Bilkent University

Department of Computer Engineering

CS 319- Object-Oriented Software Engineering

CATCH UP

Project Design Report

Group 1H

İlke Kaş, 21803184

Zeynep Büşra Ziyagil, 21802646

Bilgehan Akcan, 21802901

Yaren Yılmaz, 21803561

Ömer Onat Postacı, 21802349

Instructor: Eray Tüzün

Teaching Assistants: Erdem Tuna, Elgun Jabrayilzade

Contents

1. Introduction 4

1.1 Purpose of the system 4

1.2 Design goals 5

1.2.1 Usability 5

1.2.2 Reliability 5

1.2.3 Performance 5

1.2.4 Supportability 6

1.2.5 Cost 6

1.2.6 Maintainability 6

1.2.7 Functionality 6

2. High-level software architecture

2.1 Overview 8

2.2 Subsystem decomposition 8

2.3 Architectural Styles 8

2.3.1 Layers/Model View Controller 8

2.3.2 Client Server 9

2.4 Hardware/software mapping 10

2.5 Persistent data management 10

2.6 Access control and security 10

2.7 Boundary conditions 11

2.7.1 Termination 11

2.7.2 Failure 11

3. Subsystem Services

3.1 Design Pattern 11

3.3 Interface Management Subsystem 12

3.3 Application Management Subsystem 17

3.4 Application Object Management Subsystem 24

4. Low-level design

4.1 Object design trade-offs 48

4.1.1 Functionality vs Cost 48

4.1.2 Usability vs Security 48

4.1.3 Efficiency vs Portability 48

4.2 Final Object Design 49

4.3 Packages 50

4.4 Class Interfaces 51

4.4.1 MouseListener Interface 51

4.4.2 KeyboardListener Interface 51

5. Conclusion 52

6. References 53

Design Report

CatchUp: Classroom Helper

1 Introduction

1.1 Purpose of the system

CatchUp is a classroom helper web-based application that ensures a sophisticated interaction between the instructors, teaching assistants, and students of a course. It ensures students to keep track of their project artifacts, and assignments, while it also ensures instructors and teaching assistants to keep track of the work of the project artifacts of different student groups. The application also provides a channel-based messaging platform to ensure the communication between students, group members, and teaching assistants.

When the instructor creates a course, students and teaching assistants enroll in the course. Hereby, the instructor of the course can operate the course by assigning artifacts, sharing lecture contents, creating polls, and communicating with each student or teaching assistant; whereas students can participate in the course by uploading their work, reviewing their peers, and also works of other groups.

In this project, the purpose is to implement a classroom helper application that ensures a more advanced interaction between students, teaching assistants, and instructors to sustain a course with a term project with artifacts.

1.2 Design goals

1.2.1 Usability

In CatchUp, there are various functions- some of them are unique to user types -of different user types such as students, teaching assistants, and instructors. Due to the fact that the necessary criteria to be a usable application consist of user satisfaction, effectiveness, and efficiency, the events in Catchup are executed with minimum and optimum time for being more usable. Besides, even though the application has lots of functions for 3 user types, the interface of the application will be easy and user-friendly to satisfy the user. Thereby, the architecture and navigation of the site will be understood with minimum effort by the user.

1.2.2 Reliability

Executing the functions without any failure is one of the most essential points of the reliability of our software since the software consists of various calculations and data operations. As mentioned, there are various functions for different user types and each user type has special functions. Therefore, data operations and executing functions without any failure is important.

1.2.3 Performance

The essential points of the performance of the software are response time, latency, and server connection. The time to process a request should be low to have a good performance. The tasks and functions will be executed without any inefficiency

to avoid the latency. In addition, the data connection and the requests from the users to the Amazon server will be less than or equal to 1 seconds.

1.2.4 Supportability

CatchUp will be an extendable system that has the ability of modification or contribution to the program- as other extendable systems -without disrupting the structure of the system. With the feedback from users, appropriate modifications and additions can be applied to the system.

1.2.5 Cost

Due to the fact that the time is limited to develop our software, there is no complex design. Therefore, we do not need any tool to cost us, whilst the database server to incorporate an excessive amount of data may cost us.

1.2.6 Maintainability

Maintainable software can be repaired, improved, or modified. Therefore, our software can be easily modified, repaired, and improved and these alterations should not cause an error in any part of our software. The source of the problem can be found easily and repaired without any problem.

1.2.7 Functionality

Classroom helper applications consist of various features to increase the interaction between the students, Tas, and the instructors. However, our design has

extra features compared to other classroom helper applications to attract the attention of the user and increase the efficiency in the courses with projects.

2 High-level software architecture

2.1 Overview

In this part of the report, we divide our system to the subsystems which makes this design more reliable, understandable and writable for our team members. In our project we will use the Model View Controller (MVC) design architecture because we think that it suits best for our classroom helper application.

2.2 Subsystem decomposition

The system can be decomposed into three subsystems which are Interface Management, Application Management, Application Object Management. Interface Management Subsystem organizes the classes that constitute the user interface by acting together. Application Management Subsystem controls the application according to the user's actions. Application Object Management Subsystem constructs and controls the interactions between the objects of the system.

2.3 Architectural Styles

2.3.1 Layers/Model View Controller

In the decomposition process of the system we have used a three layer architectural style which is a common way to decompose websites. We have the interface on top and then application and its object management below. Application layer is the main controller which can interact with the other two both. We have used model use controller as our design architecture. By that we became able to control

what users can change and not change better and isolated our design in an improved way.

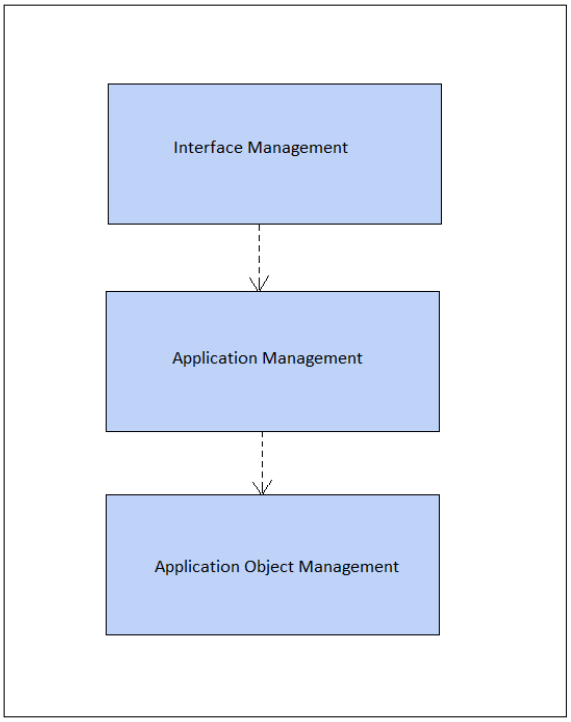


Figure 1: Layers

2.3.2 Client Server

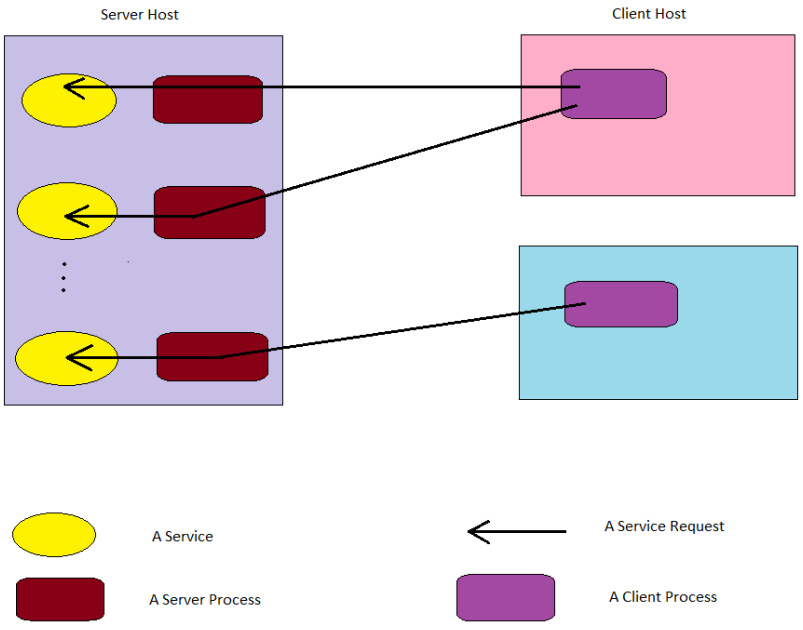


Figure 2

We have implemented the client-server model which provides an sensible approach to network systems. Server waits, listens and processes requests from clients and clients accept responses and issue requests.This diagram shows this two sided process clearly.

