Detailed Design

for

UIC Teamwork Contribution

Assessment System

Version 1.0 approved

Prepared by Yuepeng LONG, Rongkai LIU, Yitao QIU and Hongyue SHEN

AIR

Sunday, April 5, 2020

Table of Contents

Table of Contents ii

Revision History ii

1. Overview 1

1.1 Project description 1

1.2 References 1

1.3 Design purpose 1

2. Overall description 1

2.1 Class diagram 1

2.2 Refinements 1

3. Detailed design 1

3.1 Class diagram 1

3.2 Classes 1

3.2.1 Class 1 */\*Replace with real class name\*/* 1

3.2.2 Class 2 */\*Replace with real class name\*/* 2

3.2.3 Class n */\*Replace with real class name\*/* 2

4. Alternative detailed design (Optional) 2

5. More considerations 2

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Yuepeng LONG, Rongkai LIU, Yitao QIU, Hongyue SHEN | 2020-04-10 | Initial version of this document | 1.0 |
|  |  |  |  |

# Overview

## Project description

This project aims to bring UIC staffs and students a better and more convenient platform to do the record, management and assessment about teamwork contribution. With this application, teachers can easily specify the group allocation, manage the assessments uploaded by students and get the reference of teamwork contribution for grading students, while students can do submission of each assignment or group project, and assess their classmates’ work clearly.

## References

Y. P. Long, et al., “Software Requirements Specification for UIC Teamwork Contribution Assessment System v1.4” Sunday, April 5, 2020.

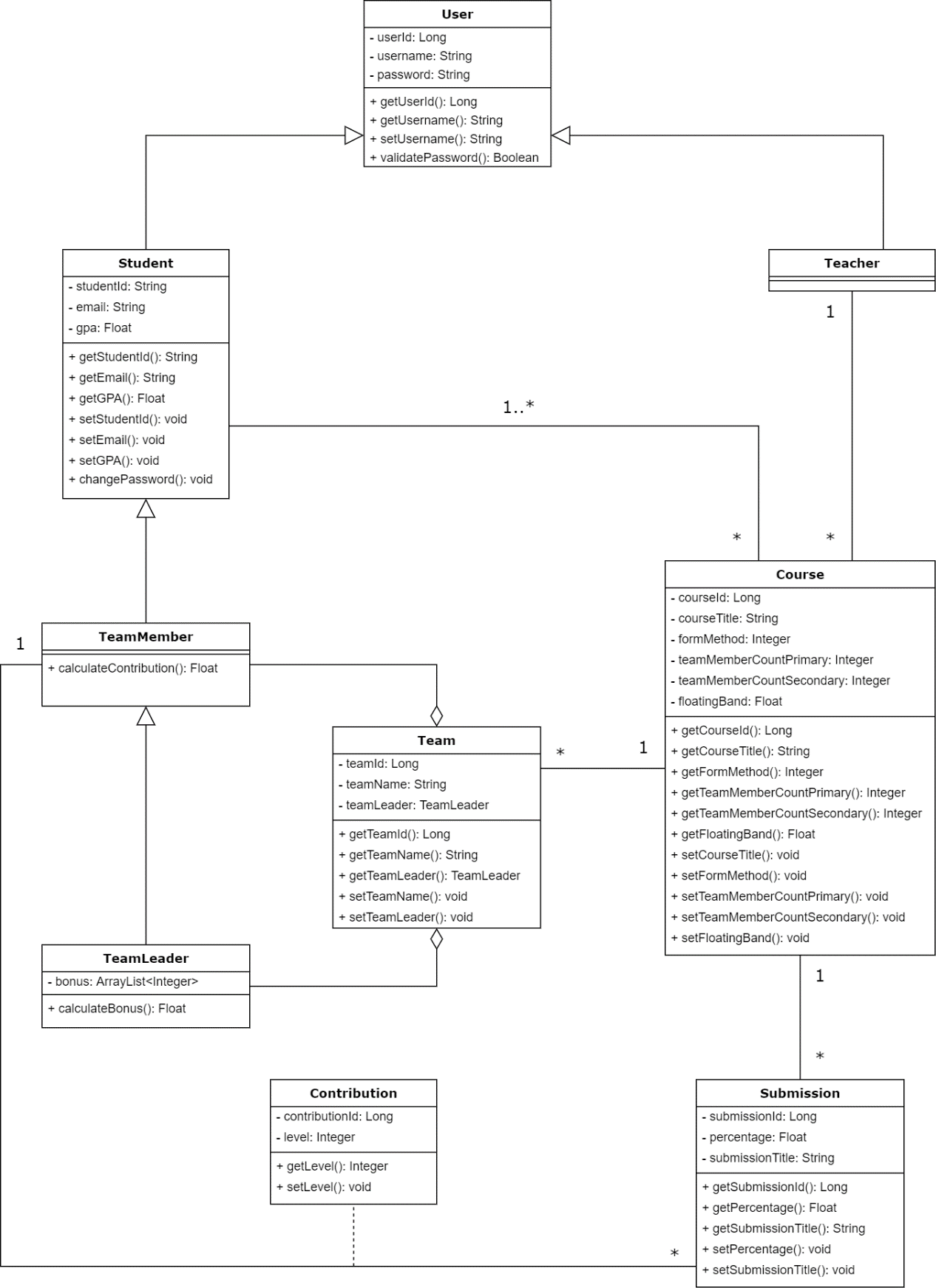
Y. P. Long, et al., “Architecture Design for UIC Teamwork Contribution Assessment System v1.1” Sunday, April 5, 2020.

