Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Peer Review System

Design Report

group-1f

Berke Ceran, Güven Gergerli, Lara Merdol, Akmuhammet Ashyralyyev, Sıla Saraoğlu

Instructor: Eray Tüzün
Teaching Assistant(s): Elgun Jabrayilzade, Erdem Tuna

1. Introduction

1.1. Purpose of the system:

Peer review system is a helper tool for the class for grouping and reviewing projects. It is a web based application, that provides users to interact with a remote server. The application aims to ensure a convenience to all types of users in class period. Peer review systems provide users a variety of functions that vary according to user type. System allows three types of users which are instructor, student and teaching assistant (TA). Peer review system is planned to be accessible, user friendly and maintainable. Furthermore, the peer review system web application is designed responsively, so users can access the system from all of their devices (computers, smartphones and tablets) that have internet connection.

1.2. Design goals:

Identifying the design goals is important for building an application from scratch. Therefore this section presents a deeper look into the non-functional requirement that is given in the Analysis Report. The system design is client-server architecture with 3 tier architectural style [1].

1.2.2. Criteria

1.2.2.1. End User Criteria

Usability:
Peer review systems provide a user-friendly interface that can be easily followed and understood by users who will use the system. The keywords or various feature names to be used in this interface will be chosen from the words or explanations that users frequently use or encounter in daily life. It will be aimed for users to follow

them comfortably. Application contains clear cut menus and navigation bars that the user can easily understand and browse while using the application. The system design in a way that reduces complexity and offers functionalities open as possible. The system provides slightly different interfaces for different user types for the purpose that ease the experience of the user by just showing the functionalities related to them. For instance, the interface that provides forming groups within students is only visible to the students. Furthermore, the interface of starting and ending group formation processes are visible to the instructor. The system provides a FAQ page that users can look at when they have confusion and if they cannot solve their problems, they can easily contact Administers by Contact-Us page.

Reliability:
Peer review systems ensure the security of the user's private data. The system uses an online database to store various data. Private data such as various private information of users, uploads they make into the system, messaging channels, or various notifications will be stored in that online database and these private data will not be shared with anyone and will not be lost for any reason such as power loss, connection down and sign out. System design in a way that ensures users a safe and protected environment.

1.2.2.2. Maintenance Criteria

Extensibility:

Object-oriented design concepts will be used to make implementation of the Peer Review System web application so flexible and easily developed.The system can be expanded and can be easily added to new things, so when new elements or features are desired to be added, new features can be added without much trouble and major changes on the main code for future use of the system.

Maintainability:

The codes will be coded in a maintainable way for the programmers who want to add to the source code of the student evaluation system or want to understand the source codes. Code block features such as what the code blocks are or what are they associated with will be clearly stated within the comment blocks.

Reusability:

Front-end implementation of the project is suitable for using in other systems. Moreover, the application will be suitable for various browsers. The implementation is suitable to reuse with minor changes for other related programs.

Portability:

Peer Review System web application is designed responsively, therefore users can access the system from all of the devices that have an internet connection. The user-interface of the program provides the same elegance and functionality both in computers, smart-phones and tablets.

Modifiability:

Peer Review System web application uses client-server architecture design pattern which divides the program into three interconnected parts. By that pattern needed modifications can be handled without damaging all of the parts. Therefore when one of the features or interface parts needs to be changed this change only affects relative areas and not affect other parts of the system.

1.2.2.3. Performance Criteria

Performance:

Performance is the other requirement that is crucial for the system. The response time of all operations, such as publishing a post on dashboard and sending a message to a user is aimed to be minimized. Maximum response time for general operations in the

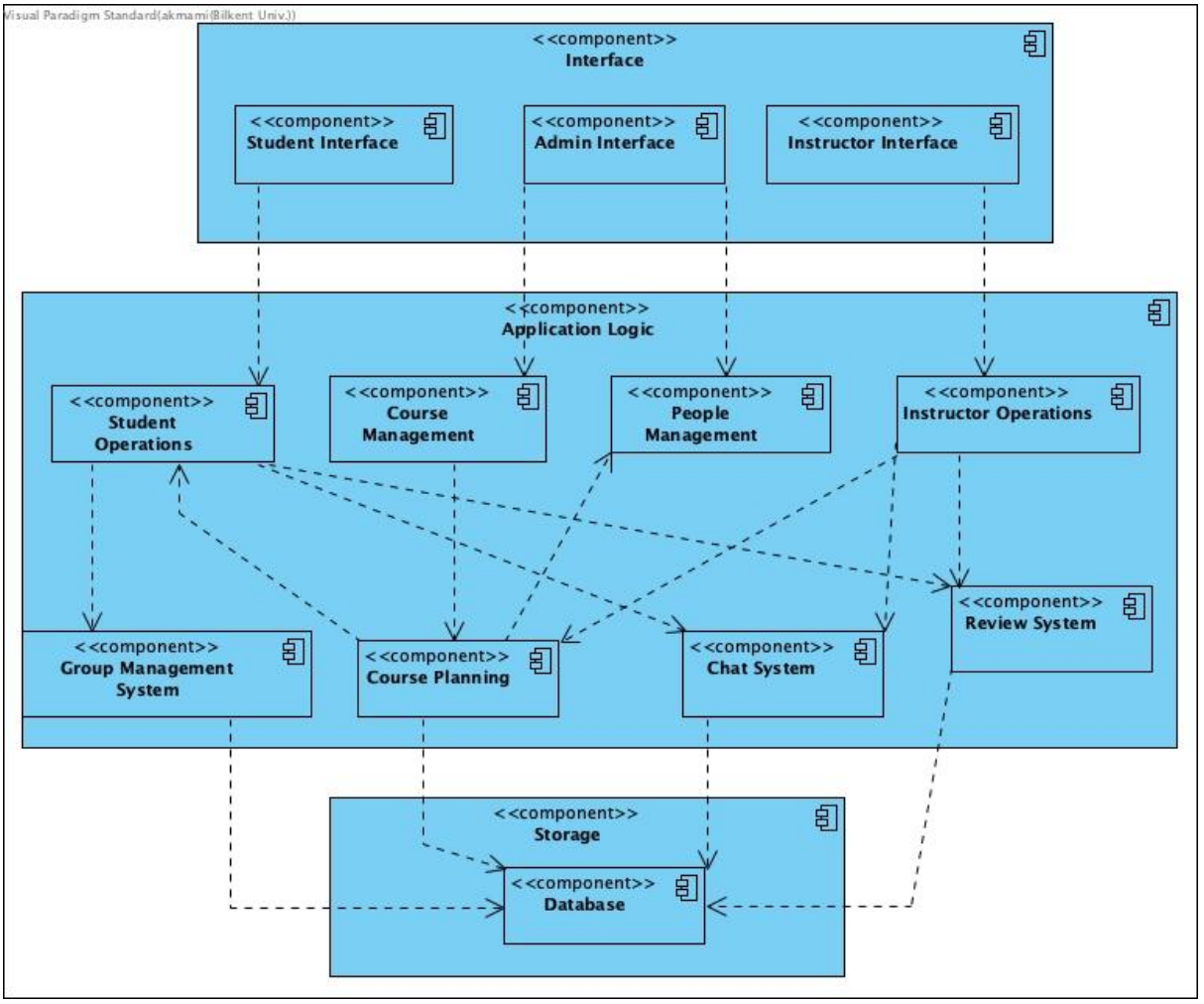
system is aimed to be 0.1 seconds. Throughput is aimed to overcome the amount of operations through the system.

1.3. Definitions

Three-tier architectural Style : Design pattern that is used in the system.
Object-oriented design: The approach that is used for solving software problems in the system.
HTML: HTML provides page structure for the user interface of the system.
CSS: CSS provides style information for formatting, control presentation, and layout.
JavaScript: JavaScript is responsible for controlling behaviours of different elements in the user interface elements in the system.
Java: Java is the programming language that is used in this project for controller and entity classes. This language has been chosen to create object-oriented design.
MongoDB: The database system which will be used in this project [2].
Singleton Design Pattern: Singleton pattern is one of the simplest design patterns in Java which is one of the design patterns that will be used in this project [3].
Template Design Pattern: One of the design patterns in Java that will be used in this project. It's logic is providing an abstract class exposing the defined way(s)/template(s) to execute its methods [4].
Strategy Design Pattern: One of the design patterns that will be used in this project to represent various strategies and a context object whose behavior varies as per its strategy object [5].

2. System Architecture

2.1. Subsystem Decomposition



In this part, the decomposition of our system will be discussed. The purpose of the decomposition into the subsystems is to design the system in a way that will make developers introduce new functionalities and classes to the project easily and divide the subsystems for different teams of the project to develop them separately. Also, by specifying subsystems, it will make much more organized and easy to understand the system.

During the decomposition, there have been used three-tier architecture which is divided into 3 main layers as it is seen in the component diagram above and these 3 layers have only access to one level below layer [1]. These three layers are:

1. **Interface:** Interface, which is responsible for all interfaces of the system, will interact with the user and system. This subsystem

takes a role of boundary and does functionalities that have been invoked by the user by clicking reserved buttons/links. Also, this system has been separated from the rest in order to track errors easily and since the technology being used in front-end changes too often (related to browsers, libraries and languages), it will be very efficient to adapt to these changes within the subsystem without changing the others. Moreover, since we are using a whole different platform (HTML/CSS/Javascript) for implementing front-end, it is appropriate that it is divided into a different subsystem[6].