2.4 Hardware/software mapping

A server is needed to host the web application. Also, database will be required to store the users' and system's data. In order to make use of the database, MongoDB will be used. For the front-end part of the project, React.js which is a Javascript library will be used. Finally, for the back-end part of the project, Java8 will be used and it will be the main tool that controls the entire system. We will use VisualStudio as IDE. Catch-up requires basşc mouse and keyboard.

2.5 Persistent data management

Our design has various data types including persistent data such as email, name, surname, and id. We will store all of our data in our database system which is MongoDB. However, even though some of our data, such as password, can be altered, persistent data will not be changed and will be persistent.

2.6 Access control and security

Users have to verify their information to sign up to the system, and also they have to authenticate themselves to sign in to the application.

2.7 Boundary conditions

2.7.1 Termination

Users can log out the application by pressing the log out button that appears when the menu icon is pressed. Also, if there is no operation done in the application for a while, the system automatically logs out the user.

2.7.2 Failure

Catch Up requires internet connection to access data or to make change in the application. In the case of internet connection loss, the user will not be able to access the application. Also, the changes done in the application during the connection loss will not be saved and all the changes that are not saved before the connection loss will be lost.

3 Subsystem Services

3.1 Design Pattern

We have not decided which design pattern we will use yet. We will decide it during the implementation stage or after learning the design patterns.

3.2 Interface Management Subsystem

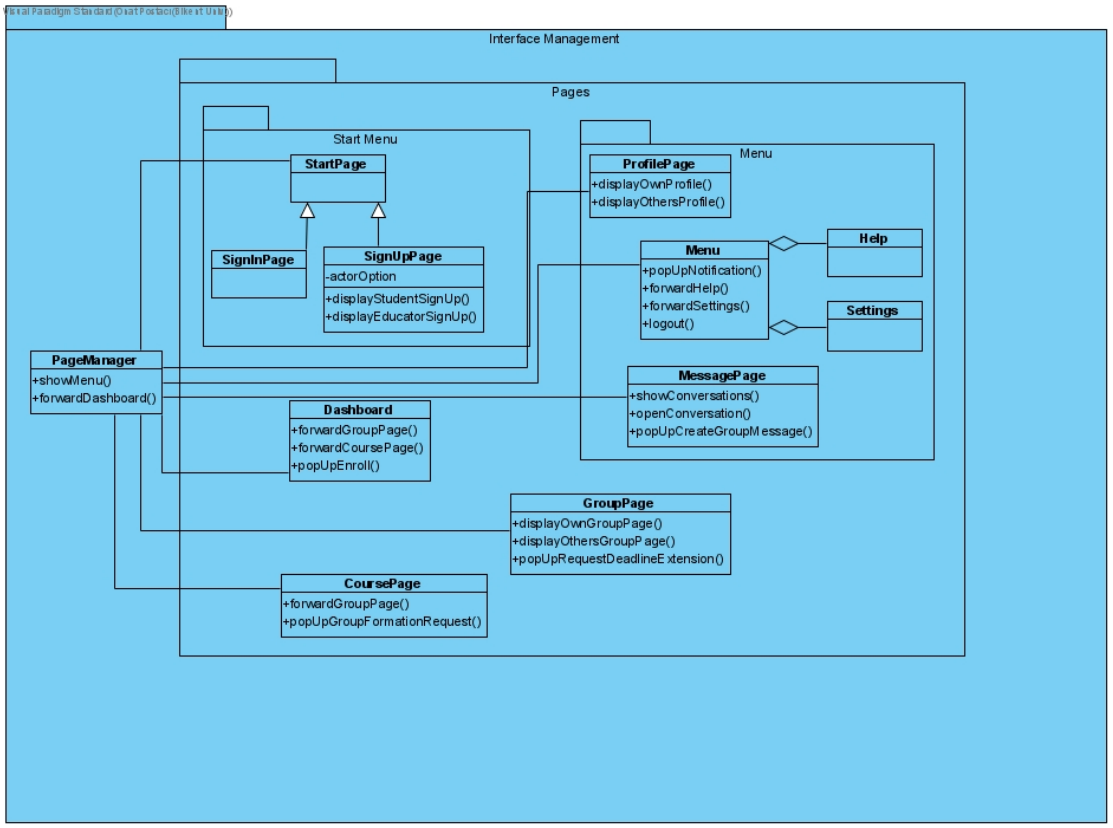


Figure 3

PageManager Class

This class manages all of the pages in the user interface and controls which page to go from any page . It enables users to have menu options and return to the dashboard option on every page.

Operations

- public void showMenu(): Shows all of the menu items.
- public void forwardDashboard(): Forwards the user to the dashboard page.

StartPage Class

This class represents the first page when a user encounters when s/he enters the system, this is the parent version of sign in and sign up class and they inherit the start page class.

SignInPage Class

This class represents the sign in page of user interface which is shown when the user enters the system.

SignUpPage Class

This class represents the sign up page of the user interface which is shown when the user first enters the system if the user does not have an account already.

Attributes

- private String actorOption: Actor role of the user in the system. Sign up information will be entered according to this choice.

Operations

- public void displayStudentSignUp(): Displays the necessary sign up information for students.
- public void displayEducatorSignUp():Displays the necessary sign up information for educators.

Dashboard Class

This class represents the dashboard page of user interface which is shown after the user signed in or signed up to the system. Also, users can select the menu, existing

course page or group page from the dashboard page. Users can enroll or create- according to user type- a new course from the dashboard page.

Operations

- public void forwardGroupPage(): Forwards the user to the group page.
- public void forwardCoursePage(): Forwards the user to the course page.
- public void popUpEnroll(): Pops up enroll course form.

CoursePage Class

This class represents the course page of the user interface which is shown after the user selects this page from the dashboard page. Also, users can select the menu or one of the groups' pages from the course page. Course page includes course information and other course related features.

Operations

- public void forwardGroupPage(): Forwards the user to group pages.
- public void popUpGroupFormationRequest(): Enables students to request for a group and form it.

GroupPage Class

This class represents the group page of the user interface which is shown after the user selects this page from the dashboard page or course page. Also, users can select the menu from the course page. This class includes group related documents and features.

Operations

- public void displayOwnGroupPage(): Displays the group page which user is a member of.
- public void displayOthersGroupPage(): Displays the group page which the user is not a member of.
- public void popUpRequestDeadlineExtention(): Processes students' deadline requests.

ProfilePage Class

This class represents the profile pages of the users when they select the go profile icon from the menu they are forwarded to their own profile. Otherwise, when they select and click on another profile they are also forwarded to this page but the profile they have chosen to go is shown.

Operations

- public void displayOwnProfile(): Displays the profile of the user.
- public void displayOthersProfile(): Displays the profile of the other users.

MessagePage Class

This class represents the message page of the user interface which is shown as one of the icons in the menu above the page. This page includes the conversation of users.

Operations

- public void showConversations(): Displays the conversations one has in groups or as pairs.
- public void openConversation(): Displays the messages of conversations.
- public void popUpCreateGroupMessage(): Enables users to form groups.

Menu Class

This class represents the menu icon above the pages. When the user clicks the menu there is a list of options which are settings, help and logout.

Operations

- public void logout(): Displays the option for user to logout.
- public void forwardHelp(): Forwards user to the help page.
- public void forwardSettings(): Forwards user to the settings page.
- public void popUpNotification(): Displays the notifications of the user from the bell icon as a pop-up window.

Help Class

This class represents the help page when users will use for necessary system information and they will be forwarded there through the menu.

Settings Class

This class represents the settings page when users need to modify their system choices and they will be forwarded there through the menu.

