



İhsan Doğramacı Bilkent University

Department of Computer Engineering

# CS319 Object Oriented Software Engineering

*ProCheck*

## Design Report Iteration 2

Group 1E

Lara Fenercioğlu - 21802536

Kimya Ghasemlou - 21801412

Sebahattin Utku Sezer - 21802798

Gökhan Taş - 21802136

Bedirhan Sakinoğlu - 21802576

Instructor: Eray Tüzün

Teaching Assistant(s): Elgun Jabrayilzade, Erdem Tuna

# Table of Contents

<b>Introduction</b>	<b>3</b>
Purpose of the system	3
Design goals	3
Usability	3
Reliability	4
Portability	4
Maintainability	4
Performance	5
<b>High-level Software Architecture</b>	<b>6</b>
Subsystem Decomposition	6
User Interface Layer	7
Management Layer	8
Data Layer	10
Hardware/Software Mapping	10
Persistent Data Management	11
Access Control and Security	11
Boundary Conditions	12
Initialization	12
Termination	12
Failure	13
<b>Low-level Design</b>	<b>13</b>
Object Design Trade-offs	13
Final Object Design	16
Packages	17
External Packages	17
Design Patterns	18
Mediator Design Pattern	18
Singleton Design Pattern	18
Class Interfaces	18

Class	19
Project	20
Group	23
User	24
Chat	26
Announcement	27
Calendar	27
Event	28
Assignment	29
Student	30
ArtifactReview	33
PeerReview	34
InstructorAndTAs	34
<b>Improvement Summary</b>	<b>37</b>
<b>Glossary &amp; References</b>	<b>38</b>

# 1. Introduction

## 1.1. Purpose of the system

This application will allow instructors to easily manage the courses that include projects as a task. Also, students will be able to work in groups and manage their work easily thanks to ProCheck (our application name). Furthermore, the main goal will be to provide enough feedback to the project groups so that they can improve their work. By giving users opportunities to both facilitate the management process and develop the projects according to the feedback that was received, one can benefit from this system fully.

## 1.2. Design goals

### Usability

Our program must be user-friendly since it will be used by many students whose first precedence is simplicity. In order to do this, we made our user interface very simple and easy to use. Moreover, it will be comprehensible for users who want to easily find what they need. Our interfaces will be interchangeable. For instance, instructors and students have different interfaces with little changes but it will not affect the simplicity of interfaces. On the other hand, our interfaces will be detailed which is mostly about the process of students' projects.

## Reliability

In order to make our program reliable, we tried to handle exceptions and errors so they won't result in a fatal error which results in a breakdown in the program or data loss. Besides, in every error situation, the user is informed about the problem by proper messages and warnings that explain the problem.

## Portability

Since our application will be a web-based application, we need to make sure that it works in every web browser. Users of this application can use different browsers such as Safari on macOS version 14.0.3, Chrome version 90.0.4430.72, Firefox version 87.0, and also Chrome on Android version 90.0.4430.66 and Safari on iOS version 14.0, therefore, it is important that the user can open this application smoothly and perform his/her actions.

## Maintainability

Since the application will be using the basis of Object-Oriented Programming (OOP), adding new features is going to be simple for any developer by using interfaces or inheritance. Moreover, OOP enables us to implement hierarchy easily by using its features such as polymorphism or inheritance.

## Performance

The application would have a decent response time to not waste the teacher's or students' time, even though performance is not a main concern in the application. It will have a response time under 5 seconds and will display the changes such as uploading homework or sending a message in less than 5 seconds.

## 2. High-level Software Architecture

### 2.1. Subsystem Decomposition

Our goal is to ensure that further enhancements will be implemented in an easy manner without touching existing functionalities. So, we have decided to use a 3-layer architectural style which are UserInterface Layer, Management Layer, and Data Layer. Our main focus is to maximize the coherence in between components while minimizing the coupling in between the components.