2. **Application Logic:** The application logic consists of operation and management subsystems as well as other systems that our application has. The operation and management subsystem has been chosen since it is very crucial to differentiate the functionalities of different types of users. The difference between them is discussed in section 2.3. Also, it contains other subsystems that are our Within the Application Logic, coherence is increased while coupling is reduced in order to make the system more adaptable to the modifications.
3. **Storage:** Storage which will store the data as well as retrieving or querying them. This part has been decomposed into another layer in order to increase the security of the system so other subsystems will have less access to the storage. Consequently, all operations relating to the storage will be done within the subsystem when they have been invoked by Application Logic layer's components.

Three-tier Architectural Style has been chosen as a design pattern due to the following reasons:

1. The system is separated into 3 main layers, which are interface, application logic and storage, which makes it easier to track the errors.
2. Security is important in web applications, so, separation of storage subsystem is required since the operation related to the database will be done only by this subsystem.
3. Technologies where our application will run will, ex. browsers, and libraries which will be used in front-end development changes so

frequently that it would be very difficult to modify the whole project if the interface was not separated from other subsystems.

4. The communication between controller and entity objects will be high, which is coherence while the communication between layers are low, which is coupling. The ideal system should be minimized in coupling and maximized in coherence, which we have done while decomposing the system.
5. The three-tier architecture is simple and it will increase the efficiency in the implementation of the project.

In conclusion, our system is decomposed in order to be as much as developer friendly and modifiable and adjustable.

2.2. Hardware/Software Mapping

Peer Review System will be implemented in front-end HTML, CSS, JavaScript, and back-end will be implemented in Java programming language. Although we use 4 kinds of languages as the software language, all transactions will be done mostly by Java. Maven, another software tool that is necessary for the project and compatible with Java, will also be used in this project. Since Maven will be used and to avoid problems and run smoother, the Java Development Kit must be in 8 and higher versions.

The people who will use the application Peer Review System do not need very systematic and high-level tools in terms of hardware, and it is sufficient to have a proper mouse and keyboard in terms of hardware requirements. In this application, the mouse will be used to click different buttons and menu selections related to application properties. In addition to this, the keyboard will also be used for various input and output (I/O) operations within the application as a hardware requirement. In addition to the computer, since the application is opened via the browser, users can also enter the application with a smartphone with internet access. As a result, when looking at these requirements, it is seen that the users who will use the application can use this application comfortably with a minimal computer and smartphone and the tools next to it.

On the storage side of the application, the various information of the users using the application or the operations they perform within the application must be stored in a certain place. A database is required to store this various data and the MongoDB database will be used for these operations. In addition, since this application works over the internet, users must have internet connection.

2.3. Persistent Data Management

Peer Review System application requires data storage or a database for various tasks. In this application, users' profile information, their chat correspondence with other users or various files that they can upload to the application, the courses, sections, groups in the sections; students, instructors and TAs etc. will be stored in the MongoDB database. Information will be stored on BSON. BSON is a computer data exchange format that means binary JSON based on the term JSON and is used to store data on mongoDB. In this field, we have different BSON data types which are user, group, chat, artifact Review, peer Review, Project, Section, Course and these are the main names of database collections. In this way, even if there is a malfunction on the end-user side, the data will be restored again as they are stored on the local host of the server.

2.4. Access Control and Security

Since various information about users in the application and other operations will be stored in the MongoDB database, their security is also very important. While this information is stored in the local host of the server, the people who can access this storage area are only the application owners, and there will be no problems in terms of security because it is necessary to have various private keys for this access.

	Instructor	TA	Student	Admin
Create Course	No	No	No	Yes
Add Section	Yes	Yes	No	No
Assign Instructor	No	No	No	Yes

Assign TA	Yes	No	No	No
Add Student to the Course	Yes	Yes	No	No
Start Group Formation	Yes	No	No	No
Assign Project	Yes	No	No	No
Assign Student to the Group	Yes	Yes	No	No
Give Artifact Review	Yes	Yes	Yes	No
Give Project Rating	Yes	Yes	Yes	No
Assign Peer Review	Yes	Yes	No	No
Perform Peer Review	No	No	Yes	No
See Peer Reviews	Yes	Yes	No	No

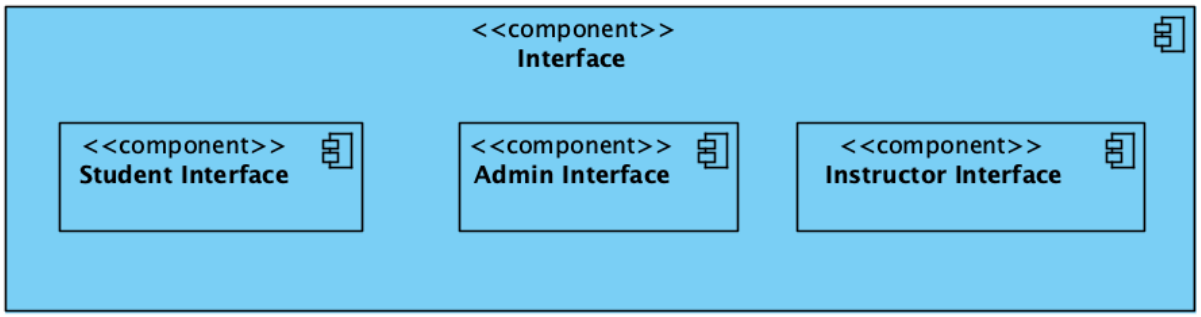
2.5. Boundary Conditions

The Peer Review System application does not require operations such as a download or copying from computer to computer, because this application runs on internet browsers, users will be able to access this application by connecting to the internet without downloading any files. This feature will increase the portability of the application and the number of users it can reach.

After users open the application on their browser, a sign-in page will appear. When the users which are the members enter the correct information here, their information will be transferred to the application through the database and this information will be presented to the user.
Users will have 2 different options to exit the application after opening the application through their own browser, the first of which is that they can exit the application by using the sign-out feature in the application, and at the same time, they can exit the application by closing the browser directly.

3. Subsystem Services

3.1. Interface Subsystem



Interface subsystem is responsible for providing UI and establishing the interaction between system and user. It contains 3 subsystem which are:

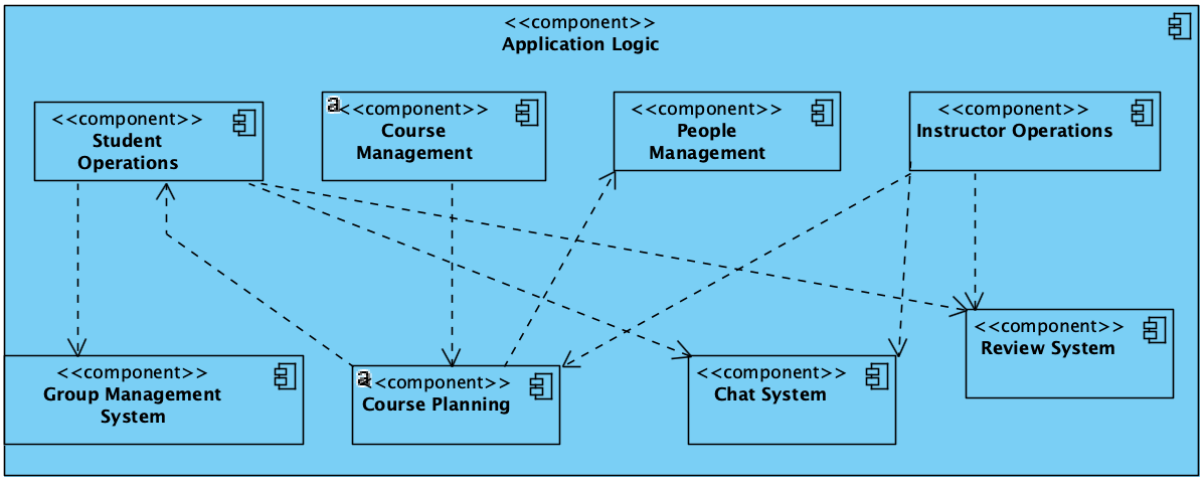
1. Student Interface
2. Instructor Interface
3. Admin Interface

Student Interface basically deals with all interfaces that are displayed to the student. The main page, profile page, section page are differ from TA and Instructor since students can not see peer review, assign students to group, strat group formations, etc. Also, all operations such as join groups, give peer review, rate the project is done by Student, hence, all these functionalities are allowed by the interface that is provided with this subsystem.

Instructor Interface differs from student in a way that is discussed earlier. Consequently, in order to provide such functionalities to the instructor, the interface is seperated. Instructor will be able to see groups and its members, see and assign students peer review, etc. So, in order to perform such functions, instructor should have additional pages, buttons, text inputs which will be dealt by Instructor Interface.

Lastly, in our Interface subsystem, there is an Admin Interface that differs entirely from other users. Admin will be able to upgrade users to Instructor and create courses, so, the interface will be designed only for allowing to perform these functions.

3.2. Application Logic



Application Logic is responsible for all operational logics in a system such as, controlling objects, checking or validating data, performing function, processing input, and entity objects. This layer can be considered as 2 different subsystems that consist of all operations that are done by users and the main systems which can be listed as:

1. Course Management
2. Student Operations
3. Instructor Operations
4. People Management
5. Group Management System
6. Course Planning
7. Chat System
8. Review System

Course Management is responsible for creating courses or deleting, which will be performed by admin. Other users will not be allowed to create courses. This subsystem will communicate with the course planning subsystem for creation of sections.