## Design purpose

This design’s purpose is to improve and refine the previous design for each class in the previous class diagram to give more detailed information for each class, which contributes to the interface design, so when the programmer implement the interface, the ambiguity is reduced and the programmer can understand the details more easily, which can save their time.

# Overall description

## Class diagram



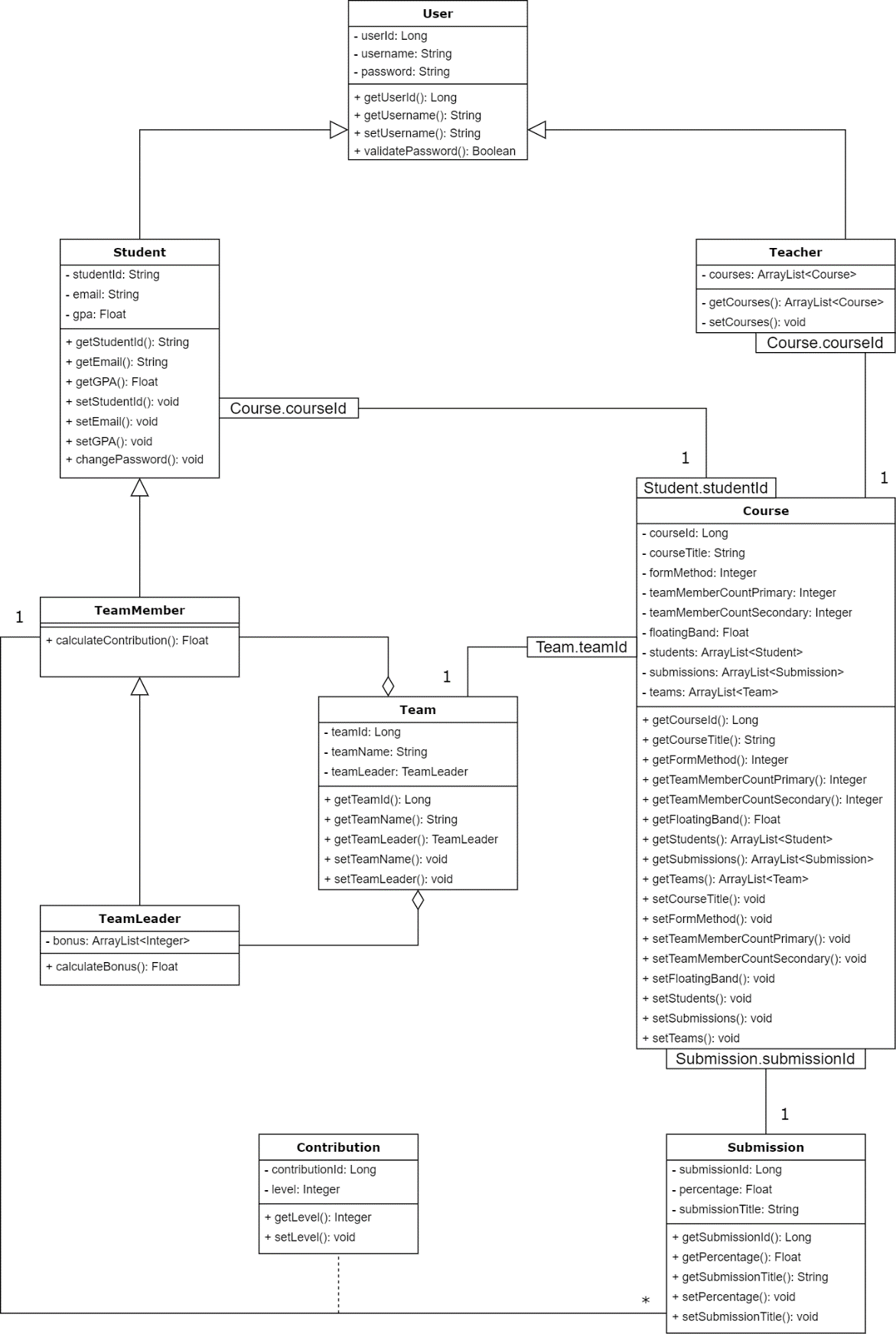
## Refinements

In the refined class diagram, attributes and features of each class with visibility (private or public) and signatures (definition of parameters and type of return value) are specified. Details are shown in part 3.2 of this document.