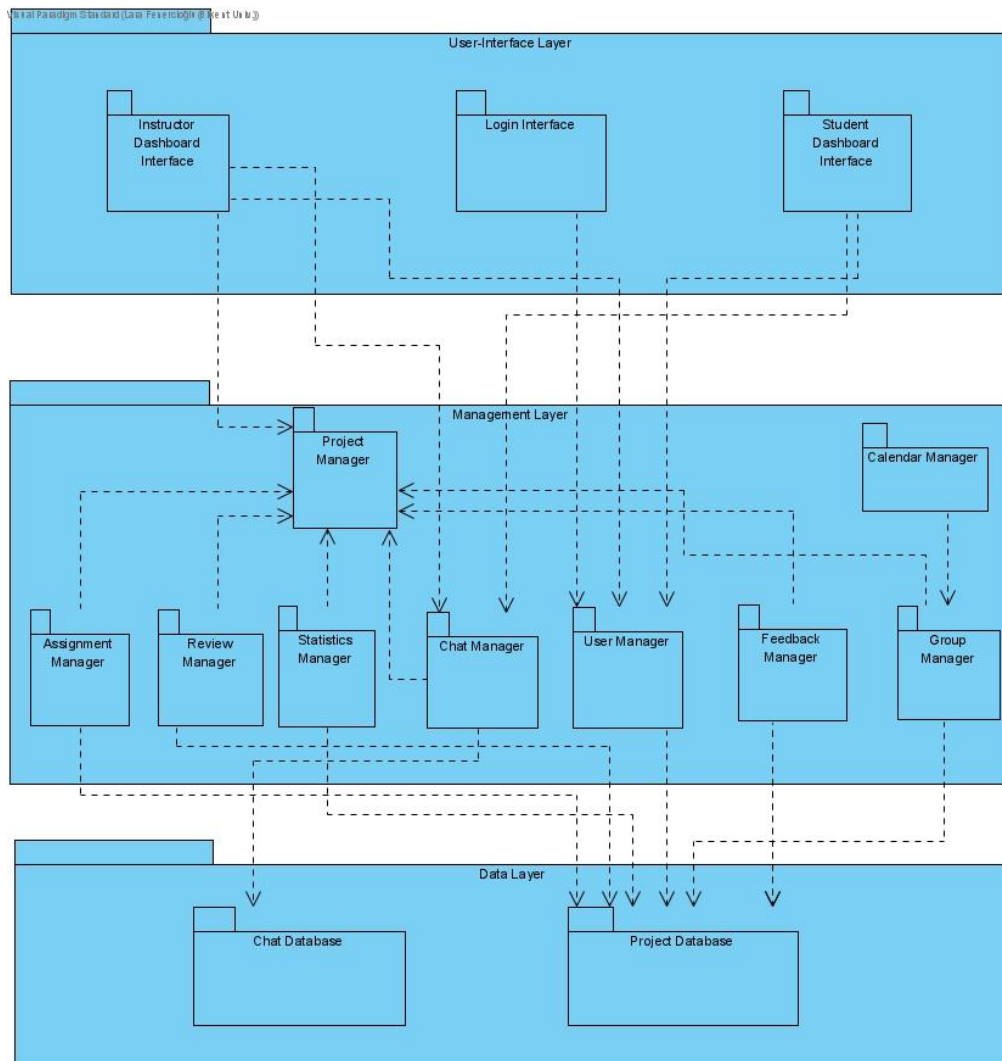


Figure 1 <https://imgur.com/B5gRKj0>

### 2.1.1. User Interface Layer

This layer is responsible for connecting users to the system with simple interfaces. It helps to visualize the system with UI components. This layer is connected to the Management Layer to modify the system or data. It has no access to data or functions directly.



**Login Interface:** It is the first interface which the users will see and enter their email - password to login into the system.

**Instructor Dashboard Interface:** After the user entered, according to the role of s/he a dashboard interface will appear. If the user is an instructor or TA this dashboard interface will be used.

**Student Dashboard Interface:** If the user is a student this interface will appear and will be used. There are small differences between instructor and student dashboards. Thus, we decided to use two different interface components.

### 2.1.2. Management Layer

This layer is responsible for doing calculations according to the user's actions, unseen by the user. The layer does the calculations according to user inputs and data according to the task.

**Project Manager:** Project information, fundamental functionalities of a project will be kept here.

**Statistics Manager:** Statistics about the course will be accessed through this package.

**Calendar Manager:** Functionalities related to calendar will be accessed from this package such as add event or edit calendar.

**User Manager:** User related activities such as chat will be kept here for further access.

**Assignment Manager:** It is the Manager which will handle the assignment submissions from students to the assignments created by the instructor. It will send assignments to the database.

**Review Manager:** It is the manager which will handle the Peer and artifact reviews submitted by students. It will be stored in a database.

**Chat Manager:** The manager who manages the chat messages from students and instructors.

**Feedback Manager:** This is the manager which will handle the feedback given by the instructor, store them in the database.

**Group Manager:** Is the manager which will handle the group creation, join requests, invitations and will store the information of the groups into database.