Student Operations is responsible for all operations done by students which are invoked by Student Interface. These operations are reviewing projects/artifacts/courses, creating/joining groups, etc. Hence, it will communicate with the Review system, Chat system and Group Management system. Also, the Course Planning subsystem will communicate with this subsystem for assigning the sections to students.

Instructor Operations is responsible for all operations done by instructors and TAs which are invoked by Instructor Interface. These operations are giving artifact review, assigning peer review, starting

group formations etc. However, among those operations, assigning projects to the section will be done only by the Instructor as well as starting group formation. However, the rest are the same, so, additional subsystems are not created for TA and due to the similarity of tasks, Instructors workload will be decreased (Instructor will be able to ask TAs for those tasks to do).

People Management is responsible for upgrading users to Instructor which will be invoked by admin. Also, assigning Instructors to the course, so they will be able to create sections, will be done via this management.

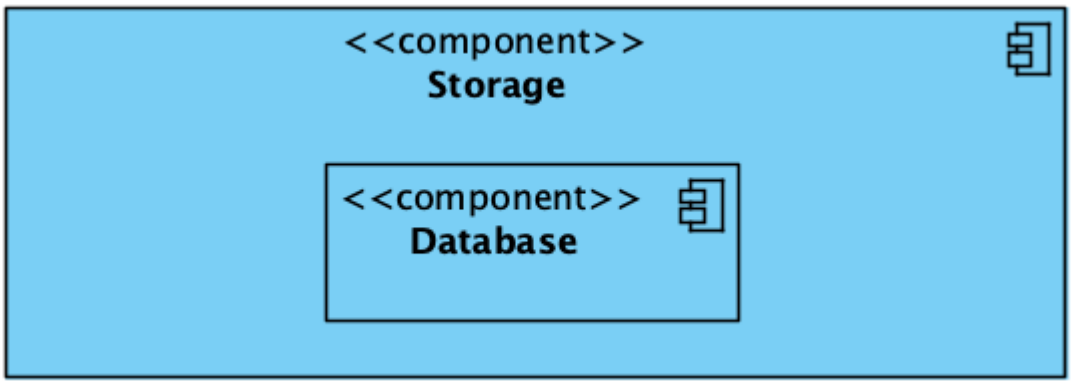
Group Management System will be responsible for all operations related to the groups such as forming, assigning artifacts etc. The important information that will be stored in Group Object should be stored in the database as well, so this subsystem will communicate with the Database subsystem.

Chat System is responsible for handling message sending and receiving. This component will communicate with the database for storing or receiving messages.

Course Planning is responsible for all operations such as creating sections and assigning TAs and students to that section. The creation of sections and assignments will be done by the Instructor, so there will be communication by them. Also, this subsystem will communicate with Student Management so the information that they are enrolled in section will be sent to the students and related operations will be done there. Also, since users should be upgraded to the TAs status by Instructor, after this assignment which will be done by Instructor, the communication from Course Planning to People Management is required as it is shown above in decomposition.

Review System is responsible for creating review questions as well as answering them (assigning answers). The creation of questions will be invoked by the instructor and answered by the student, hence, those subsystems will communicate with the Review System. For storing reviews to the database, this component will communicate with the database.

3.3. Storage



Storage subsystem is responsible for all operations related to databases. This subsystem consists of a Database component that will have all required functions that will perform searching, inserting, retrieving data in the database.This component will also establish the connection between system and database so data can be transmitted. The connection will be done whenever the system will require insertion or retrieval of data from the database [7].

4. Low Level Design

4.1. Trade-Offs:

4.1.1 Space vs Speed:

Speed of the web based application is one of the most crucial trades for it, however our program needs to store large amounts of data such as user names, uploading files, reviews. Furthermore Peer Review System web application uses a database to store user information and other crucial data. Therefore, speed is a trade off with memory in implementation.

4.1.2 Development Time vs Performance:

For the front-end part, Peer Review System web application uses HTML, CSS, and Javascript because it is easy to learn, has a rich library and simply to navigate. However, there are more high performance open sources like ReactJS. React is a more flexible library that mixes HTML with JavaScript. But react is more complex and hard to understand, thus due to having a limited time for

implementation, we choose to use HTML, CSS, and Javascript instead of more efficient libraries.

4.1.3. Functionality vs Usability:

The system prefers usability over functionality unless the system satisfies main functions. Therefore, during the period of analysis the project we focus on main tasks instead of unnecessary additional tasks. Although we prefer simplicity, the Peer Review System web application offers many functions to the user that are detailed in the Analysis Report. Majority of those functions are easy to understand by the help of the user-friendly interfaces that the application has and they are functional.

4.1.4. Memory vs Maintainability:

The system prefers maintainability over memory because the classes in the project will have many common attributes and operations which will cause unnecessary memory allocation. Since our aim is to maintain main functionalities of the application, we will try to implement interface classes or try to connect the classes with inheritance to reduce the memory usage to minimum. On the other hand, since we grouped the common operations and attributes together, it would be easy to maintain and update the system.

4.2. Design Decisions-Design Patterns

During the process of low level design we have used two design patterns: singleton design pattern, template design pattern and strategy design pattern.

4.2.1 Singleton Design Pattern

We have used a singleton design pattern in our project Semester class to provide only one instance of Semester is created and when the semester of the project is finished, there will be only one instance to destroy.

4.2.2 Template Design Pattern

We have used a template design pattern in our AssignArtifact class to provide an abstract class and the children of the specific class will

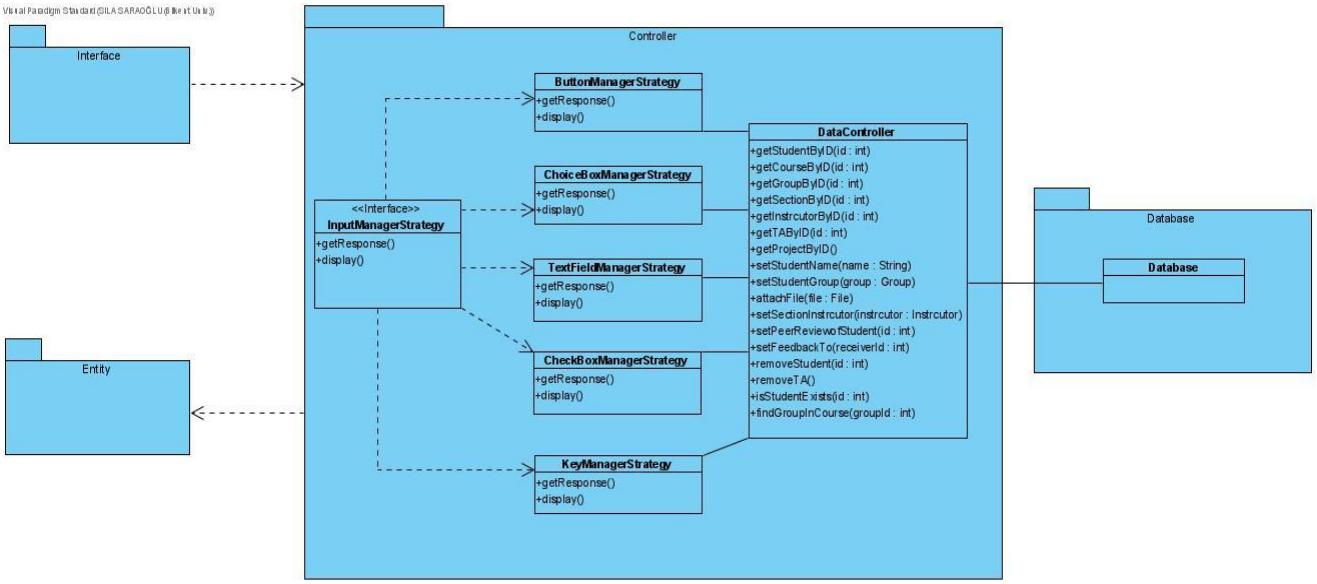
implement the method giveArtifactReview method which will have different functionality for students and TA's or instructor.

4.2.3 Strategy Design Pattern

We have used a strategy design pattern in our InputManagerStrategy class to provide different strategies for different user input in different ways.

4.3. Final Object Design

The diagram of final object design is divided into two pieces to give a more clean structure.

