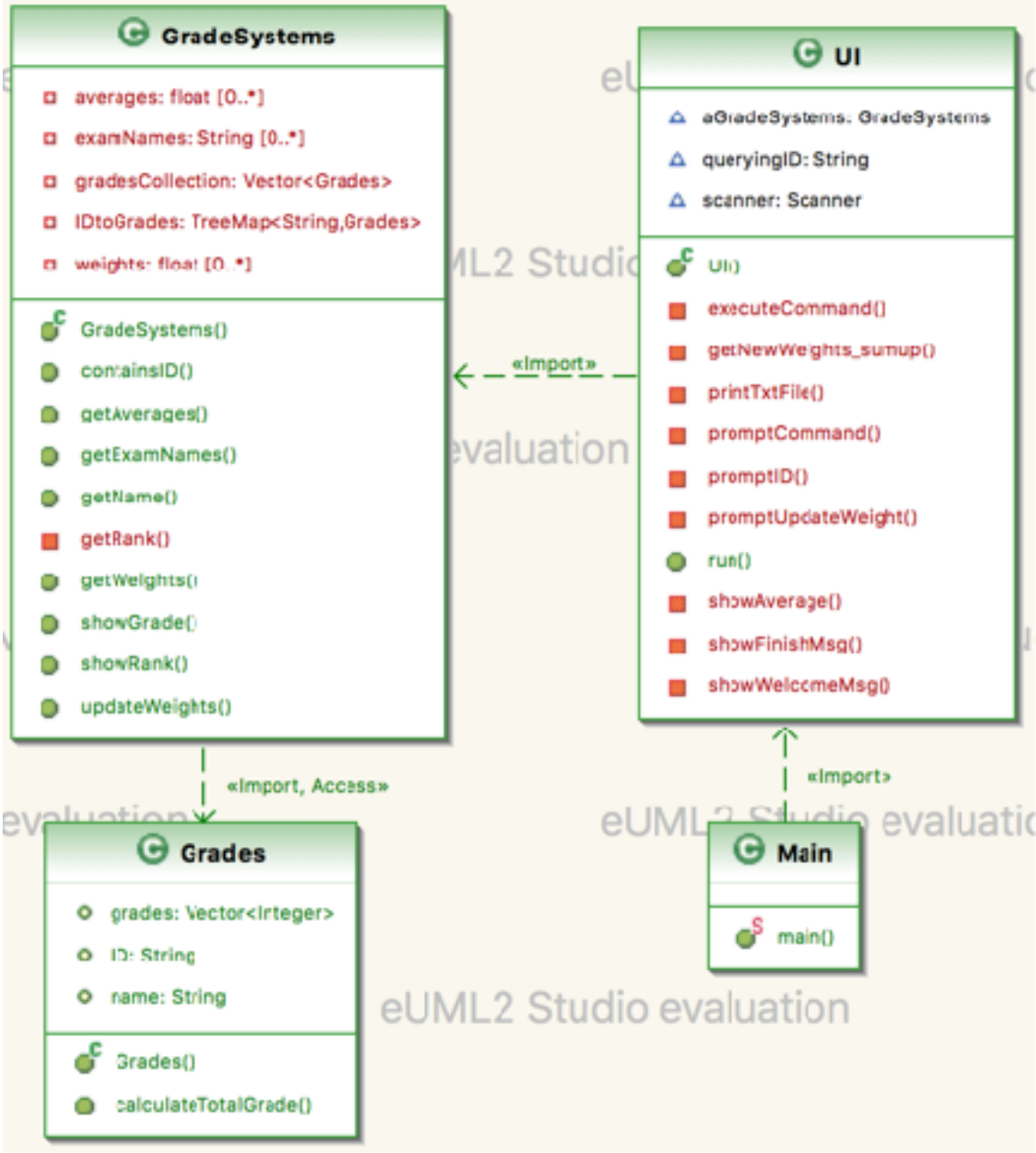
Architectural Design

103062224,103062137 - 19 March 2017

I. Class interface



II. Detailed design



The Grades provide some useful utilities for Grade System to utilize. In fact, it can be used as a inner class of Grade System.

public Grades(String rawInput):

Read a line of rawInput like “985002029 張瑞麟 91 80 91 97 89”, parsing it and store the information, detecting possible format errors as well (NumberFormatException).

public float calculateTotalGrade(float[] weights):

calculate the weighted grade with weights provided by caller.

The Grade System provides lots of API for UI to use, such as get functions, show functions, checking ID exists in system, and update weights with the new_weights passed. Below is the detailed design for each function (get functions are ignored).

public GradeSystems ():

read the file "gradeinput.txt" and detect possible exceptions. For each line read, construct a Grades with the line and put the Grades into a GradeCollection. At the same time, put the grades to a map(search tree) with ID as key, Grades itself as value.

public void updateWeights (float[] newWeights):

copy the newWeights as system's weights.

public boolean containsID(String ID):

use the map to test if the querying ID is in the system.

private int getRank(String ID):

calculate how many people have higher weighted grade compared to the querying ID, and that's the rank.

public void showRank(String ID):

use getRank to get rank of querying ID, and show the rank message on the screen.

public void showGrade(String ID):

get the Grades instance of querying ID, and put the information of this instance on the screen.

public float[] getAverages():

for each exam, sum up the scores over the class and divide it with the number of people in the class. Once it's calculated, it will be stored, and the function will return the stored information when called next time instead of recalculate again.

UI utilize the API provided by Grade System, and only provide run() for main to use. UI prompts user to input commands, execute the corresponding commands, and detect whether it's a illegal command. Below is the whole process of run(), which accept all the scenarios.

```
public void run() {
        while (promptID() == true) {
                if (aGradeSystems.containsID(queryingID) == false) {
                        System.out.println("Sorry, but we don't find ID: " +
queryingID);
                        continue;
                }
                showWelcomeMsg();
                while (promptCommand());
                showFinishMsg();
        }
        System.out.println("Thanks for using grade system!!");
}
```

boolean promptID():

prompt and read ID from user, return false iff reading "Q" or "q".

boolean promptCommand()
void executeCommand(String command):

promptCommand read a line as command, and call executeCommand with the line. executeCommand will do the corresponding command using Grade System's API. promptCommand return false iff reading a "E" command.

void promptUpdateWeight(),
float getNewWeights_sumup(String[] examNames, float[] new_weights):

promptUpdateWeight show the old weights and prompt user to input new weights. Then, call getNewWeights_sumup to read new weights, detect if the input is legal floating number, and sum up the new weights. At the end, test if summation of new weight is 100%.

The other functions are just showing some messages on the screen, and I am not going to give a detailed design about them.