### 2.1.3. Data Layer

This layer will be responsible for hiding the data from the other layers and saving it. All of the information will be stored in a database.

**Chat Database:** Messages between users will be kept in this database. Managing the communication will be carried out here.

**Project Database:** All things related to the project will be saved in this database.

## 2.2. Hardware/Software Mapping

Our application does not require any specific hardware component to run. Since the application will not have complicated interactions between objects, it certainly does not require any high-end hardware. However, the user will need a keyboard, touchpad, or a mouse to interact with the application if the user uses a computer. Additionally, this application will be implemented by using Java, Javascript programming languages and HTML, CSS markup languages. We are planning to use Java Spring Framework for the back-end development. Since Spring is very popular and used in web development, we can easily implement it and find much more resources than other frameworks. Also we will use MongoDB which is a NoSQL database in order to store the data. The detailed information about the database is explained below. Since it will be

web-application, the user doesn't need to install anything, the user only needs to load the website. But the versions of the browsers are important and they are Safari on macOS version 14.0.3, Chrome version 90.0.4430.72, Firefox version 87.0, and also Chrome on Android version 90.0.4430.66 and Safari on iOS version 14.0.

### 2.3. Persistent Data Management

The database will hold usernames and passwords in order to login the system. Groups, group members will be held to form project groups. Project information, class statistics such as grades, averages will be held to be used in a graph which will show the average of the class. Artifact submissions, reviews about them and feedback about them, instructor announcements will be held to inform both students and instructor/TA. Participants and channels will be held for the chatting feature of the application. Also messages from participants will be stored as well. Since the calendar will be used in the dashboard, event related information such as dates and event topic will be stored.

### 2.4. Access Control and Security

In our application, no user can access other user's private information such as their password. All the information about users will be held on a database. No user can reach these data that are in the database, so they cannot manipulate the data directly. The database will keep every user's

data including their project papers and chat logs. The application will not have a user authentication system. Meaning that, if a user shares their user information with someone, they will be able to reach their account. This situation results in weakness on the security side. Additionally, some functionalities can be only done by instructors, for instance announcing, changing project settings, grading, and editing assignment operations. On the other hand, forming groups, peer reviewing, artifact reviewing, and submitting assignments are special to each student.

## 2.5. Boundary Conditions

### 2.5.1. Initialization

Our game will not require any installation process in order to enter. For initialization, the users should have an internet connection and load the page successfully. Then users must be registered to the system in order to log in. After that, the users can enter the system and focus on their projects, group formations, etc.

### 2.5.2. Termination

When the user wants to exit the program, s/he can log out or simply close the application window. If the user closes the window, the program will automatically log out from the user's account in order to prevent the user's account to remain logged in public computers after closing the program.

### 2.5.3. Failure

Loss of network connection can lead to some major problems. If the user loses the internet connection while uploading something, sending a message, reviewing an artifact, and other similar actions, the data may not be saved.

If the user attempts to log out or close the window while using any function of the application, data will not be saved since the action is not finished yet.

## 3. Low-level Design

### 3.1. Object Design Trade-offs

#### **Usability vs Functionality:**

Usability is one of our biggest concerns since this program will be mostly used by students that are mostly impatient and want to easily find what they want. Also our program must be clear and simple since it focuses on a specific purpose. Therefore, we focus on more usable programs whereas we make a sacrifice from functionality. Our program is about projects and thus, limited but enhanced functions will be enough for students and instructors to develop a project and assess it.

**Efficiency vs Portability:**

Since it will be a web based application portability is not a concern and it can be used by either a computer, Android, or iOS phone through different browsers such as Chrome, Firefox, Safari, etc. . The efficiency may decrease due to portability of the application because it will not be designed specifically to the platforms such as phone browsers.

**Performance vs Security:**

Security is not the one of the main concerns of this project. Although we want our application to be secure, we did not want to compromise on performance. Since our application is going to be used on a very large variety of devices, and the application has different kinds of tasks, we prioritize performance over security; because we wanted to facilitate users' experience.

**Reliability vs Cost:**

Our program will handle most of the exceptions and errors without any damage and notify users with warnings and messages, therefore, reliability is one of the aspects that we will focus on. In order to achieve this we will spend many hours and try various test programs to detect errors and handle them in a proper way which means its cost will increase.

**Maintainability vs Cost:**

We are planning to develop a maintainable program since our product can be used by different courses for different assignments and different purposes. Therefore we have to spend more time to add changes easily which will increase the cost of our project.