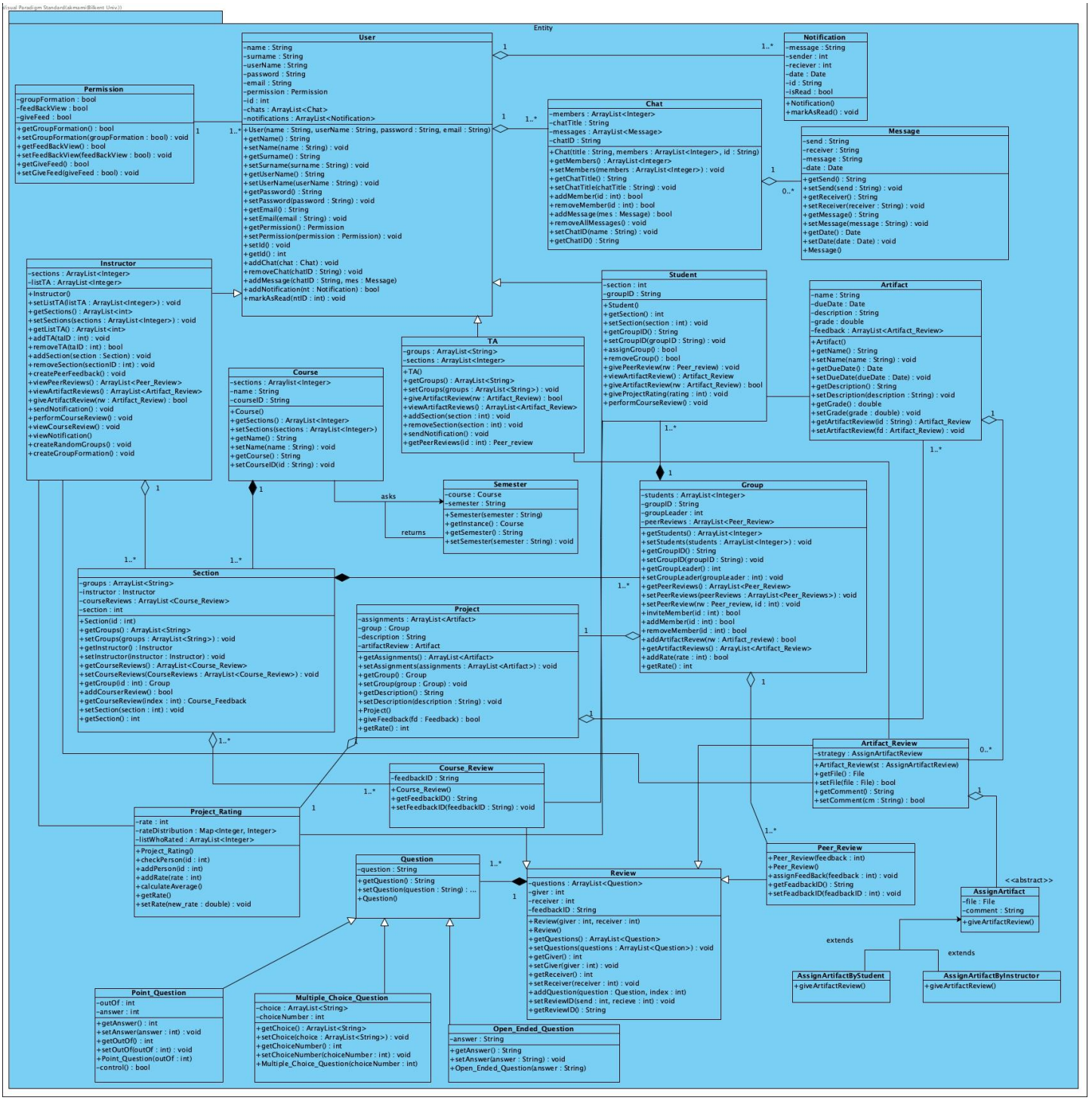


When the low level design of the project is designed we divide the classes of the same purposes and put them into the same packages. Since we are using HTML/CSS and Javascript for our interface of the project, we divide them into one package called interface. As is the nature of the HTML, CSS, and JavaScript; we did not have any classes for our user interface. The package is drawn for showing that they do interact with the controller package. As we indicate in the subsystem decomposition, we are using three tier layer architecture and each layer has only access to one layer below. Our first layer is an interface package. The interface package only interacts with our InputManagerStrategy interface class in our controller package. We think

about our controller and entity package as our second layer of our architecture called Application logic in our subsystem decomposition. This second layer has only access to the database package as the last layer of the project.

In the controller package, we tried to implement a strategy design pattern to project as implementing an interface class called InputManagerStrategy and concrete strategy classes which implements this interface. Then this manager strategy classes are associated with the DataController class which interacts with our database system. The reason why we implement a single class that interacts with our database is to meet the requirements of security purposes. The database package in the diagram represents the storage subsystem in the subsystem decomposition.

Above, we did not give details of the entity package to give a more precise picture of the interaction between packages. Following is the content of our entity package which interacts with our controller package in the application logic subsystem [9].



4.4. Packages

We will use two kinds of packages in our implementation. First packages that are defined by us which content can be seen in above section 4.1 and some external packages that will be used for various implementations.

4.4.1. Packages created by Developers

4.4.1.1. Controller Package

This package includes manager classes and controllers, this package interacts with all other packages. For instance the Input Manager which interacts with interface and DataController interact with Database are included in this package.

4.4.1.2. Entity Package

This package contains all of the entity objects that can be found in our system such as Student Class or Assignment class . This package only interacts with the controller package's classes.

4.4.1.3. Database Package

The Database package holds a database class which contains a list of students, peer reviews, assignment files, messages and all other data that must be stored. This package also interacts with just the controller.

4.4.1.4. Interface Package

The Interface package interacts with the controller. Due to the fact that we use CSS, HTML and JavaScript for the user interface of our project this package is initially empty.

4.4.2 External Packages

4.4.2.1. java.util.*

Java.util packages contain the list, date or calendar classes which will be used in the project. For example, this package will be used for creating ArrayList which will hold users of the program. This package contains a scanner class which we can use for testing our code (giving input and testing the code). Furthermore, this package contains the Date class which will be used in our program for specifying chat messages dates etc [8].

4.4.2.2 java.io

Java.io packages contain a file class which will be used in the project. This class will be used to implement the attach file feature in the project [8].

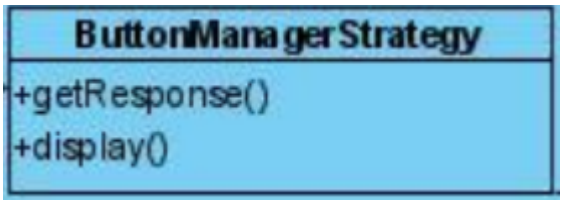
4.4.2.3 com.mongodb.*

Com.mongodb.* packages store the tools that will be an intermediary to exchange information in the database and the java program. It contains fundamental classes such as MongoClient, MongoDatabase, MongoCollection<Document> etc.

4.5. Class Interfaces

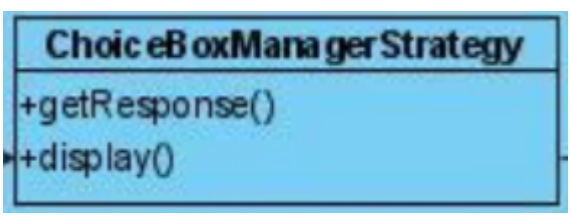
4.5.1. Controller Package Classes

4.5.1.1. ButtonManagerStrategy



Operations:
public bool getResponse ( ) : Check if there is a response on button or not.
public void display (): Display the response of the button depending on the button's function.

4.5.1.2. ChoiceBoxManagerStrategy

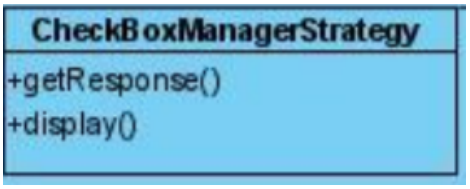


Operations:
public int getResponse ( ) : This method for getting the response choice from the question.
public void display (): Display the response of the choicebox. For example, displaying a message to the user that the choice is successfully chosen.

4.5.1.3. TextField Manager Strategy

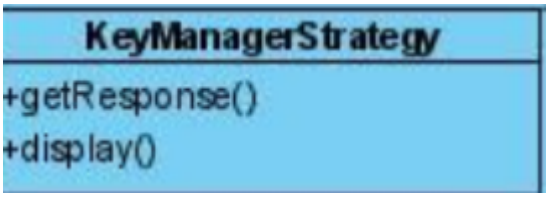
Operations:
public String getResponse ( ) : This method gets the response text in the textfield.
public void display (): Display the response text.

4.5.1.4. CheckBoxManagerStrategy



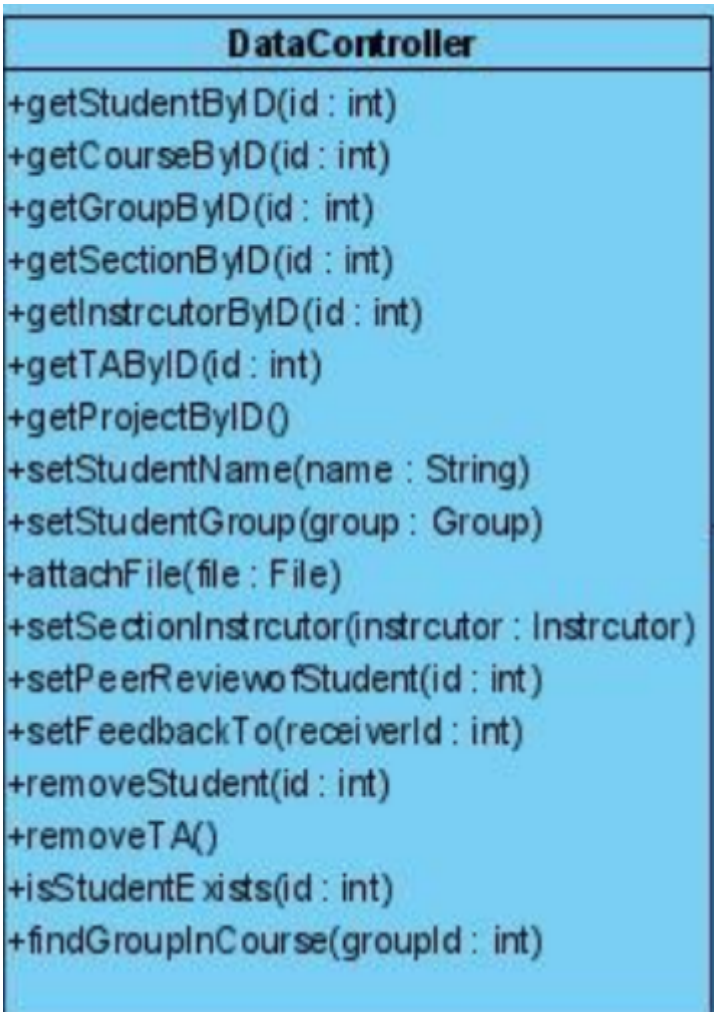
Operations:
public bool getResponse ( ) : Check if there is a response on box or not.
public void display (): Display the response.

