CS102	**Spring 2017/18**	Project Group	7
Instructor:	**Aynur Dayanık**		
Assistant:	**Cihan Eryonucu**		

~ BilSwap ~

Add_Droppers

Yavuz Bakman, Alperen Oziş, Furkan Ahi, Hilmi Barut

Atay Kaylar

Criteria	TA/Grader	Instructor
Presentation		
Overall		

Detailed Design Report

(version 1.0)

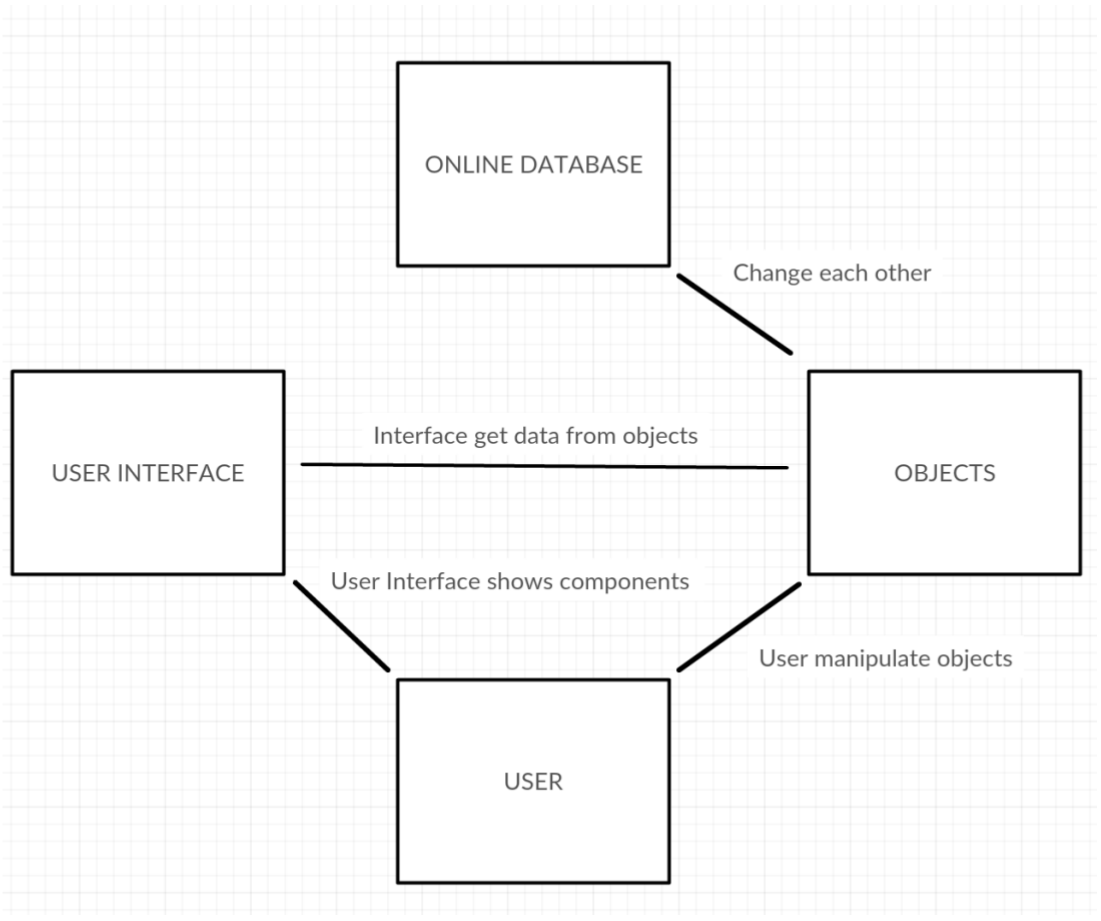
3 December 2018

1. Introduction

The program is web-based desktop program. By the program the user can exchange her/his courses with other users. The program has classes which interacts with each other in a coherent and logical way. Therefore, the program becomes reusable and changeable. This program also interacts with database simultaneously to make connection between users in the chatrooms and store profile information, requests, password, username, mails, course information. Thus, the program is designed in an appropriate way.