## 3.2. Final Object Design

Visual Paradigm Standard Data Representation Notation

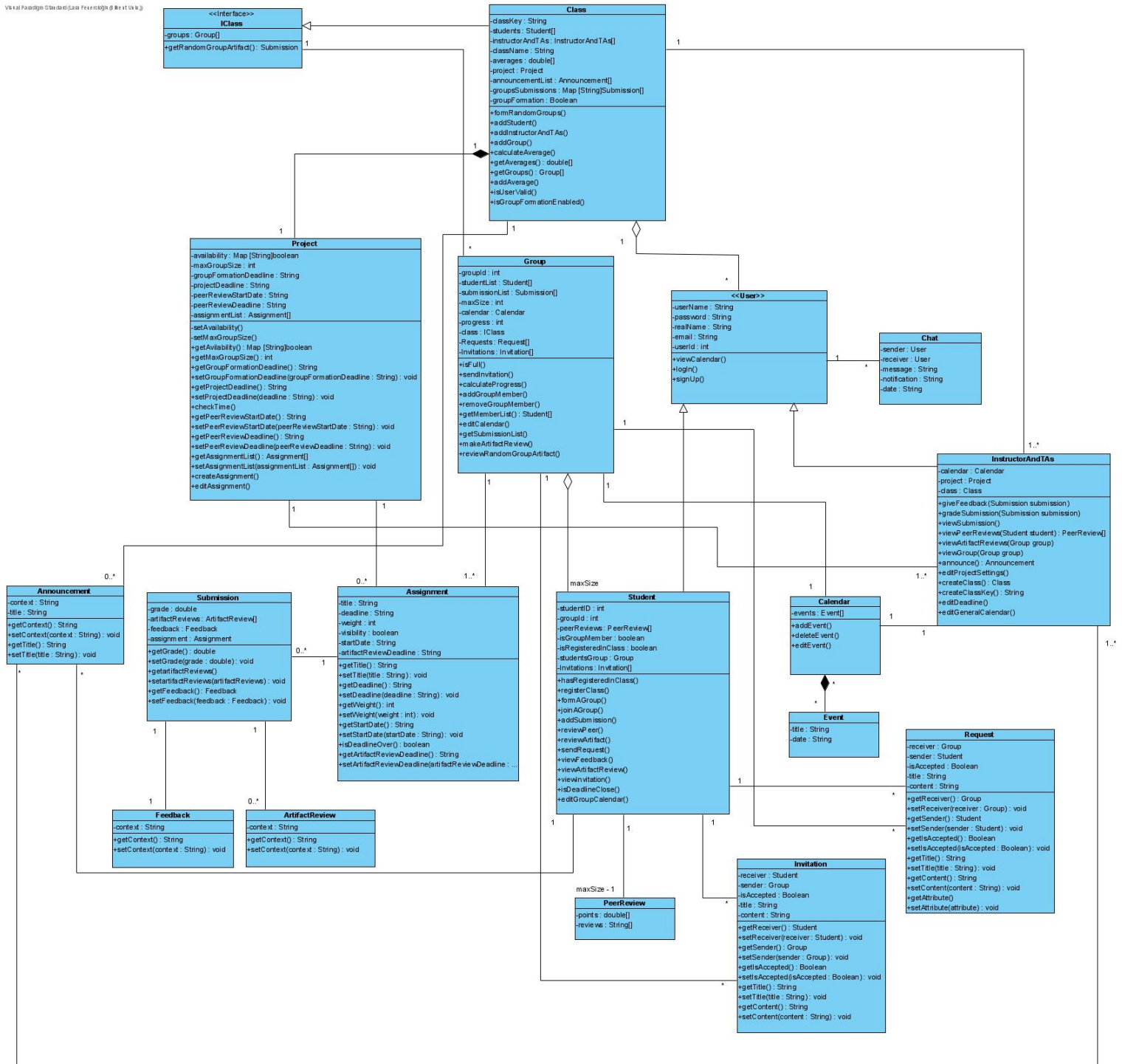


Figure 2 <https://imgur.com/1F2SgVX>

### 3.3. Packages

We will be using external packages which are introduced by external libraries such as javax. By using these we will be able to easily use classes that we need.

#### 3.3.1. External Packages

**java.util:** Data structures will be used from this package and also java.util.random will be used in order to give a random class key while opening a class[3].

**com.google.api.services.calendar.Calendar:** This package is used to manipulate events on a calendar. A template of a calendar will be used[5].