4.5.1.5. Key Manager Strategy



Operations:
public bool getResponse ( ) : Check if there is a response on key or not.
public void display (): Display the response.

4.5.1.6. DataController



Operations:

public Student getStudentByID(int id) : Request the student data from the database by it's ID.

public Course getCourseByID(int id): Request the course data from the database by it's ID.

public Group getGroupByID(int id): Request the group data from the database by it's ID.

public Section getSectionByID(int id): Request the section data from the database by it's ID.

public Instructor getInstructorByID(int id): Request the instructor data from the database by it's ID.

public TA getTAByID(int id): Request the teaching assistant data from the database by it's ID.
public Project getProjectByID(int id): Request the project data from the database by it's ID.
public void setStudentName (Student student,String name): Set the student name in the database.
public void setStudentGroup (Group group): This method use for setting the new group into database
public void attachFile(int groupId, File file): This method is used for adding attachment files into database under specified groups.
public void setSectionInstructor(instructor instructor): This method is used for setting section's instructor.
public void setPeerReview of Student (int id): This method is used for setting peer review of specific students .
public void setFeedbackTo (int receiveID) : This method is used for setting feedback of specific students .
public void removeStudent (int id): This method for removing a student from the system. For example, if the student withdraws from the course.
public void removeTA (int id): This method for removing teaching assistants, if some issue happens.
public void isStudetExist (int id): This method for checking if a specific student is found in the system.
public Group findGroupInCourse(int groupId): This method for finding a group in the system. For example, to reach the group's project.

4.5.2 Database Package Classes

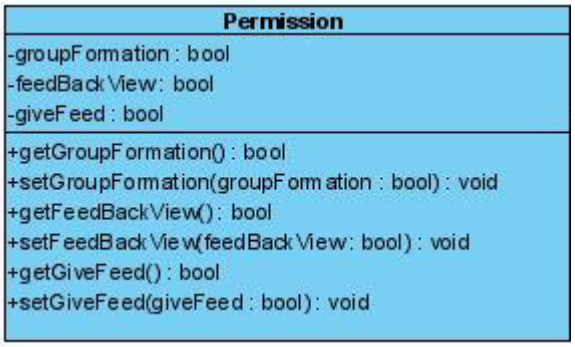
4.5.2.1. Database Class

No attribute of operation defined yet. This is the database of our system that contains all of the information about students, groups, instructors etc.

4.5.3 Entity Package Classes

4.5.3.1. Permission:

The Permission Class is used to determine which tasks 3 types of users registered in the application will be assigned (Student, Instructor, or TA).



Attributes:

boolean groupFormation: Permission for activate groupFormation period

boolean feedBackView: Permission for view feedback

boolean giveFeedback: Permission for give feedback

Operations:

set and get methods of the attributes: Methods for interact private attributes

4.5.3.2. User:

In the application, the User Class makes users register or access their own pages after registration. According to the permission class, users can be of 3 types, student, TA, or instructor.

Attributes:

String name: Name of the user.

String surname: Surname of the user.

String userName: UserName of the user.

String password: Password of the user.

String email: Email of the user.

Permission: permission: Permission object that determines the permissions of the user.

int id:ID of the user.

ArrayList<Chat> chats: Chats that user is part of.

ArrayList<Notifications> notifications: Notifications that concern the user<.

Operations:

set and get methods of the attributes: Methods for interact private attributes

User(String userName, String password, String email): Constructor of the user.

void addChat(Chat chat): Method for adding a new chat object to the chat list of the user.

removeMessage(Chat chat): Removing specific messages from the chat object of the user.

addMessage(String chatID, Message mes): Method for adding new messages to the chat object of the user.

void removeChat(String chatID): Removing specific chat from the chat list of the user.

bool addNotification(Notification nt): Adding new notification on different issues.

void markAsRead(int ntID): Marking different messages as read.

4.5.3.3 Notification:

The Notification Class enables various notifications to be created and sent. It will also store the message.

Attributes:

String message: Notification message text.

int sender: User Id of the sender user.

int receiver: User Id of the receiver user.

Date date: Date of the notification sent.

String id : Id of notification sent.

Bool isRead: Notice of notification is read or not.

Operations:

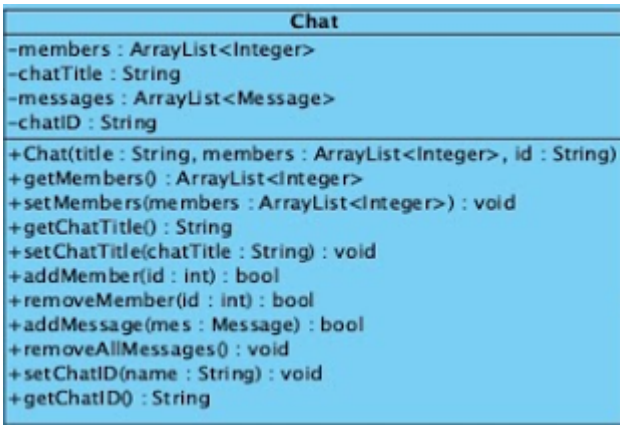
Notification(): Constructor of the Notification class.

void markAsRead(): Marking a specific notification as read.

Notification(): Constructor of the notification object.

4.5.3.4 Chat:

Chat Class allows users in the application to message with each other or to send and receive files with each other.



Attributes:

ArrayList<int> members: Users id that participate in the chat

String chatTitle: Chat title.

ArrayList<Messages> messages: User messages in the chat.

String chatID: Id of specific chat.

Operations:

Chat(String title, ArrayList<Integer> members, String Id):

Constructor of the Chat class.

set and get methods of the attributes: Methods for interact private attributes

bool addMember(int id): Method for adding new participants.

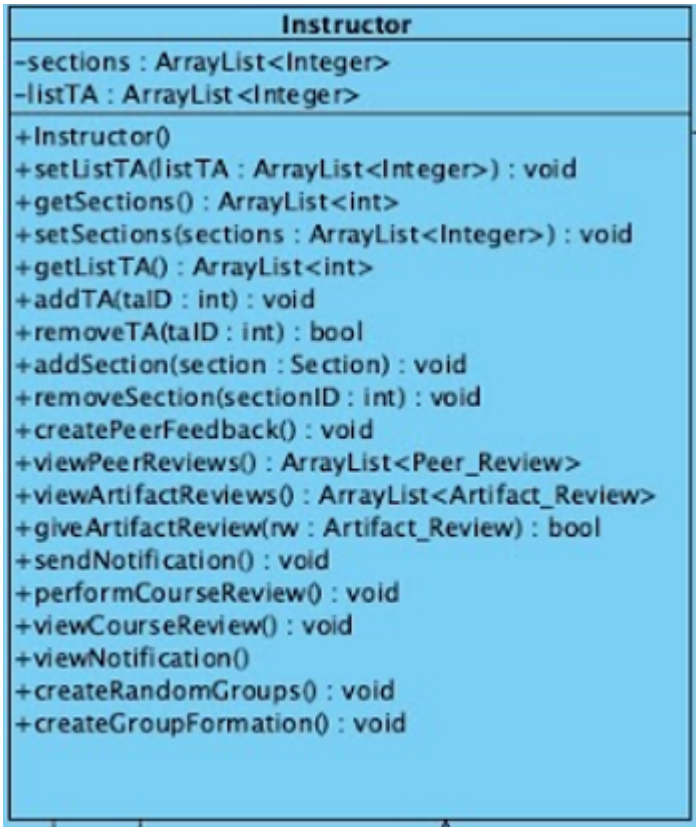
bool removeMember(int id): Method for removing participants.

bool addMessage(Message mes): Method for adding a new message to the chat.

void removeAllMessages(): Removing all messages in the chat screen.

4.5.3.5 Instructor:

The Instructor Class is specific to the instructor and allows the instructor-related work to be done.