3.3 Application Management Subsystem

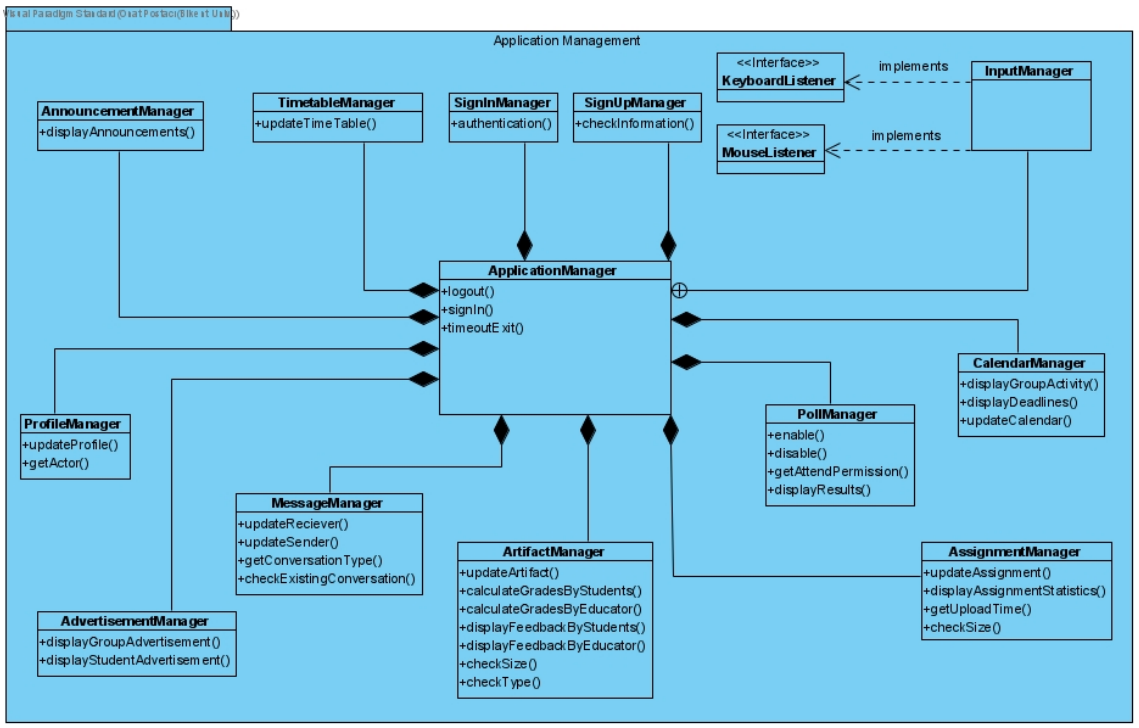


Figure 4

AnnouncementManager

This class manages -displays them at the course page- all of the announcements of the course shared by the instructor.

Operations

- public void displayAnnouncement(): Displays the announcement shared by the instructor at the course page.

TimetableManager

This class manages the timetable of the instructor.

Operations

- public void updateTimeTable(): Proceed changes in timetable.

SignInManager

This class controls the sign-in page by authenticating the users. If the user cannot authenticate himself/herself, he/she cannot sign in.

Operations

- public boolean authenticate(): Authenticates the users by getting their emails and passwords.

SignUpManager

This class controls the sign-up page of our design. It checks the necessary information entered by the user.

Operations

- public boolean checkInformation(): Checks the information entered by the user to sign up. Checks whether the information exists in our database system.

CalendarManager

This class manages the calendar object. It demonstrates the group activities, such as meetings, at the calendar, while it also controls the view of the calendar in terms of the deadlines of the upcoming assignment and artifacts.

Operations

- public void displayGroupActivity(): Displays the group activities such as meetings, presentation on the calendar.

- public void displayDeadlines(): Displays the deadlines of the upcoming assignments, or artifacts.

PollManager

This class manages the polls that are created by the instructor. It controls the polls by enabling, disabling them. It also controls the attending permission of the users. A user cannot attend a poll two times. Also, this class shows the results of the polls when they are terminated.

Operations

- public void enable(): Enables the poll so that it is visible, and users can attend.
- public void disable(): Disables the poll so that it is invisible and users cannot attend.
- public boolean getAttendPermission(): Gets the permission of users to attend the poll. Returns true if the attend count of a user is 1 or less, false if not.
- public void displayResults(): Displays the results of the poll when it is disabled.

AssignmentManager

This class manages the assignment object. As seen in the class-object diagram, assignment has a size. This class checks the size of an assignment when an assignment is tried to upload. It keeps track of the uploading time of an assignment, and updates the assignment and its information if necessary. Lastly it displays the assignment statistics such as the number of students upload the assignment on time.

Operations

- public void updateAssignment(): Updates the assignment file, the assignment information, or both of them.
- public void displayAssignmentStatistics(): Displays the statistics of an assignment such as how many students uploaded the assignment on time and how many students did not upload the assignment on time.
- public int getUploadTime(): Gets the time when the assignment is uploaded and returns the difference between the deadline and upload time in terms of seconds.
- public boolean checkSize(): Checks the size of the assignment. Returns true if the size is less than or equal to the boundary size, false if not.

ArtifactManager

This class controls the operations on the artifact object. Artifact is anything done by students. For example, an artifact can be a report written by the group members. Therefore, an artifact has a type and file size.

Operations

- public void updateArtifact(): Updates the content of the artifact or the artifact file.
- public double calculateGradesByStudents(): Calculates the average grade given by the students.
- public double calculateGradesByEducator(): Calculates the average grade given by educator.
- public void displayFeedbackByStudents(): Displays the feedback given to an artifact by the other students.

- public void displayFeedbackByEducator(): Displays the feedback given to an artifact by the educator and teaching assistants.
- public boolean checkSize(): Checks the size of the uploaded artifact file and returns true or false according to the validity.
- public boolean checkType(): Checks the type of the uploaded artifact file and controls whether the file type conforms to the expected file type, and returns true or false according to the validity.

MessageManager

This class controls messages. It renews the received messages and the sended messages.

Operations

- public void updateReceive(): Updates the messages that a user received.
- public void updateSender(): Updates the messages that a user sends.
- public String getConversationType(): Gets the conversation type. Returns “personal” or “group”.
- public void checkExistingConversation():Checks for message group.

AdvertisementManager

This class controls the advertisements. It displays an advertisement differently according to the type of the advertisement. If the advertisement is group advertisement the class displays it in a different form from the student advertisement.

Operations

- public void displayStudentAdvertisement(): Displays an advertisement by a student to join a group.
- public void displayGroupAdvertisement(): Displays an advertisement by a group to find a student.

ProfileManager

This class manages the profiles of the users. However, it controls the profiles according to the type of the actors consisting of instructor, student, and teaching assistant. As actors edit their profiles, their profiles are updated; therefore, this class updates their profile.

Operations

- public void updateProfile(): Updates the profiles of users according to their type.
- public String getActorType(): Returns the actor types assigned to the actors student, teaching assistant and instructor.

InputManager

This class looks after the inputs that come from the mouse and keyboards by using the interfaces of “MouseListener” and “KeyboardListener”.

ApplicationManager

This class is associated with all the other classes seen in the above figure of Application Management Subsystem with Has-A relationships. Basically, it contains all the actions that can be taken .

Operation

- public logout(): Logs out the user from the system.

- public signIn(): Signs in the user to the system.
- public void timeoutExit(): Logs out the user automatically in case of any time-out