**javax.websocket:** This package is used for creating lifecycle endpoint classes to implement the chat in the web application[2].

**org.springframework.boot:** Since we will use Spring Framework, we will be using this package[6].

**javax.servlet:** This package includes classes and interfaces that describes and introduces servlet classes and the runtime environment[4].

## 3.4. Design Patterns

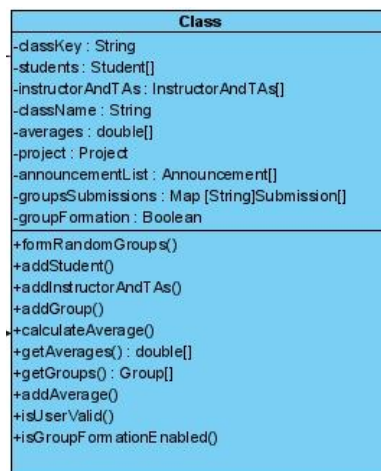
### 3.4.1. Mediator Design Pattern

In this project we are using the Mediator design pattern because this pattern avoids direct communication between the objects and forces them to collaborate only with this mediator object. One example is the IClass class.

### 3.4.2. Singleton Design Pattern

We are also going to use the Singleton design pattern in order to create an object while making sure that only one object gets created. These classes will provide a way to access its only object which can be accessed without needing to instantiate the object of the class.

## 3.5. Class Interfaces



### 3.5.1. Class

This is the class which contains attributes and operations related to Class.

#### **Attributes:**

- **private String classKey:**  
Holds the key of the class to register the course
- **private Students[] students:**  
Contains the students in the class.
- **private InstructorAndTAs instructorAndTAs:**  
Holds instructor and TAs of the class.
- **private Group[] groups:**  
Contains the groups that are formed in the class.
- **private String className:**  
Holds the name of the course.
- **private double[] averages:**  
Holds the averages of assignments.

#### **Operations:**

- **public boolean formRandomGroups():**  
A method which creates random groups if there are still students without any group.
- **public boolean addStudent():**  
Adds a new student to class.
- **public boolean addInstructorAndTAs():**

Adds instructor and TAs to class.

- **public boolean addGroup():**

Adds a new group to class.

- **public double calculateAverage():**

Calculates average of an assignment.

- **public double getAverages():**

Returns average of an assignment.

- **public void setAverages():**

Inserts calculated average to averages[] array.

Project
-availability : Map [String]boolean -maxGroupSize : int -groupFormationDeadline : String -projectDeadline : String -peerReviewStartDate : String -peerReviewDeadline : String -assignmentList : Assignment[]
-setAvailability() -setMaxGroupSize() +getAvailability() : Map [String]boolean +getMaxGroupSize() : int +getGroupFormationDeadline() : String +setGroupFormationDeadline(groupFormationDeadline : String) : void +getProjectDeadline() : String +setProjectDeadline(deadline : String) : void +checkTime() +getPeerReviewStartDate() : String +setPeerReviewStartDate(peerReviewStartDate : String) : void +getPeerReviewDeadline() : String +setPeerReviewDeadline(peerReviewDeadline : String) : void +getAssignmentList() : Assignment[] +setAssignmentList(assignmentList : Assignment[]) : void +createAssignment() +editAssignment()

### 3.5.2. Project

It is used for projects that will be done by students.

#### Attributes:

- **private Map[String]boolean availability:**

Decides which submission is visible.

- **private int maxGroupSize:**

Holds the maximum group size which will be determined by the instructor.

- **private String groupFormationDeadline:**

Holds the final date to form the groups.

- **private String projectDeadline:**

Holds the deadline of the project.

- **private String peerReviewStartDate:**

Holds the start date to review group members.

- **private String peerReviewDeadline:**

Holds the final date to review group members.

**Operations:**

- **public void setAvailability():**

Sets the availability of the submission.

- **public void setMaxGroupSize():**

Sets the maximum size of the groups.

- **public boolean getAvailability():**

Gets the availability of the submission.

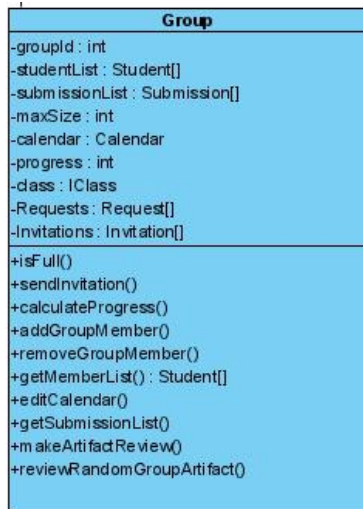
- **public int getMaxGroupSize():**

Gets the maximum size of the groups.