Attributes:

ArrayList<Integer> sections: the sections of the instructor

ArrayList<Integer> listTA: the list of the TAs of the section

Operations:

set and get methods of the attributes: Methods for interact private attributes

Instructor(): the constructor of the class

void addTA(int taId): Method for assigning TA to an instructor

boolean removeTA(int taId): Method for removing a TA from an instructor

void addSection(Section section): Method for adding a section to an instructor

void removeSection(int sectionId): Method for removing a section from an instructor

void createPeerFeedback(): Method for creating a peer feedback form for the student which can be done by an instructor

ArrayList<Artifact_Review> viewPeerReviews(): Method for viewing a peer reviews

ArrayList<Feedback> viewArtifactReviews(): Method for viewing artifact reviews.

boolean giveArtifactReview(ARtifact_Review rw): Method for giving a project feedback

void sendNotification(): Method for sending notifications

void performCourseReview(): Method for performa course reviews

void viewCourseReview(): Method for viewing a course feedback

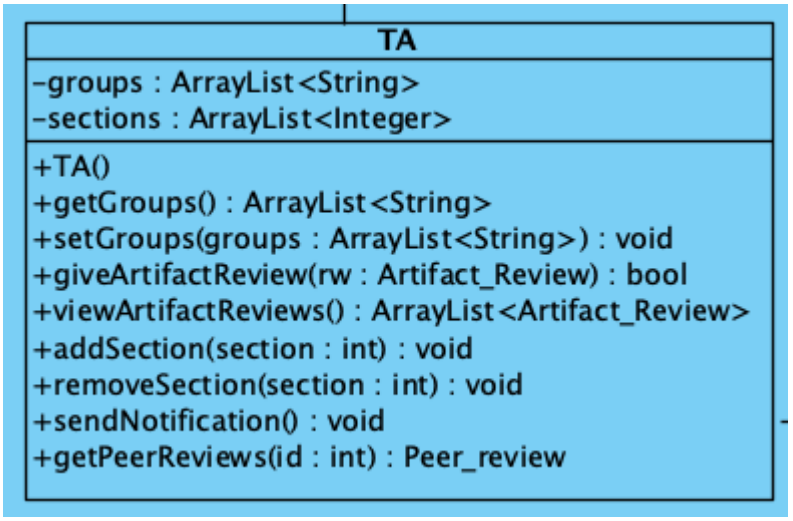
void viewNotification(): Method for viewing a notification

void createRandomGroups(): Method for creating groups with random distribution

void createGroupFormation(): Method for starting group formation process

4.5.3.6 TA:

The TA Class is specific to the instructor and allows the instructor-related work to be done.



Attributes:

ArrayList<String> groups: Group names that teaching assistant is responsible for.

ArrayList<Integer> sections: Sections names that teaching assistant is participating.

Operations:

set and get methods of the attributes: Methods for interact private attributes

bool giveArtifactReview(): Method for TA to give artifact reviews to the projects

Feedback viewArtifactReview (): Method for TA to view artifact feedback of the project.

ArrayList<Artifact_Reviews> void viewArtifactReviews(): Method for viewing a artifact feedback

void addSection(int section): Method for adding a section to TA

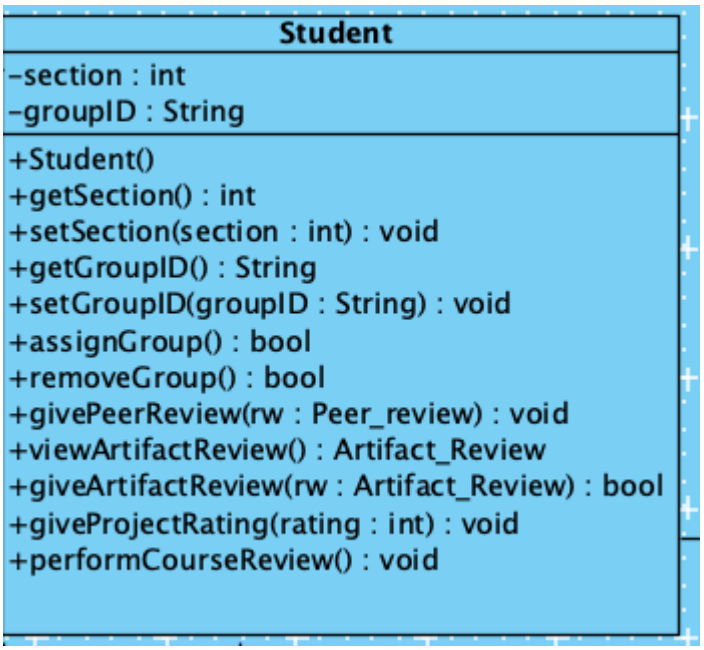
void removeSection(int section): Method for removing a section from a TA

TA(): Constructor of TA class

void sendNotifications(): Method for sending notifications

4.5.3.7 Student:

The Student Class is specific to the instructor and allows the instructor-related work to be done.



Attributes:

int section: Section id of the student

String groupID: Group id of the student

Operations:

Student(): Constructor of the Student class

set and get methods of the attributes: Methods for interact private attributes

boolean assignGroup(): Method for assigning a group to a student

boolean removeGroup(): Method for removing a group from a student

void givePeerReview(Peer_review rw): Method for giving peer reviews.

Artifact_Review viewArtifactReview(): Method for viewing artifact reviews.

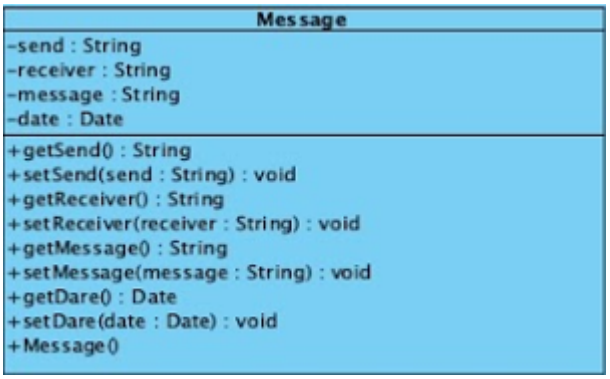
bool giveArtifactReview(Artifact_Review rw): Method for giving artifact reviews.

void giveProjectRating(int rating): Method for giving project ratings to projects.

void performCourseReview(): Method for performing course reviews.

4.5.3.8 Message:

Message Class shows the properties of the messaging channel in the Chat class.



Attributes:

String send: The name of the sender of the message.

String receiver: The name of the receiver of the message.

String message: The message's text prompt.

Date date: The sending date of the message.

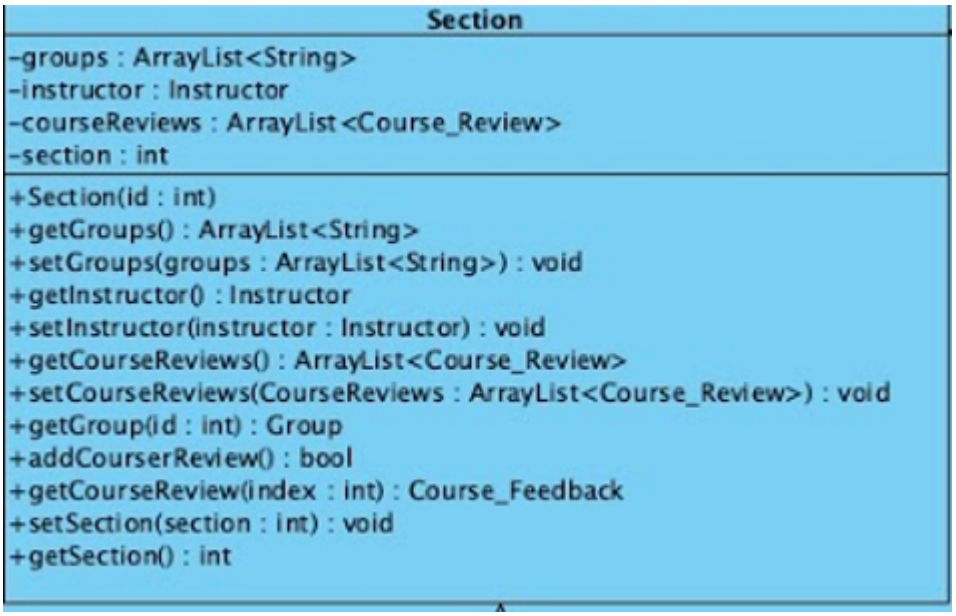
Operations:

set and get methods of the attributes: Methods for interact private attributes

Message(): Constructor of message class.

4.5.3.9 Section:

Section Class is used to determine the various sections determined by the instructor and their various features.



Attributes:

ArrayList<String> groups: The list that holds every group in the section.

Instructor instructor: The instructor of the section.

ArrayList<Course_Review> CourseReviews: The list that holds every course reviews that is filled by the students of the section.

int section: Id of the specific section.

Operations:

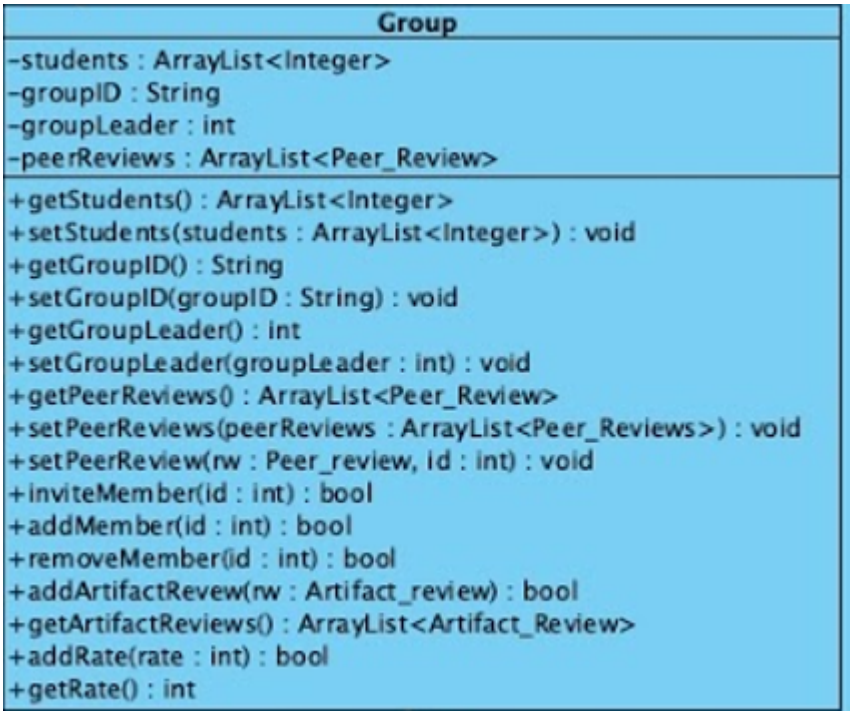
set and get methods of the attributes: Methods for interact private attributes

Section(int id): Constructor of section class.

bool addCourserReview(): The method for adding the course review to the section and returning true if successful.