For restructuring, 5 qualifiers are added in order to change the One-to-Many relationships to One-to-One relationships.

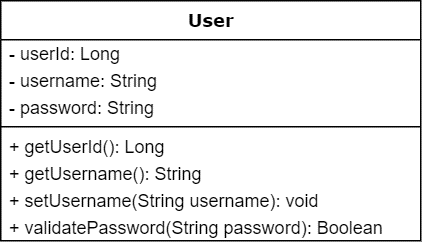
# Detailed design

## Class diagram



## Classes

### User



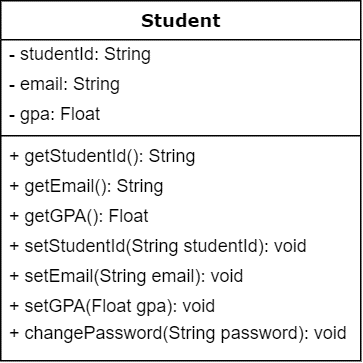
***Explanations***

There is no change for any attributes in the class, since all attributes have already been with types in the last version and no attribute is added or deleted.

For class methods, definition of parameters is added.

1. For method “setUsername()”: add a parameter “username” with type of String.
2. For method “validatePassword()”: add a parameter “password” with type of String.

### Student



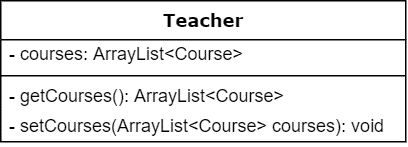
***Explanations***

For all attributes, there is no any change because all attributes have been with type in last version, and all the attributes are necessary, without needing any more attributes.

For all operations, the changes are adding input parameters and type of specific parameters for the class.

1. For operation “setStudentID()”: add a String type input parameter “studentId”.
2. For operation “setEmail()”: add a Sting type input parameter “email”.
3. For operation “setGPA()”: add a Float type input parameter “gpa”.
4. For operation “changePassword()”: add a String type input parameter “password”.

### Teacher



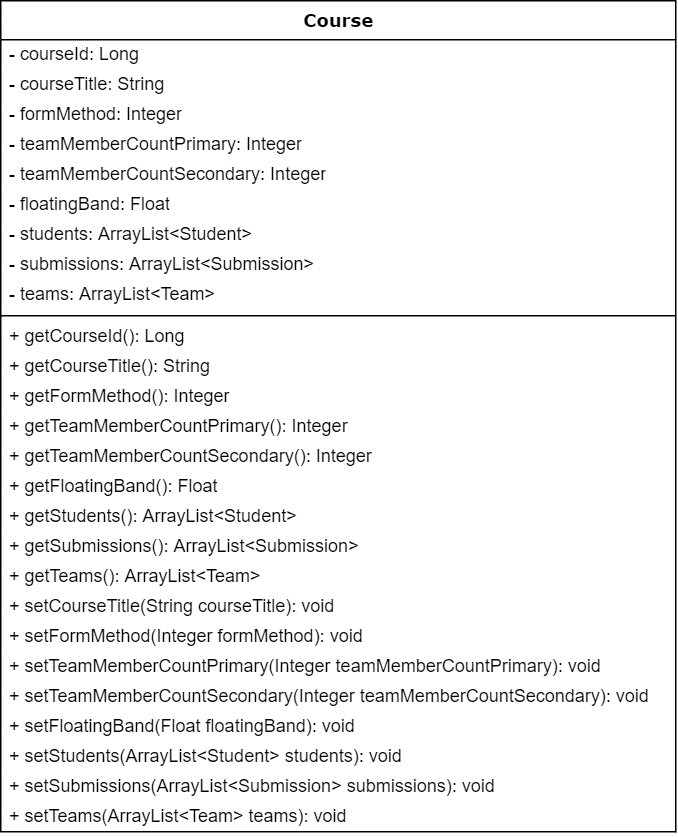
***Explanations***

For all attributes, add a new attribute called “courses” to represent the courses one teacher have been assigned to the teacher.

For all operations, the changes are adding input parameters and type of specific parameters for the class.