- **public String getGroupFormationDeadline():**

Gets the deadline to form groups.

- **public void setGroupFormationDeadline(String groupFormationDeadline):**  
Sets the deadline to form groups.
- **public String getProjectDeadline():**  
Gets the deadline of the projects.
- **public void setProjectDeadline(String deadline):**  
Sets the deadline of the projects.
- **public boolean checkTime():**  
Checks the remaining time to complete the project.
- **public String getPeerReviewStartDate():**  
Gets the start date of the peer review.
- **public void setPeerReviewStartDate(String peerReviewStartDate):**  
Sets the start date of the peer review.
- **public void getPeerReviewDeadline():**  
Gets the deadline of the peer review.
- **Public void getPeerReviewDeadline(String peerReviewDeadline):**  
Gets the deadline of the peer review.



### 3.5.3. Group

#### Attributes:

- **private int groupId:**

Each group will have a unique Id.

- **private Student[] studentsList:**

The list of members of the groups.

- **private Assignment[] assignmentList:**

The list of assignments done by the group.

- **private int maxSize:**

Size of the group.

- **private Calendar calendar:**

The calendar object of the group which will content the events.

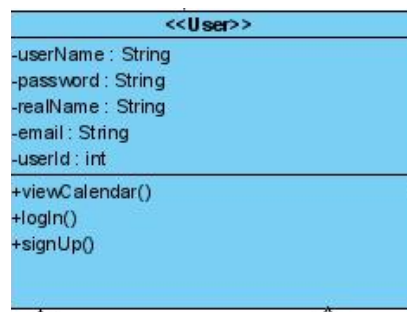
- **private int progress:**

Percentage of progress of work done by the group based on total workload.



### Operations:

- **public boolean isFull():**  
Checks if maximum group size is reached.
- **public boolean sendInvitation():**  
Method for sending an invitation to an individual.
- **public double calculateProgress():**  
Method for calculating overall progress of the project.
- **public boolean addGroupMember():**  
Adds a student to the group as a member.
- **public boolean removeGroupMember():**  
Removes a student from the group and returns true if successful  
false if not successful.
- **public Student[] getMemberList():**  
Return the list of members of the group.



#### 3.5.4. User

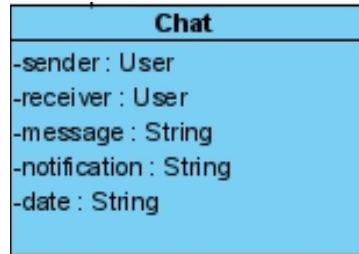
This class will be used to define users of the system who have to be registered.

### **Attributes:**

- **private String userName:**  
Holds the name of the user.
- **private String password:**  
Holds the password of the user.
- **private String realName:**  
Holds the real name of the user.
- **private String email:**  
Holds the email of the user.
- **private int userId:**  
Holds the ID of the user.

### **Operations:**

- **public isValid():**  
Checks if userName and password is valid.
- **public editCalendar():**  
Method for editing calendar to add new events.
- **public viewCalendar():**  
Method for displaying calendar.
- **public login():**  
Method for logging in to the program.
- **public signUp():**  
Method for creating a new account.



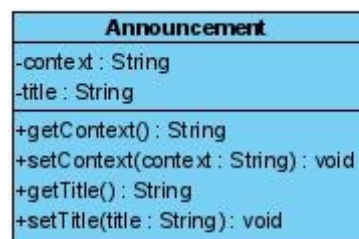
### 3.5.5. Chat

This class will be used in order to store chat related information.

#### Attributes:

- **private User sender:**  
Holds the sender of a message
- **private User receiver:**  
Holds the receiver of a message
- **private String message:**  
Holds the message as a String
- **private String notification:**  
Holds the chat notification as a String
- **private String date:**  
Holds the date of a message

#### Operations: -



### 3.5.6. Announcement

This class will be used in order to show an announcement.

#### Attributes:

- **private String context:**

Attribute will store the text related to the announcement.

- **private String title:**

Attribute will store a name which is about an announcement or feedback's topic.

#### Operations:

- **getContext():** Getting the context of an announcement.
- **setContext(context:String):** Setting the context.
- **getTitle():** Getting the title of an announcement.
- **setTitle(title:String):** Setting the title.



### 3.5.7. Calendar

This class will be used for the calendar.