3.4 Application Object Management

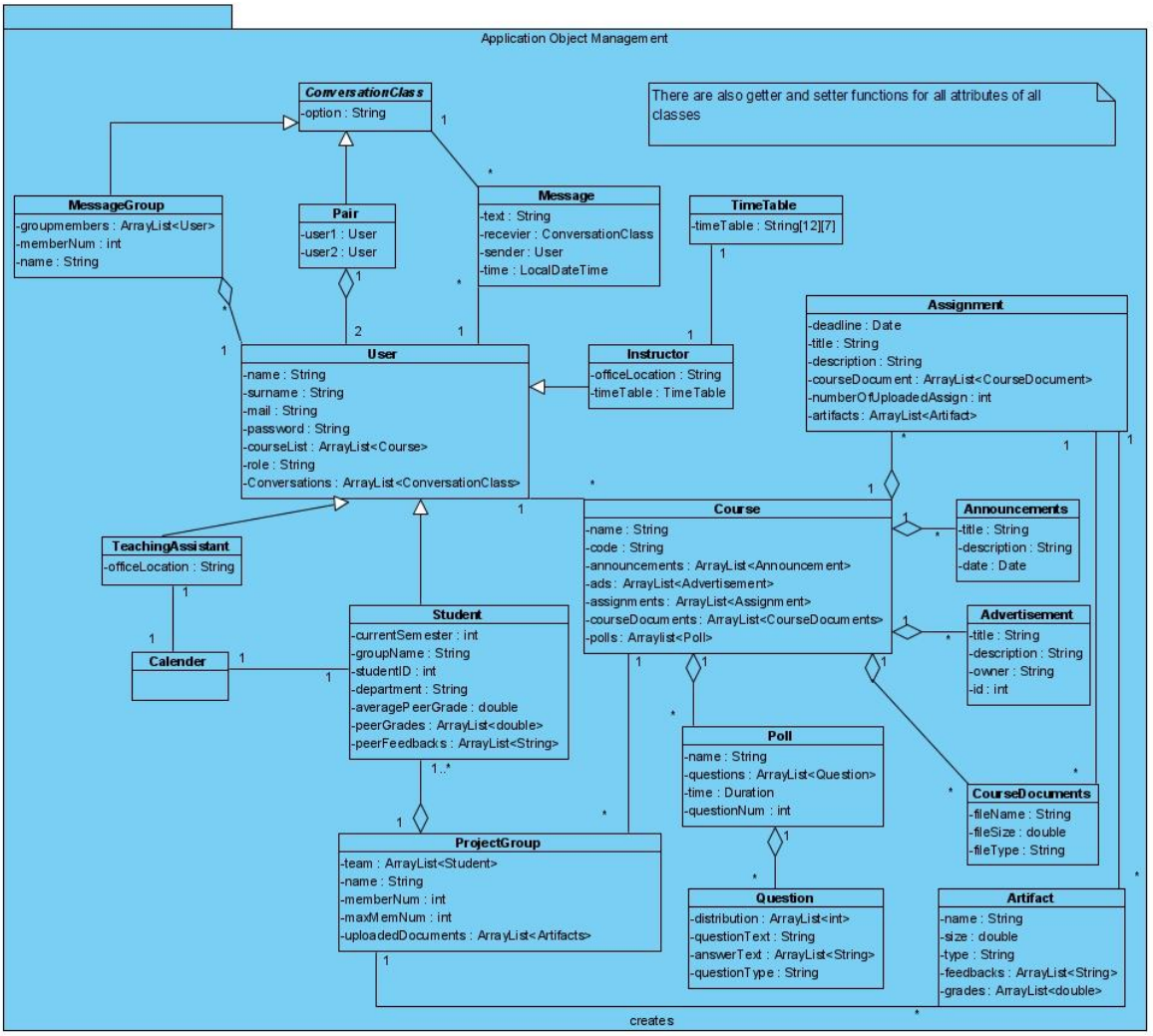


Figure 5

In this diagram, we did not write any functions for the readability of the diagram. We will show them in the next sections.

User Class

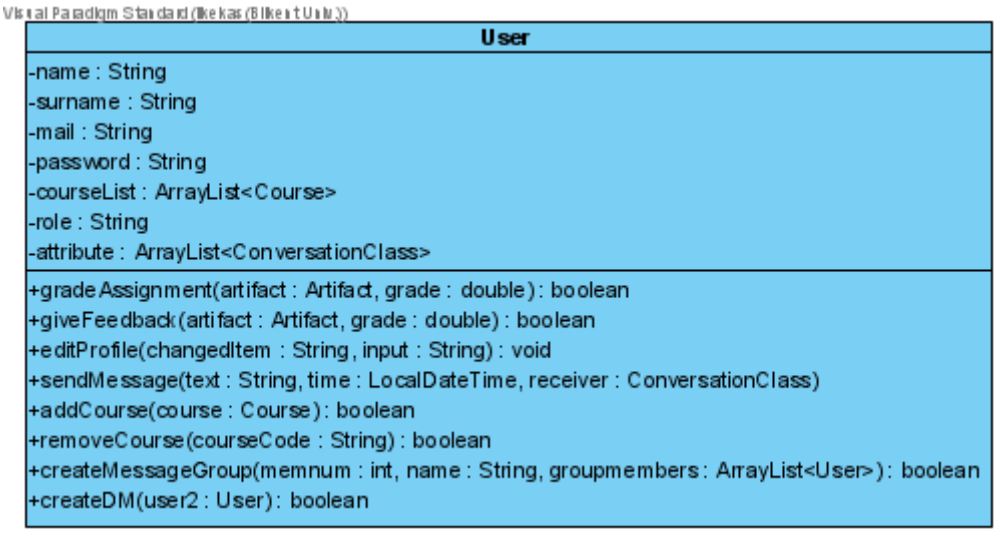


Figure 6

This class represents and defines the features and actions of all the user options in the Catch Up. All users in the system have name, surname, mail address, password and a list of courses.

Attributes

- private String name: Name of the user
- private String surname: Surname of the user
- private String mail: Mail address of the user
- private String password: Password of the user
- private ArrayList<Course> courseList: Course list of the user
- private String role: Role of the user in system

Operations

- public String getName(): Gets the user name

- public void setName(String name): Sets the user name
- public String getSurname(): Gets the user surname
- public void setSurname(String surname): Sets the user surname
- public String getMail(): Gets the user mail address
- public void setMail(String mail): Sets the user mail address
- public String getPassword(): Gets the user password
- public void setPassword(String password): Sets the user password
- public ArrayList<Course> getCourseList(): Gets the user course list
- public void setCourseList(ArrayList<Course> courseList): Sets the user course list
- public String getRole(): Gets the role of the user
- public void setRole(String role): Sets the user role
- public ArrayList<ConversationClass> getConversations(); Gets the conversations of the user
- public boolean gradeAssignment(Artifact artifact, double grade): Gives the given grade to the given artifact
- public boolean giveFeedback(Artifact artifact, String feedback): Gives the given feedback to the given artifact
- public void editProfile(String changedItem, String input): Edits the selected profile part
- public boolean sendMessage(String text, LocalDateTime time, ConversationClass receiver): Creates and sends a message to a given receiver
- public boolean addCourse(Course course): Adds the given course to the user's course list
- public boolean removeCourse(String courseCode): Removes the selected course from user's course list.

- public boolean createMessageGroup(int memnum, String name, ArrayList<User> groupMembers): Creates message group
- public boolean createDM(User user1): Creates pair for direct messages

Student Class

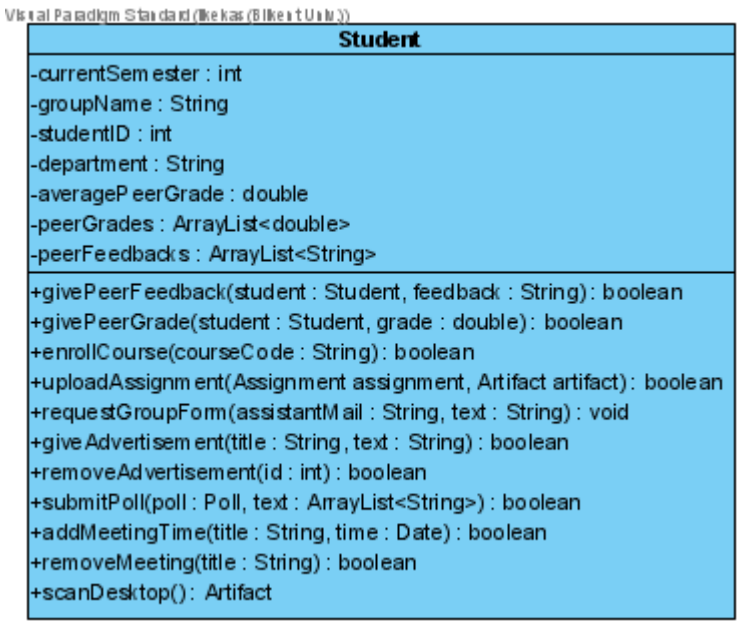


Figure 7

This class is a subclass of the user class. Besides the user features, students have the information of their current semester, group name, id, department and peer grade average in the system.