4.5.3.10 Group:

Group Class allows the various students to perform tasks related to the groups they have and the characteristics of these groups.



Attributes:

ArrayList<Integer> students: The list of the students id.

String groupId: The id of the group.

int groupLeader: The student id of the groupLeader.

ArrayList<Peer_Review> peerReviews: The list of the peer reviews that are filled by the students.

Operations:

set and get methods of the attributes: Methods for interact private attributes

bool inviteMember(int id): Invites a specified student to the group

bool addMember(int id): Adds a member to the group.

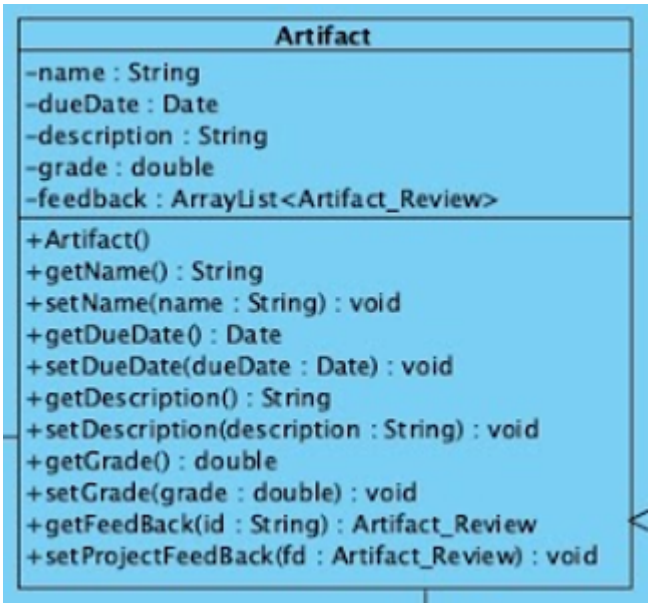
bool removeMember(int id): Removes the member from the group.

bool addArtifactReviews(Artifact_review rw): Method for adding artifact reviews.

bool addRate(int rate): Method for adding rates.

4.5.3.11 Artifact:

The Artifact Class helps the instructor to set tasks or project-related responsibilities to students or various groups.



Attributes:

String name: Name of the assignment

Date dueDate: Due date of the assignment

String description: Description of the assignment

double grade: Grade that assignment take

ArrayList<Artifact_Review> feedback : Feedback that assignment take

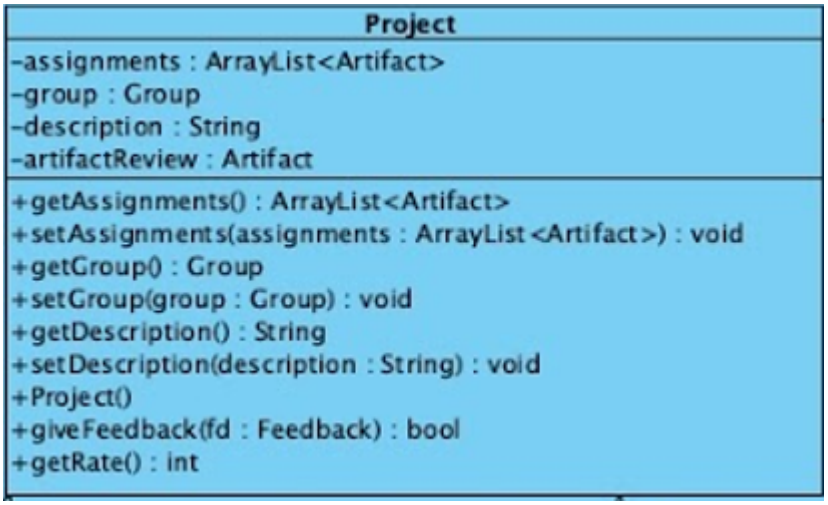
Operations:

Artifact(): Constructor of the assignment object.

set and get methods of the attributes: Methods for interact private attributes

4.5.3.12 Project:

Project class determines the properties of projects owned by various groups.



Attributes:

ArrayList<Artifact> assignments: Artifact list that one project holds.

Group group: Group that contains participants that participate in the project

Artifact artifactReview: Artifact review that project takes

String description: the description of the project

Operations:

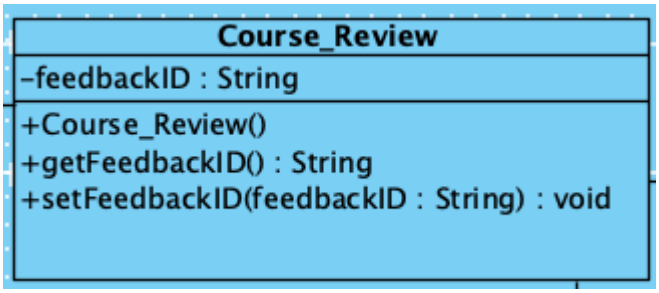
set and get methods of the attributes: Methods for interact private attributes

Project(): the constructor of the Project class

bool giveFeedback(Feedback fd): Method for giving feedback.

4.5.3.13 Course_Review:

Course_Review Class contains reviews related to the given courses from the instructor.



Attributes:

String feedbackID: the id of the feedback

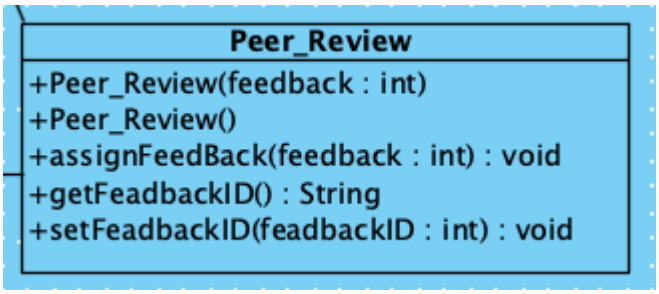
Operations:

Course_Review(): the constructor of the class

set and get methods of the attributes: Methods for interact private attributes

4.5.3.14 Peer_Review:

Peer_Review Class contains peer reviews related to between group members in a particular group.



Attributes:

String feedbackID: the id of the feedback

Operations:

Peer_Review(int feedback): the constructor of the class

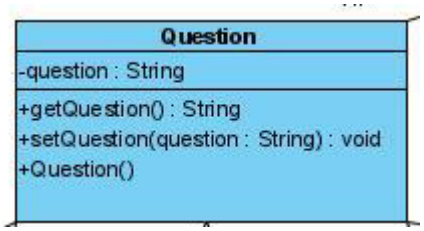
Peer_Review(): the default constructor of the class

set and get methods of the attributes: Methods for interact private attributes

void assignFeedback(int feedback): Method for assigning peer feedback

4.5.3.15 Question:

The Question Class allows setting up the questions to be asked while giving feedback. This Class is divided into 3 separate private classes.



Attributes:

String question: the string form of the question

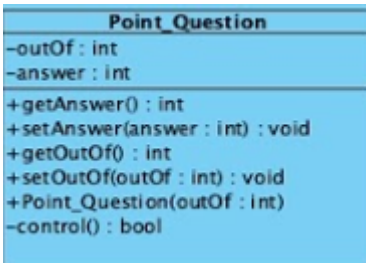
Operations:

set and get methods of the attributes: Methods for interact private attributes

Question(): Constructor of the Question class

4.5.3.16 Point_Question:

The Point_Question Class allows determining the point question among the questions to be created while giving feedback.



Attributes:

int outOf: the maximum grade of the answer for example 50 score out of 100 - 100 is the maximum grade-.

int answer: the given grade

Operations:

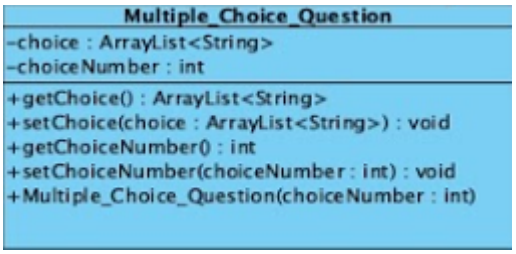
set and get methods of the attributes: Methods for interact private attributes

Point_Question(int outOf): Constructor of class.

bool control(): Controlling the point questions.

4.5.3.17 Multiple_Choice_Question:

Multiple_Choice_Question Class allows determining multiple-choice questions among the questions to be created while giving feedback.



Attributes:

ArrayList<String> choice: the choices of the question

int choiceNumber: the number of choices

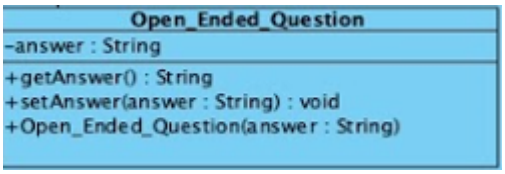
Operations:

set and get methods of the attributes: Methods for interact private attributes

voidMultiple_Choice_Question(int choiceNumber): initializing number of multiple choice questions.

4.5.3.18 Open_Ended_Question:

Open_Ended_Question Class allows determining open-ended questions among the questions to be created while giving feedback.