#### Attributes:

- **private Event [] events:** Calendar holds events as an array

### Operations:

- **addEvent():** This method is used to add an event to a calendar.
- **deleteEvent():** This method is used to delete an event.
- **editEvent():** This method is used to edit a calendar.



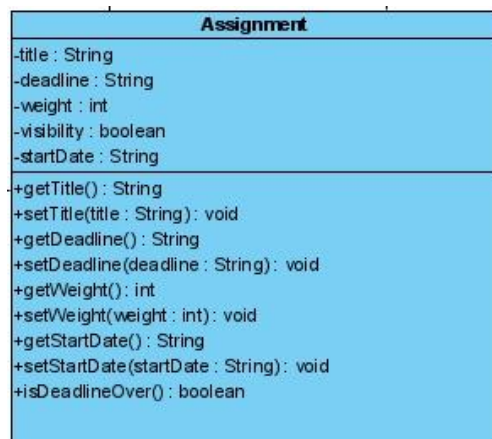
### 3.5.8. Event

This class will be used for events that will be displayed in the calendar.

### Attributes:

- **private String title:**  
Holds the title of the event which is announced by instructor or TA.
- **private String date:**  
Holds the date of the event.

### Operations:



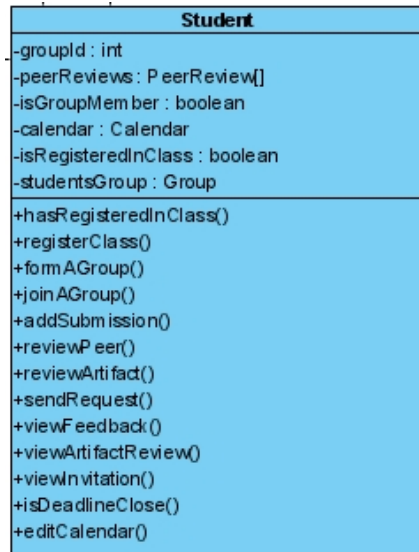
### 3.5.9. Assignment

This class will be used for assignments.

#### **Attributes:**

- **private String title:**  
Title of the assignment.
- **private String deadline:**  
Deadline of the assignment.
- **private double grade:**  
Grade of the assignment.
- **private int weight:**  
Weight of the assignment.
- **private boolean visibility:**  
Is true if it is visible, false if not visible.
- **private ArtifactReview artifactReview:**  
The artifact review of the assignment.
- **private boolean isCompleted:**  
Is true if it is completed, false if is not completed.
- **private String startDate:**  
The start date of the assignment.

#### **Operations:**



### 3.5.10. Student

This class is used to define students.

#### Attributes:

- **private groupId:**

This will store the group id of the student's belonging group.

- **private peerReviews:**

Since each student will have a peer review made by the group members, their reviews will be stored in this PeerReview array, this array will be made of PeerReview objects.

- **private isGroupMember:**

His attribute will be used in order to specify which students have a group or not.

- **private calendar:**

Each student will have their own calendar which they can add their own event to.

- **private isRegisteredInClass:**

In order to connect a student to a class, s/he has to register into a class since registering a class will enable the student to perform specific actions, so we need to check whether the student belongs to a class or not.

- **private studentsGroup:**

In order to access a student's belonging group, there has to be a Group object so that the student can reach detailed information such as to his/her own group member list.

### **Operations:**

- **public boolean hasRegisteredInClass():**

This operation will check whether the student has registered in a class.

- **public void registerClass():**

Since a student must register a class, s/he has to register the class by using this operation. It will basically get the class key and enter that key in order to register a specific class.

- **public void formAGroup():**

Students will form groups, so each student has the right to form a group with initially no participants. Afterwards, a student without a group can join this newly created group.

- **public void joinAGroup():**



If a student has no group and wants to join a group then this operation will be used. Students will be able to already create groups and then will call joinAGroup(). Afterwards, this operation will send a notification to that group to notify the members. These members will then decide to accept this invitation.

- **public void addSubmission():**

This method is for students to upload artifact submissions.

- **public void reviewPeer():**

Each student has to do a peer review at the end of the project.

Each student will review the members of the group.

- **public void reviewArtifact():**

Students will be able to review the artifacts of different groups.

These artifact reviews will be distributed to each group randomly after each artifact submission.

- **public void sendRequest():**

Sends the request to join a group.

- **public void viewFeedback():**

Method for viewing feedback.

- **public void viewArtifactReview():**

It is used to view the artifact review which is done by other groups.