Attributes

- private int currentSemester: Current semester of the student
- private String groupName: Name of the student's group
- private int studentID: ID of the student
- private String department: Department of the student
- private double averagePeerGrade: Average of the received peer grades of the student

- private ArrayList<double> peerGrades: All the peer grades of student
- private ArrayList<String> peerFeedbacks: All the peer feedbacks of student

Operations

- public int getCurrentSemester(): Gets the current semester of student
- public void setCurrentSemester(int currentSemester): Sets the current semester of the student
- public String getGroupName(): Gets the group name of student
- public void setGroupName(String groupName): Sets the group name of student
- public int getStudentID(): Gets the student ID
- public void setStudentID(int studentID): Sets the student ID
- public String getDepartment(): Gets the student department
- public void setDepartment(String department): Sets the student department
- public double getAveragePeerGrade(): Gets the average peer grade of the student
- public void setAveragePeerGrade(double grade): Sets the average peer grade of the student
- public ArrayList<double> getPeerGrades(): Gets the all peer grades of the student
- public ArrayList<String> getPeerFeedbacks(): Gets the all peer feedbacks of the student
- public boolean givePeerGrade(Student student, double grade): Gives grade to the selected student
- public boolean givePeerFeedback(Student student, String feedback) : Gives feedback to the selected student
- public boolean enrollCourse(String courseCode): Enrolls the student to the course with given course code

- public boolean uploadAssignment(Assignment assignment,Artifact artifact): Uploads the project artifact to the assignment
- public void requestFormGroup(String assistantMail, String text): Sends a group formation request to the assistant
- public boolean giveAdvertisement(String title, String text): Gives advertise in order to find group
- public boolean removeAdvertisement(int id): Removes the given advertise
- public boolean submitPoll(Poll poll, ArrayList<String> text): Submits the answered pole
- public boolean addMeetingTime(String title, Date time): Adds a meeting to the given day
- public boolean removeMeeting(String title): Removes the added meeting
- public Artifact scanDesktop(): Scans the desktop to upload the selected artifact

ProjectGroup Class

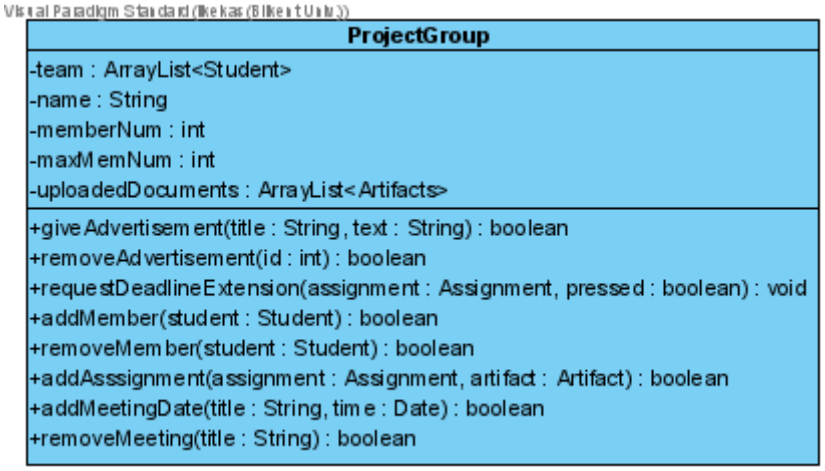


Figure 8

Project groups consist of students. Thus, this class has the list of students that are in the project group. Also, every project group has a name, number of current members and number of maximum members that can be in a group.

Attributes

- private String name: Name of the group
- private ArrayList<Student> team: Team members of group
- private int memberNum: Current member number of group
- private int maxMemNum: Maximum member number in group
- private ArrayList<Artifact> uploadedArtifacts: Uploaded documents of the group

Operations

- public String getName(): Gets the group name
- public void setName(String name): Sets the group name
- public ArrayList<Student> getTeam(): Gets the team members
- public int getMemberNum(): Gets the current group size
- public void setMemberNum(int memberNum): Sets the current group size
- public int getMaxMemNum(): Gets the maximum group size
- public void setMaxMemNum(int maxMemNum): Sets the current group size
- public ArrayList<Artifact> getUploadedDocuments(): Gets the uploaded documents of the group
- public boolean giveAdvertisement(String title, String text): Gives advertisement for group
- public boolean removeAdvertisement(int id): Removes the given advertisement
- public void requestDeadlineExtension(Assignment assignment): Requests extension deadline for assignment

- public boolean addMember(Student student): Adds the selected student to the group
- public boolean removeMember(Student student): Removes the selected student from group
- public void addAssignment(Assignment assignment,Artifact artifact): Uploads the group assignment
- public boolean addMeetingDate(String title, Date time): Adds meeting date for a group
- public boolean removeMeeting(String title): Removes the given meeting

Instructor Class

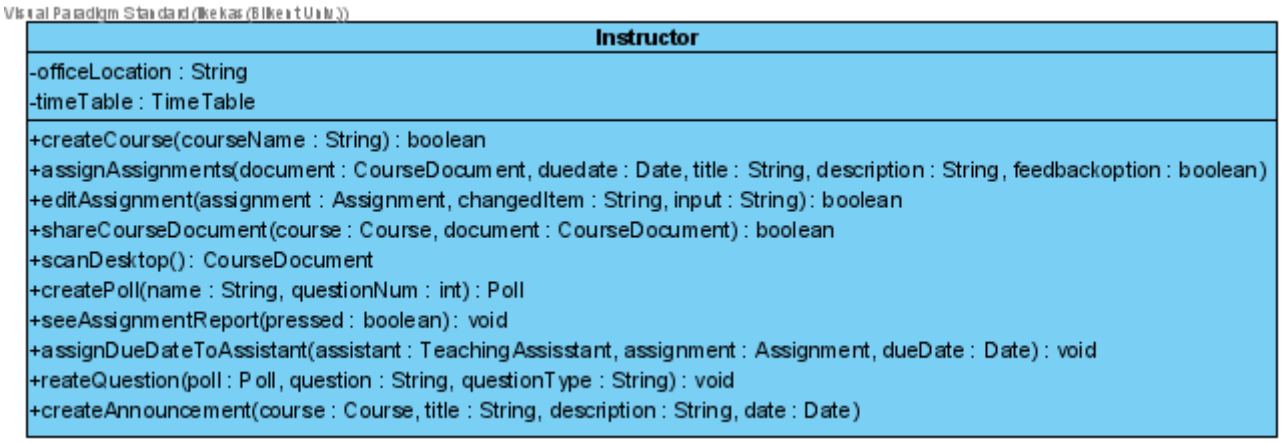


Figure 9

This class is a subclass of the user class. Besides the user features, instructors have their office location and time table information in the system.

Attributes

- private String officeLocation: Location of the office of instructor
- private TimeTable timeTable: Time table of instructor

Operations

- public String getOfficeLocation(): Gets the office location
- public void setOfficeLocation(String officeLocation): Sets the office location
- public TimeTable getTimeTable(): Gets time table of instructor
- public void setTimeTable(TimeTable timeTable): Sets time table of instructor
- public String createCourse(String courseName): Creates a new course
- public boolean assignAssignments(CourseDocuments document, Date duedate, String title, String description, boolean feedbackOption): Assigns a new assignment for course
- public boolean editAssignment(Assignment assignment, String changedItem, String input): Edits the previously created assignment
- public boolean shareCourseDocument(Course course, CourseDocuments document): Shares a course document
- public CourseDocument scanDesktop(): Scans the desktop to upload the selected course document
- public Poll createPoll(String name, int questionNum): Creates a poll
- public void seeAssignmentReport(): Shows assignment statistic report
- public void assignDueDateToAssistant(TeachingAssistant assistant, Assignment assignment, Date duedate): Assign due date to assistant
- public void createQuestion(Poll poll, String question, String questionType): Creates a question for poll
- public boolean createAnnouncement(Course course, String title, String description, Date date): Creates announcement for the course
- public void editTimeTable(String date, String time, String title, String editType): Edits the time table of instructor

TeachingAssistant Class

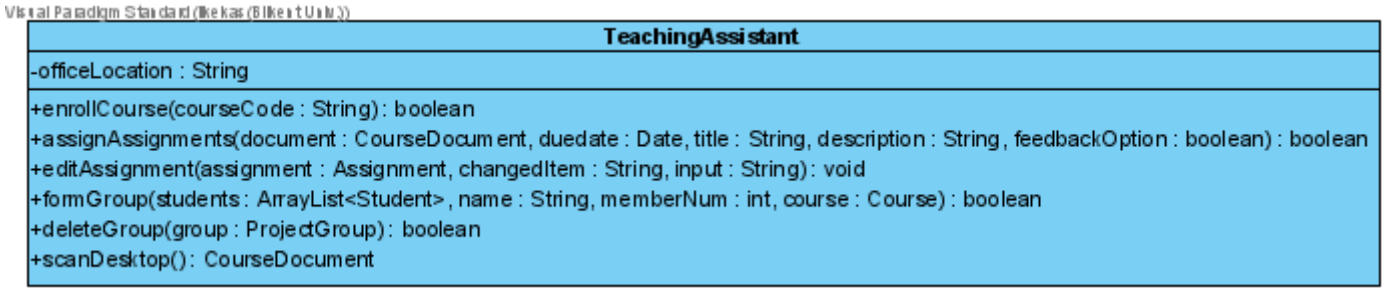


Figure 10

This class is a subclass of the user class. Besides the user features, teaching assistants have their office location information in the system.