Attributes:

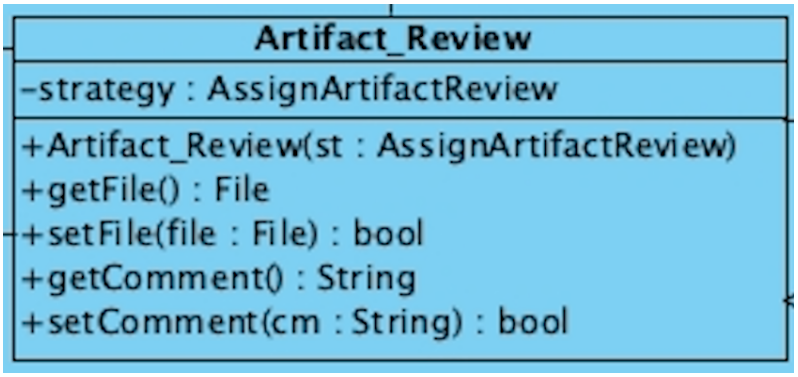
String answer: the answer of the question

Operations:

set and get methods of the attributes: Methods for interact private attributes

Open_Ended_Question(String answer): the constructor of the class

4.5.3.19 Artifact_Review:



Attributes:

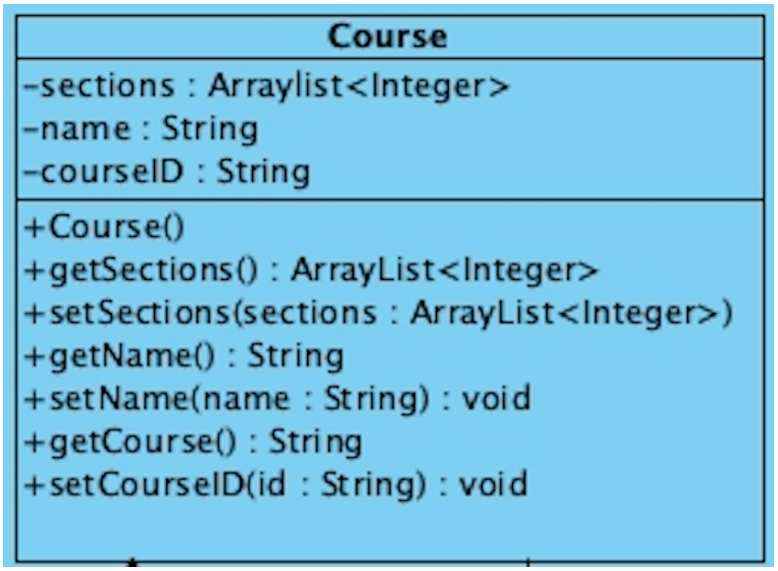
AssignArtifactReview strategy: Template pattern for giving artifact review method.

Operations:

Artifact(AssignArtifactReview st)(): Constructor of artifact_review class.

4.5.3.20 Course:

The Course class is used to determine all of the sections that belong to the course.



Attributes:

ArrayList<Integer> sections: sections of the course

String name: the name of the course

String courseId: the id of the course

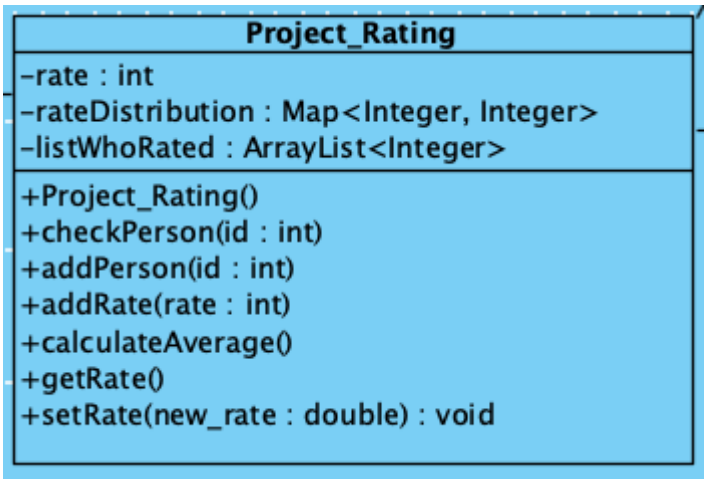
Operations:

set and get methods of the attributes: Methods for interact private attributes

Course(): The constructor of the Course class

4.5.3.21 Project_Rating:

The Project_Rating class is used to determine different projects rates.



Attributes:

int rate: Rate of project

Map<Integer,Integer> rateDistribution: Rate distributions of project.

ArrayList<Integer> listWhoRated: Id of person who was given rates.

Operations:

Project_Rating(): Constructor of Project_Rating class.

checkPerson(int id): Method that checks whether person rated project earlier or not.

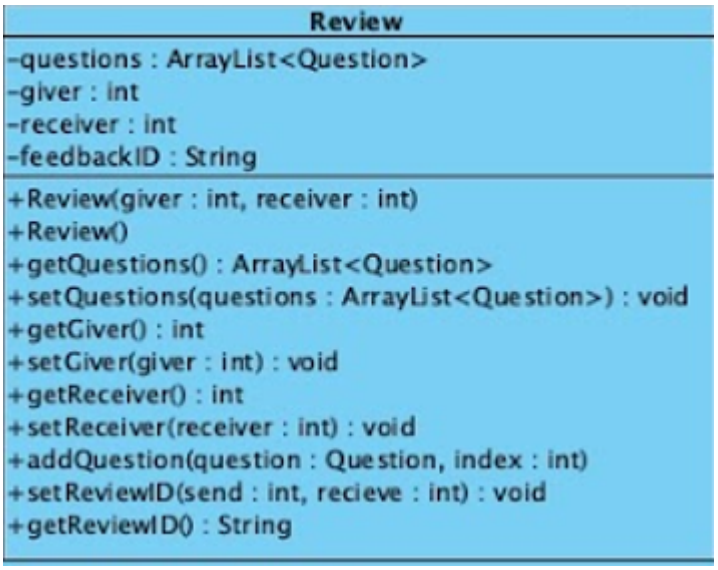
addPerson(int id): Method that adds person to rated people to the project.

addRate(): Method that adds rate to the map.

calculateAverage(): Method that calculates the average of rate from map.

4.5.3.22 Review

The Review class is used to giving reviews.



Attributes:

ArrayList<Question> questions:

int giver: Id of person who gives review.

int receiver: Id of person who got review

String feedBackID: Id of feedback.

Operations:

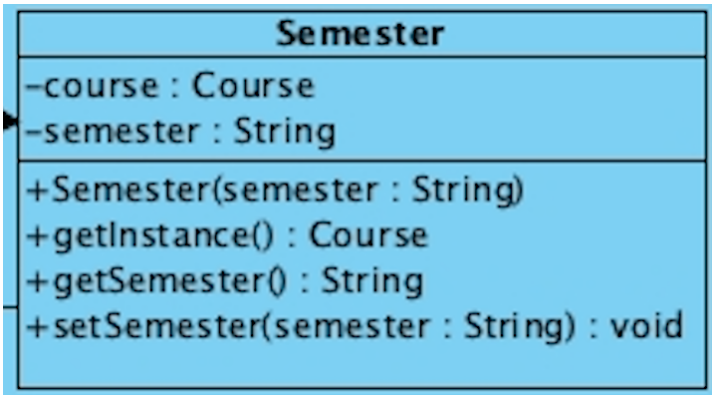
Review(int giver, int receiver): Constructor of review class.

Review(): Default constructor of review class.

void addQuestion(Question question, int index): Method for adding questions to review.

4.5.3.23 Semester

The Semester class is used to giving reviews.



Attributes:

Course course: Course will be a static object so it will be a single object in our project.

String semester: The semester in format of string

Operations:

Semester(String semester): Constructor of Semester class.

4.6. Improvement Summary

- The final object design is updated according to feedback.
- New design patterns are added to the final object design.
- The class interfaces are updated according to final object design.
- The subsystem decomposition is updated.

5. References

[1] A. Singh, “Client-Server Architecture: Types Of Client Server Architecture,” *Tech Blog*, 26-Jul-2020. [Online]. Available: https://msatechnosoft.in/blog/types-of-client-server-architecture/. [Accessed: 11-Apr-2021].

[2] The most popular database for modern apps. (n.d.). Retrieved April 11, 2021, from https://www.mongodb.com/

[3]"Design Pattern - Singleton Pattern - Tutorialspoint", *Tutorialspoint.com*, 2021. [Online]. Available: https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm. [Accessed: 25- Apr- 2021]

[4]"Design Patterns - Template Pattern - Tutorialspoint", *Tutorialspoint.com*, 2021. [Online]. Available: https://www.tutorialspoint.com/design_pattern/template_pattern.htm. [Accessed: 25- Apr- 2021]

[5]"Design Patterns - Strategy Pattern - Tutorialspoint", *Tutorialspoint.com*, 2021. [Online]. Available: https://www.tutorialspoint.com/design_pattern/strategy_pattern.htm. [Accessed: 25- Apr- 2021]

[6] Cox, L. (2020, October 26). Web design 101: How HTML, CSS, and Javascriptwork.RetrievedApril11,2021,from https://blog.hubspot.com/marketing/web-design-html-css-javascript

[7] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database system concepts*. New York, NY: McGraw-Hill.

[8] Java package - JAVATPOINT. (n.d.). Retrieved April 11, 2021, from https://www.javatpoint.com/package#:~:text=A%20java%20package%20is%20a,io%2C%20util%2C%20sql%20etc.

[9] Brügge, B., & Dutoit, A. H. (2004). *Object oriented software engineering using UML, patterns, and Java*. Upper Saddle River, NJ: Pearson Education.