2. Details

2.1 System Overview



The program is based on desktop. The system will interact with the database continuously. and following tools will be used:

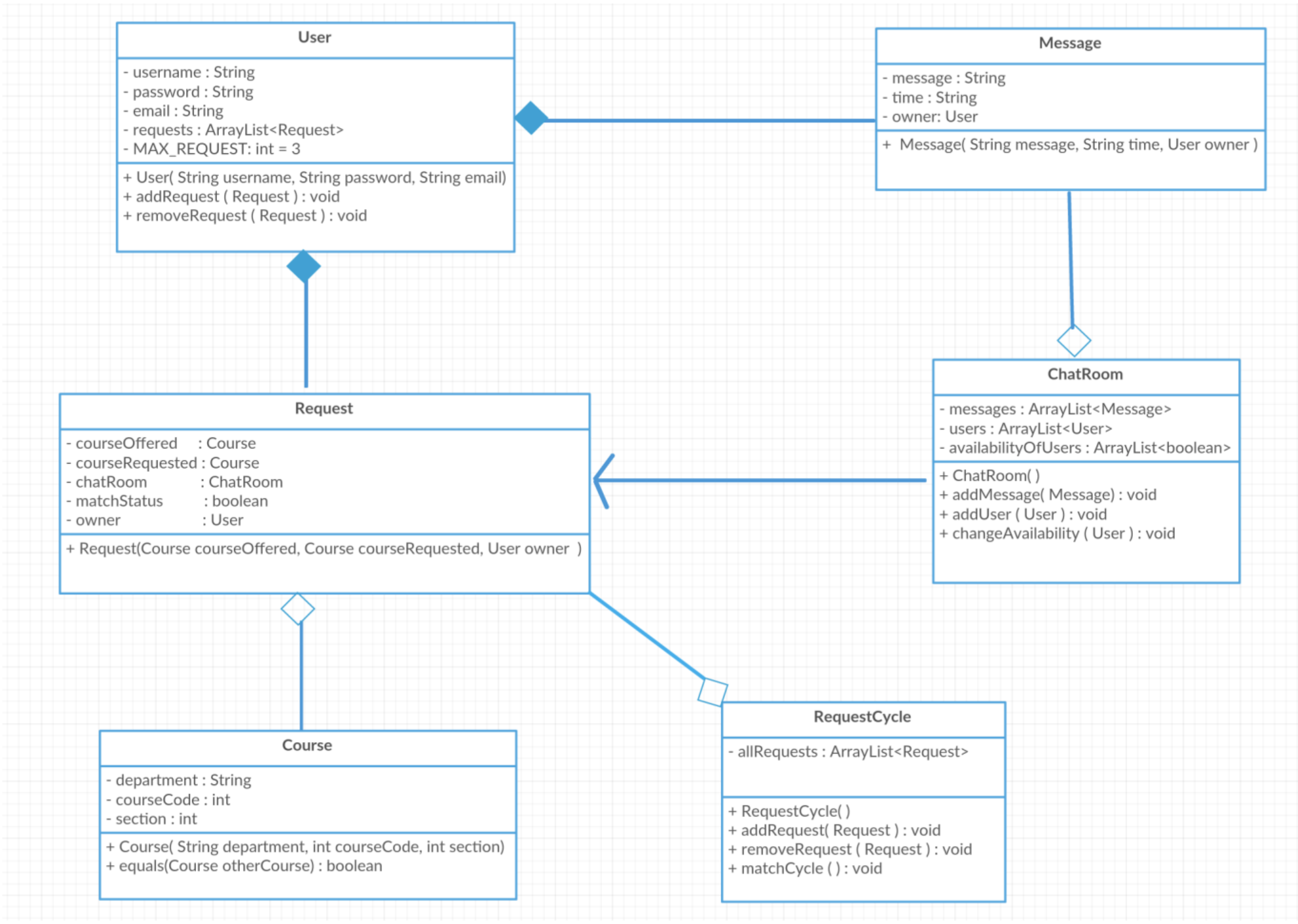
- MongoDb for the database.

- JSON to interact with MongoDb.

- Java Swing library for GUI.