Attributes

- private String officeLocation: Office Location of the assistant

Operations

- public String getOfficeLocation(): Gets the office location of the TA
- public void setOfficeLocation(String office):Sets the office location of the TA
- public boolean enrollCourse(String courseCode): Enrolls the course by codes
- public boolean assignAssignments(CourseDocument document, Date duedate, String title, String description, boolean feedbackOption): Assigns assignments
- public void editAssignment(Assignment assignment, String changedItem, String input): Edits assignments
- public boolean formGroup(ArrayList<Student> students , String name, int memberNum , Course course): Forms group
- public boolean deleteGroup(ProjectGroup group): Deletes the group
- public CourseDocument scanDesktop(): Scans the desktop to upload the selected course document

Course Class

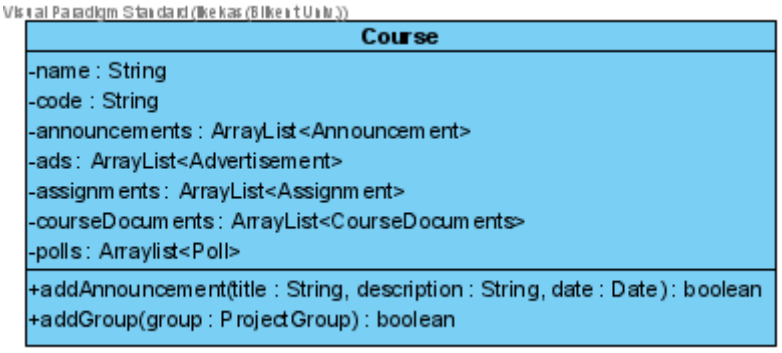


Figure 11

All users in the system can have many courses. These courses have a name and a unique code. Also, there are many assignments, ads, course documents and polls in every course.

Attributes

- private String name: Name of the course
- private String code: Code of the course
- private ArrayList<Announcement> announcements: Announcements that are done in course
- private ArrayList<Advertisement> ads: Advertisements that are given for that course
- private ArrayList<Assignment> assignments: Assignments that are assigned for that course
- private ArrayList<CourseDocuments> courseDocuments: Course Documents that are uploaded for that course
- private ArrayList<Poll> polls: Polls that are created for that course

Operations

- public String getName(): Gets the name of the course
- public void setName(String name): Sets the name of the course

- public String getCode(): Gets the code of the course
- public void setCode(String code):Sets the code of the course
- public ArrayList<Announcements> getAnnouncements(): Gets the announcements of the course
- public ArrayList<Advertisement> getAds(): Gets the advertisements of the course
- public ArrayList<Assignment> getAssignments(): Gets the assignments of the course
- public ArrayList<CourseDocuments> getCourseDocuments(): Gets the course documents of the course
- public ArrayList<Poll> getPolls(): Gets the polls of the course
- public boolean addAnnouncement(String title, String description, Date date): Add announcements to the course
- public boolean addGroup(ProjectGroup group): Add group to the course

Poll Class

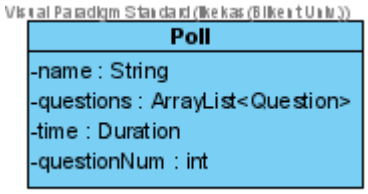


Figure 12

Polls can be created by instructors for students to reflect their opinions on course subjects. Each poll has a name, certain number of questions and a time duration to be answered.

Attributes

- private String name: Name of the poll

- private ArrayList<Question> questions: Questions in the poll
- private Duration time: Accessibility time of the poll
- private int questionNum: Number of the questions of the poll

Operations

- public Duration getTime(): Gets the accessibility time of the poll
- public void setTime(Duration time): Sets the accessibility time of the poll
- public String getName(): Gets the name of the poll
- public void setName(String name): Sets the name of the poll
- public ArrayList<Question> getQuestions(): Gets the questions in the poll
- public int getQuestionNum(): Gets the question number in the poll
- public void setQuestionNum(int num):Sets the question number in the poll

Question Class

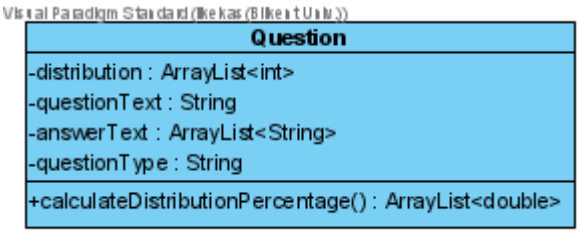


Figure 13

This class represents poll questions. Every question has a question type, question text to ask the question, list of answer text to demonstrate different answer options and list of distribution for each answer option.

Attributes

- private ArrayList<int> distribution: Distribution of the votes for each question
- private String questionText: Text of the question
- private ArrayList<String> answerText: Text of the answer

- private String questionType: Type of the question

Operations

- public String getQuestionText(): Gets the text of the question
- public void setQuestionText(String text): Sets the text of the question
- public String getAnswer(): Gets the answer of the question
- public void setAnswer(String answer): Sets the answer of the question
- public void setDistribution(ArrayList<int> distr): Sets the distribution
- public ArrayList<double> calculateDistributionPercentage(): Calculates the distribution percentage
- public String getQuestionType(): Gets the type of the question
- public void setQuestionType(String questionType): Sets the type of the question

Advertisement Class

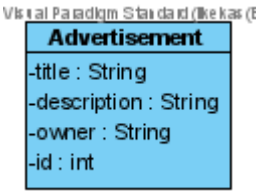


Figure 14

This class is for individual students and groups to give ads for a team or a team member to make group formation easier.

Attributes

- private String title: Title of the advertisement
- private String description: Description of the advertisement
- private String owner: Owner of the advertisement
- private int id: ID of the advertisement

Operations

- public String getTitle(): Gets the title of the advertisement
- public void setTitle(String title): Sets the title of the advertisement
- public String getDescription(): Gets the description of the advertisement
- public void setDescription(String description): Sets the description of the advertisement
- public String getOwner(): Gets the owner of the advertisement
- public void setOwner(String owner): Sets the owner of the advertisement
- public int getID(): Gets the ID of the advertisement
- public void setID(int id): Sets the ID of the advertisement

Assignment Class

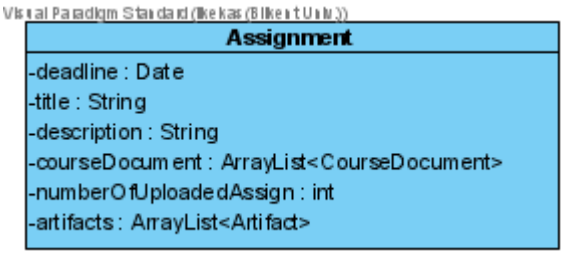


Figure 15

This class is for instructors to give project related assignments such as reports to project groups and for project groups to submit the assignment. Each assignment has a title, description, a deadline. Also, it has two different course documents list one for instructors to give assignment details with a document such as pdf and another list that includes each groups' submitted assignments. Finally, assignments have grades and feedback that provides artifact and peer review features to the system.

Attributes

- private Date deadline: Deadline of the assignment

- private String title: Title of the assignment
- private String description: Description of the assignment
- private ArrayList<CourseDocument> courseDocumentInstr: Course documents that are uploaded by instructor for the assignment
- private int numberOfUploadedAssign: Number of the uploaded course documents by the students to the assignment
- private ArrayList<CourseDocument> courseDocumentStu: Course documents that are uploaded by students for the assignment

Operations

- public Date getDeadline(): Gets the deadline of the assignment
- public void setDeadline(Date deadline): Sets the deadline of the assignment
- public String getTitle(): Gets the title of the assignment
- public void setTitle(String title): Sets the title of the assignment
- public String getDescription(): Gets the description of the assignment
- public void setDescription(String description): Sets the description of the assignment
- public ArrayList<CourseDocument> getCourseDocument(): Gets the course documents that are uploaded by the instructor
- public int getNumberOfUploadedAssign(): Gets the number of uploaded assignments by students
- public void setNumberOfUploadedAssign(int number): Sets the number of uploaded assignments by student
- public ArrayList<Artifact> getArtifacts(): Gets the course documents that are uploaded

CourseDocuments Class

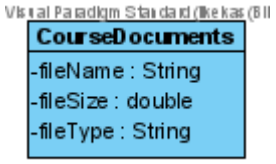


Figure 16

This class is for instructors to share project related slides or assignments in the system. Each course document has a file name, file size and a file type. It is for uploading files to the related course by all users.