- **public void viewInvitation():**

Method to view invitations sent by a group.

- **public boolean isDeadlineClose():**

Is true if the deadline is close and false if it is not, will be used to show higher priority assignments in a different way.

- **public void editCalendar():**

Each student can edit the calendar and add new events.



### 3.5.11. ArtifactReview

This class is used for the peer review part.

#### **Attributes:**

- **private deadline:**

It will store the deadline of an artifact review.

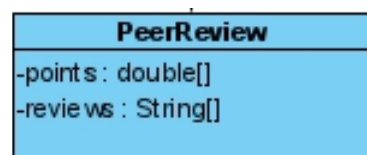
- **private startDate:**

It will keep the date of which the artifact review will begin to get submissions.

- **private context:**

This will store a short explanation of what the artifact review is about.

#### **Operations:**



### 3.5.12. PeerReview

This class is used for the peer review part.

#### Attributes:

- **private double[] points:**

This will store the scores, points related to a single peer review.

- **private String[] reviews:**

It will store the textual reviews about a single peer review.

#### Operations:

InstructorAndTAs
-calendar : Calendar -project : Project -class : Class
+giveFeedback(Submission submission) +gradeSubmission(Submission submission) +viewSubmission() +viewPeerReviews(Student student) : PeerReview[] +viewArtifactReviews(Group group) +viewGroup(Group group) +announce() : Announcement +editProjectSettings() +createClass() : Class +createClassKey() : String +editDeadline() +editGeneralCalendar()

### 3.5.13. InstructorAndTAs

This is the class which contains attributes and operations related to Instructor or TA's.

#### Attributes:

- **private Calendar calendar:**

It is the Calendar object specialized to each group. Will contain events of the group.

## **Operations:**

- **public boolean createAssignment():**  
The teacher will create a new assignment.
- **public boolean giveFeedback():**  
The teacher will give feedback to the student's submission.
- **public boolean gradeSubmission():**  
The teacher will submit the grade for the submission.
- **public Assignment viewSubmission():**  
The teacher will view the submission made by a specific student.
- **public ArtifactReview viewArtifactReview():**  
The teacher will view the artifact review of a specific group.
- **public Group viewGroup():**  
The teacher will view the members of a group.
- **public boolean announce():**  
The teacher will make announcements to the class.
- **public boolean editProjectSettings():**  
The teacher can edit project settings.
- **public boolean createClass():**  
The teacher will create a new class at the beginning of the semester.
- **public boolean createClassKey():**  
Will create an access key by the teacher to provide to the students.
- **public boolean editDeadline():**

The teacher will edit the deadline of the project.

- **public boolean editCalendar():**

The teacher or the instructor will edit the deadlines in the calendar, which will be visible in the students' calendar, too.

## 4. Improvement Summary

In the second iteration we have changed the subsystem decomposition diagram because the dependencies between packages were the opposite. Also, we have created four more packages which are Calendar Manager, Project Manager, Statistics Manager and User Manager. We have also added the versions of the apps that the application will work with. Moreover, we have changed the class diagram. We have added the missing multiplicities and also missing relations between classes. We have added IClass as an interface so that the Group class won't reach what Class can do specially. We have created new associations with the new classes as well because we thought more thoroughly this time and found out that the classes Request and Invitation were missing. Submission class was added as well because we thought that assignments and each group's submission should be different from each other. We also have added new operations and attributes to the classes which we forgot in the first place. Lastly, we have introduced the design patterns which will be helpful in the implementation part of the project and we decided to use Mediator and Singleton design patterns.

## 5. Glossary & References

- [1] Brügge, B., & Dutoit, A. H. (2004). *Object oriented software engineering using UML, patterns, and Java* (Third ed.). Upper Saddle River, NJ: Pearson Education.
- [2] Creating websocket applications in the Java EE PLATFORM. (n.d.). Retrieved from <https://javaee.github.io/tutorial/websocket002.html#BABEAEFC>
- [3] Java technologies for web applications. (n.d.). Retrieved from <https://www.oracle.com/technical-resources/articles/java/webapps.html>
- [4] Javax. Servlet (java ee 5 sdk). (n.d.). Retrieved from <https://docs.oracle.com/javaee/5/api/javax/servlet/package-summary.html>
- [5] Schedule (Java EE 6 ). (2011, February 10). Retrieved from <https://docs.oracle.com/javaee/6/api/javax/ejb/Schedule.html>
- [6] Uses of package.org.springframework.core. (n.d.). Retrieved from <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/core/package-use.html>