1. A new operation “getCourses()” with the return value of type ArrayList<Course> is added.
2. For operation “setCourses()”: add a ArrayList<Course> type input parameter “courses”.

### Course



***Explanations***

For the attributes, we add these new attributes:

1. Add an ArrayList<Student> type attribute “students”
2. Add an ArrayList<Submission> type attribute “submissions”
3. Add an ArrayList<Team> type attribute “teams”

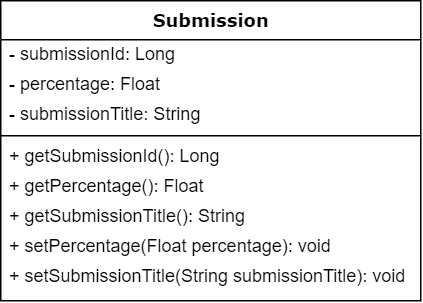
For some operations, we add some input parameters and type of these parameter:

1. For operation “setCourseTitle()”: add a String type input parameter “courseTitle”.
2. For operation “setFormMethod()”: add an Integer type input parameter “formMethod”.
3. For operation “setTeamMemberCountPrimary()”: add an Integer type input parameter “teamMemberCountPrimary”.
4. For operation “setTeamMemberCountSecondary()”: add an Integer type input parameter “teamMemberCountSecondary”.
5. For operation “setFloatingBand()”: add a Float type input parameter “floatingBand”.

For the operations, we add some new operations:

1. Add a public method “getStudents()”, the return type is ArrayList<Student>.
2. Add a public method “getSubmissions()”, the return type is ArrayList<Submission>.
3. Add a public method “getTeams()”, the return type is ArrayList<Team>.
4. Add a public method “setStudents()” with an input parameter “students” of type ArrayList<Student>.
5. Add a public method “setSubmissions()” with an input parameter “submissions” of type ArrayList<Submission>.
6. Add a public method “setTeams()” with an input parameter “teams” of type ArrayList<Team>.

### Submission



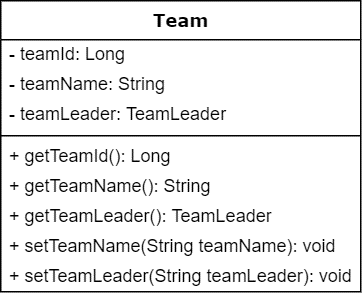
***Explanations***

For all attributes, there is no change because all attributes have been with type in last version and all the attributes are necessary, without needing anymore attributes.

For all operations, we have added parameters to all the “set” functions:

1. For operation “setPercentage()”: add a Float type parameter “percentage”.
2. For operation “setSubmissionTitle()”: add a String type parameter “submissionTitle”.

### Team



***Explanations***

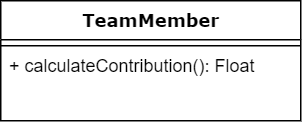
There is not any change for all the attributes, because all the attributes have been written in the architecture design document and it is not necessary to add any new attributes.

For some operations, we add some input parameters and type of these parameters:

1. For operation “setTeamName()”: add a String type input parameter “teamName”.

2. For operation “setTeamLeader()”: add a TeamLeader type input parameter “teamLeader”.

### TeamMember



***Explanations***

For all attributes, there is not any change because all attributes have been with type in last version, and all the attributes are necessary, without needing any more attributes.

For all operations, there are no more operations to be added.

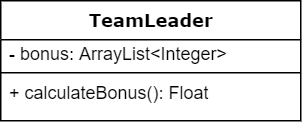
***Constraints***

public Float calculateContribution();

Pre-condition: the related course of the team (which the team member is in) must have at least one Submission.

Post-condition: the return value cannot be less than 0.0.

### TeamLeader

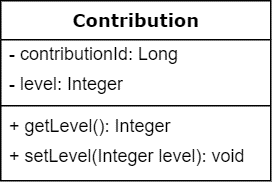


***Explanations***

For all attributes, there is no change because all attributes have been with type in last version and all the attributes are necessary, without needing anymore attributes.

For all operations, there is no change because all operations have been with type in last version and all the operations are necessary, without needing anymore operations.

### Contribution



***Explanations***

For all attributes, there is no change because all attributes have been with type in last version and all the attributes are necessary, without needing anymore attributes.

For all operations, we have added parameters to all the “set” functions:

1. For operation “setLevel()”: add an Integer type parameter “level”.

# Alternative detailed design

N/A

# More considerations

In order to better understand and imagine details of our system design, it is recommended that readers can refer the User Interface part of our “Architecture Design for UIC Teamwork Contribution Assessment System v1.1” as a reference.