Attributes

- private String fileName: Name of the course document (file)
- private double fileSize: Size of the course document (file)
- private String fileType: Type of the course document (file)
- private ArrayList<String> feedbacks: Feedbacks that are given to the assignment
- private ArrayList<double> grades: Grades that are given to the assignment

Operations

- public String getFileName(): Gets the name of the file
- public void setFileName(String fileName): Sets the name of the file
- public double getFileSize(): Gets the size of the file
- public void setFileSize(String size): Sets the size of the file
- public String getFileType(): Gets the type of the file
- public void setFileType(String type): Sets the type of the file

Announcement Class

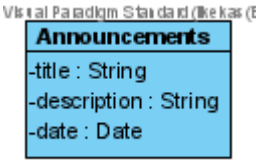


Figure 17

This class is for instructors to make announcements to all of the course's students. Each announcement has a title, description and a date.

Attributes

- private String title: Title of the announcement
- private String description: Description of the announcement
- private Date date: Date of the announcement

Operations

- public String getTitle(): Gets the title of the announcement
- public void setTitle(String title):Sets the title of the announcement
- public String getDescription(); Gets the description of the announcement
- public void setDescription(String description): Sets the description of the announcement
- public Date getDate(): Gets the date of the announcement
- public void setDate(Date announcementDate): Sets the date of the announcement

TimeTable Class

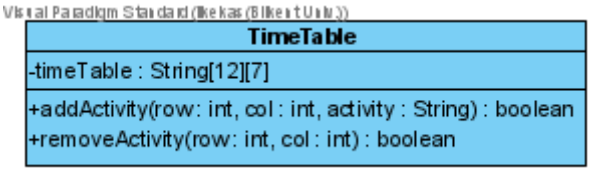


Figure 18

This class is for instructors to show their time table at the system so that students can see the empty time slots of the instructor in order to ask them for an appointment.

Attributes

- private String Timetable[12][7]: Multidimensional array that holds the timetable of the instructor

Operations

- public void setTimeTable(String[12][7] timeTable):Sets the timetable
- public String[12][7] getTimeTable(); Gets the timetable
- public boolean addActivity(int row,int col, String activity): Adds activity to the timetable
- public boolean removeActivity(int row, int col): Removes the activity from the timetable

Calendar Class

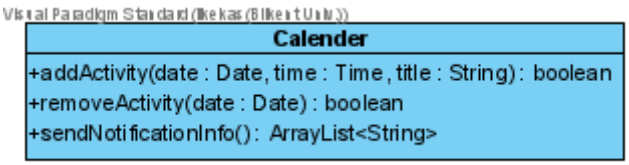


Figure 19

This class is for students and teaching assistants to see the upcoming assignments or meetings on a calendar with due dates. We will extend java.util.calender class for it.

Operations

- public boolean addActivity(Date date, Time time, String title): Adds activity to the calendar
- public boolean removeActivity(Date date): Removes activity from the calendar
- public ArrayList<String> sendNotificationInfo(): Sends the activity information to the notifications

Message Class

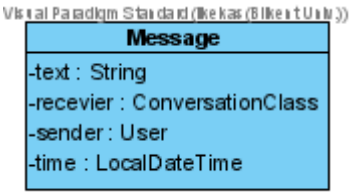


Figure 20

This class is for messaging in the system. Messages have a sender, a receiver, time and a text that includes the message information.

Attributes

- private String text: Text of the message
- private ConversationClass receiver: Receiver of the message
- private User sender: User that send the message
- private LocalDateTime time: Time of the message that has been sent

Operations

- public String getText(): Gets the message text
- public void setText(String text): Sets the message text
- public User getReceiver(): Gets the receiver
- public void setReceiver(User receiver): Sets the receiver
- public User getSender(): Gets the sender of the message
- public void setSender(User sender): Sets the sender of the message
- public LocalDateTime getTime(): Gets the time of the message
- public void setTime(LocalDateTime time): Sets the time of the message

Conversation Class

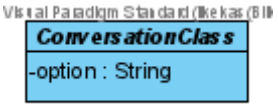


Figure 21

This class is for representing people that take roles in messaging. Each conversation class can have one or more messages.

Attributes

- private String Option: Shows whether Conversation Class is pair or message group

Operations

- public void setOption(String option): Sets the option of conversation class
- public String getOption():Gets the option of conversation class

Pair Class

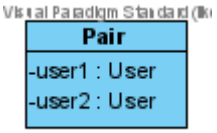


Figure 22

This class is the subclass of conversation class. It is for private messages between two users. Thus, every pair has user1 and user2.

Attributes

- private User user1: one of the user
- private User user2: another one of the user

Operations

- public User getUser1(): Gets the user1
- public void setUser1(User user1): Sets the user1
- public User getUser2(): Gets the user2
- public void setUser2(User user2): Sets the user2

MessageGroup Class

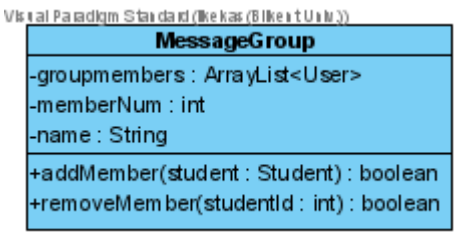


Figure 23

This class is the subclass of conversation class. It is for group messages in the system. Message groups consist of one or more users. Every message group has a name, number of members and list that includes the users in the group.

Attributes

- private ArrayList<User> groupmembers: Array list that will keep the group members
- private int memberNum: Number of the members in the group
- private String name: Name of the groups

Operations

- public ArrayList<User> getGroupMembers(): Returns the group members
- public void setGroupMembers(ArrayList<User> groupmembers): Sets the group members
- public int getMemberNum(): Returns the number of members in the group
- public void setMemberNum(int memberNum): Sets the number of members in the group
- public String getName(): Gets the name of the group
- public void setName(String name): Sets the name of the group
- public boolean addMember(Student student): Adds the member to the message group
- public boolean removeMember(int studentId): Removes the one of the member with given ID.

Artifact Class

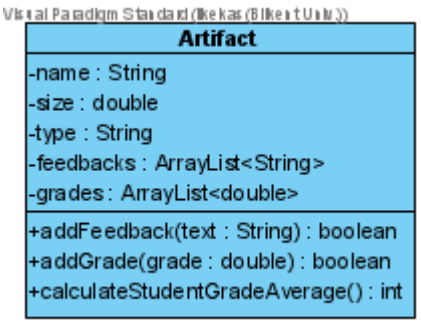


Figure 24

This class is for representing artifacts of the projects that are created and uploaded by students.

Attributes

- private String name: Name of the artifact
- private double size: Size of the artifact document
- private String type: Type of the artifact document
- private ArrayList<String> feedbacks: Feedbacks that are given by users to this artifact
- private ArrayList<double> grades: Grades that are given by users to this artifact

Operations

- public String getName(): Gets the name of the artifact
- public void setName(String artifactName): Sets the name of the artifact
- public double getSize(): Gets the size of the file
- public void setSize(String size): Sets the size of the file
- public String getType(): Gets the type of the artifact
- public void setType(String type): Sets the type of the artifact
- public ArrayList<String> getFeedbacks(): Gets the feedbacks that are given to the artifact
- public ArrayList<double> getGrades(): Gets the grades that are given to the artifact
- public boolean addFeedback(String text): Adds feedback to this artifact
- public boolean addGrade(double grade): Adds grade to this artifact

- public int calculateStudentGradeAverage(): Calculates the average of the grades that are given by students

4 Low-level Design

4.1 Object design trade-offs

4.1.1 Functionality vs Cost

Functionality is essential to execute the most basic tasks to be a classroom helper application. As we add new features to our design, our tasks increase. Our aim is to execute the tasks and increase the functionality with minimal cost. Therefore, we choose functionality over cost.

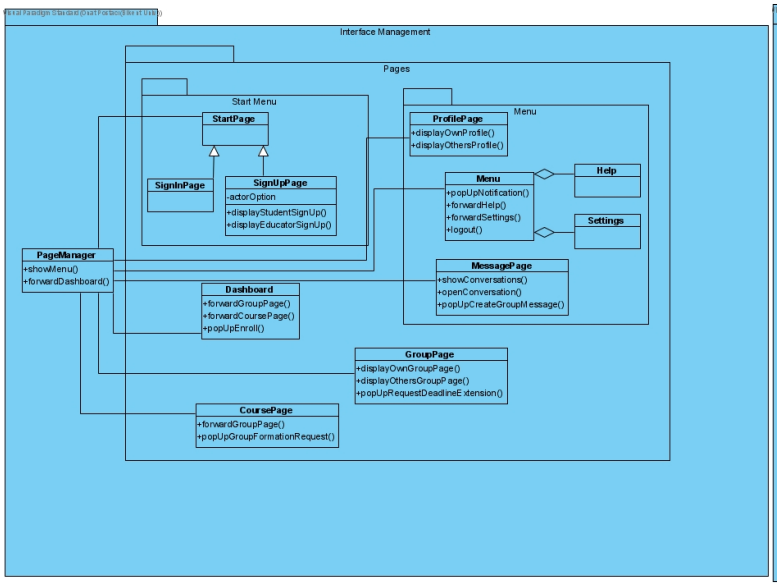
4.1.2 Usability vs Security

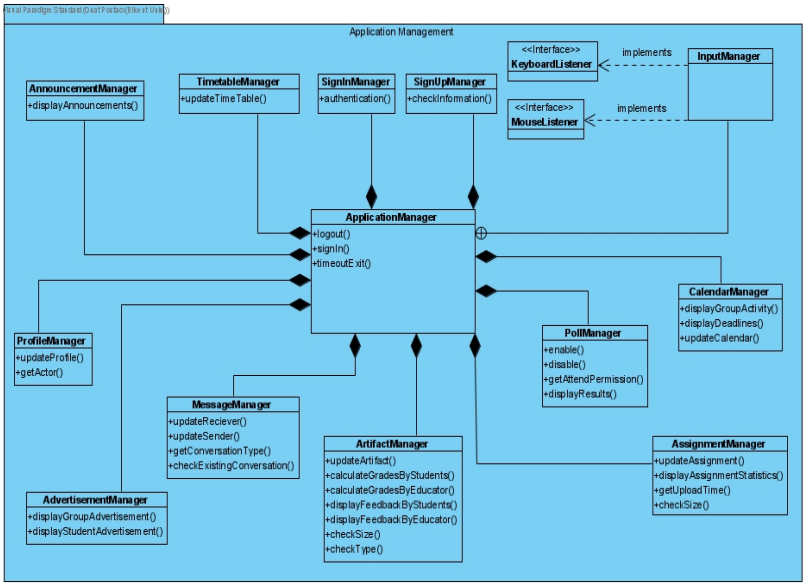
Our program will hold some private data such as the password of the user, reviews of other students' works, or grades; therefore, the security of the program is a priority for us compared to usability even though usability is significant.

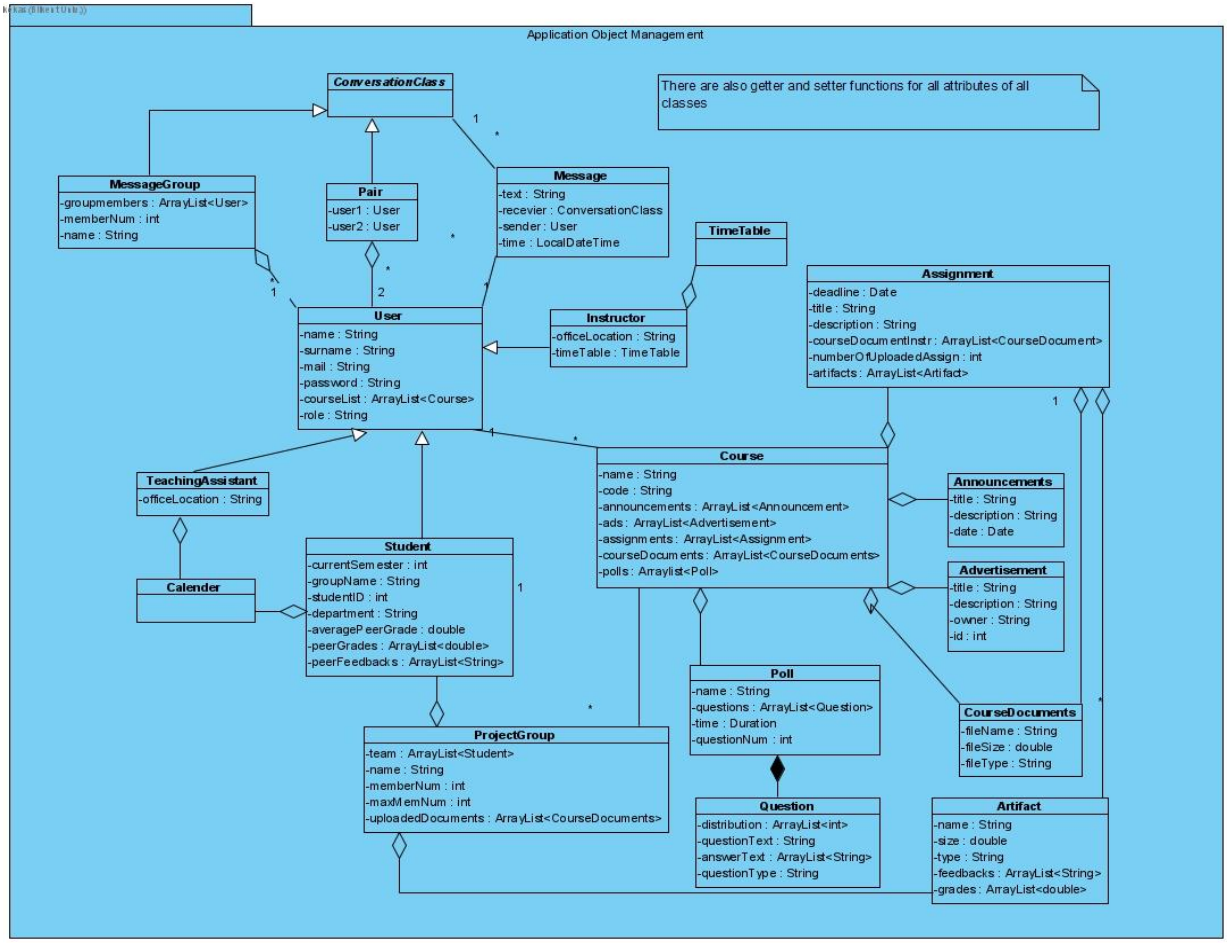
4.1.3 Efficiency vs Portability

Using the same software in different environments is essential for our design since it will be developed by group work. Therefore, even though efficiency is inevitably important, portability is indispensable.

4.2 Final object design







4.3 Packages

In Catch Up, there are three subsystems named Interface Management, Application Management and Application Object Management. They are all packages. We use subsystems to decrease complexity of the system by dividing the system into smaller parts.

Our first package is the Interface Management package. Packages can be divided into smaller parts by creating more packages. For example, inside the Interface Management package, there is another package named Pages along with a class for managing pages. Pages package contains different kinds of pages. In order to classify the pages, two more packages are used. One of them is the Menu package. This package contains menu-related pages which are the Help and Setting pages, and Profile and Message pages. Another package is the Start Menu package consisting of the pages related to the opening screen of the application such as sign in and sign up pages.

Another package is the Application Management package. It contains manager classes such as announcement, timetable, profile, advertisement, message, artifact, poll, calendar, assignment, input, sign up and sign in manager.

The last package in our system is the Application Object Management package. In this package, interactions between the classes are shown.

4.4 Class Interfaces

Interfaces define methods like all classes; however, they do not implement these methods like normal classes. Yet, classes that implement interfaces implement the methods defined by the interfaces. In our design we use the interfaces of KeyboardListener and MouseListener by implementing them in our InputManager class which exists in the package of “Application Management”.

4.4.1 KeyboardListener Interface

The KeyboardListener interface keeps track of the events happening on the keyboard and thereby by implementing it, our InputManager class keeps track of the inputs from the keyboard. When a user presses the “enter” key to execute a task, the class will help to execute the following function.

4.4.2 MouseListener Interface

The MouseListener interface keeps track of the clicks of the mouse and thereby by implementing it, our InputManager class keeps track of the inputs from the mouse actions.

5 Conclusion

For a project, it is highly essential to divide it into smaller parts in order to handle the project and share the tasks among stakeholders in an easier way. Therefore, for the design part of the project, the system is decomposed into subsystems. We have 3 subsystems for our project, namely Interface Management, Application Management and Application Object Management. Also, subsystems can

be decomposed into smaller parts by creating packages. Packages contain classes that carry out similar activities and contribute to the project in a similar way. Interface Management subsystem basically organizes the user interface classes and in that subsystem, interactions between the packages and classes are shown to make the user interface design more understandable. Application Management subsystem can be considered as the brain of the system. It controls the actions taken by the user and helps to make changes in the user interface. Lastly, Application Object Management subsystem focuses on the interactions between the objects of the system. Therefore, even though all these subsystems do different jobs, they come together and form a meaningful whole. In addition, while decomposing system and defining the boundaries of every class, package and subsystem, it is also important to pay attention to the design trade-offs such as functionality, usability or efficiency and design accordingly.

6 References

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.