- Bson to interact with MongoDb

2.2 Core Design Details


User
- username : String
- password : String
- email : String
- requests : ArrayList<Request>
- MAX_REQUEST: int = 3
+ User(String username, String password, String email)
+ addRequest (Request) : void
+ removeRequest (Request) : void
Message
- message : String
- time : String
- owner: User
+ Message(String message, String time, User owner)
ChatRoom
- messages : ArrayList<Message>
- users : ArrayList<User>
- availabilityOfUsers : ArrayList<boolean>
+ ChatRoom( )
+ addMessage(Message) : void
+ addUser (User) : void
+ changeAvailability (User) : void
Request
- courseOffered : Course
- courseRequested : Course
- chatRoom : ChatRoom
- matchStatus : boolean
- owner : User
+ Request(Course courseOffered, Course courseRequested, User owner)
Course
- department : String
- courseCode : int
- section : int
+ Course(String department, int courseCode, int section)
+ equals(Course otherCourse) : boolean
RequestCycle
- allRequests : ArrayList<Request>
+ RequestCycle( )
+ addRequest(Request) : void
+ removeRequest (Request) : void
+ matchCycle ( ) : void

This model created by UML diagram shapes. These shapes and definitions are following:

If two classes in a model need to communicate with each other, there must be link between them, and that can be represented by an association →

Aggregation implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist. —◇

Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House —◆

Note that every classes' instance variables have get and set methods. Therefore, it is not shown in both diagram and explanation of diagram.

1.User Class:

You can create new users by this class to interact with the program.

Instance Variables

1.username: String – This is user's unique username created in registration.

2.password: String – This is user's password with minimum 6 length created in registration.

3.email: String – This is user's Bilkent e-mail.

4.requests: ArrayList<Request> -- This is an arraylist of user's swap request in program.

5.MAX_REQUEST: int = 3 – This is constant showing allowed number of making request.

Public Methods and Constructors:

1.User(String username, String password, String email) – You can initialize the new user by providing username, password and email.

2.addRequest(Request): void – You can add new request into user’s arraylist of requests by submitting new request in the program.

3.removeRequest(Request): void- You can remove the request which already exists in arraylist of requests by cancelling the request in the program.

2.Request Class:

By this class, users can create new requests and cancel them.

Instance Variables:

1. courseOffered: Course – This the course user is offering.
2. courseRequested: Course – This is the course user is requesting.
3. chatRoom: ChatRoom – This is the chat room which is unique to this request.
4. matchStatus: boolean – This is the variable keeping the Request’s match status.
5. owner: User This is the user who created this request.

Public Methods and Constructors:

1. Request(Course courseOffered, Course courseRequested, User owner): void – You can initialize a new request by providing course offered, course requested and an owner.

3.RequestCycle Class:

To add requests into cycle and match with other requests, this class is used.

Instance Variables:

1.allRequests : ArrayList<Request> -- This is arraylist of all swap requests in the program.

Public Methods and Constructors:

1.Request Cycle() – This is the constructor of the RequestCycle with no explicit parameter.

2.addRequest(Request) : void – This method add new requests into system to match with other requests. matchCycle method is called at the end of this method to make matches again.

3.removeRequest(Request) :void – This method remove the request which already exists in the system. matchCycle method is called at the end of this method to make matches again.

4.matchCycle() : void – This method does real work. It matches all swap requests in the program. It creates a unique chatroom object and add this chatroom into the requests matched each other for each matching. So, the matched requests’ owner can access the same chatroom and different chatroom is created for each match.

4.ChatRoom Class:

Users who are in same chatroom can send message each other by this class's objects.

Instance Variables:

1.messages : ArrayList<Message> -- This is an arraylist storing all messages in the chatroom object.

2.users : ArrayList<User> -- This is an arraylist storing all users who can access this chatroom.

3.availabilityOfUsers : ArrayList<boolean> This is a synchronous arraylist with user arraylist. It shows whether the user attend the chatroom.

Public methods and Constructors:

1.ChatRoom() -- This is the constructor of Chatroom with given no parameter.

2. addMessage(Message) : void – User can add new message into chatroom by this method.

3. addUser(User) : void – By this method, users can be added into users arraylist.

4.changeAvailiability (User) : void – This method change availability status of the users. When the user enters the chatroom it makes true corresponding boolean in availabilityOfUsers arraylist. If the user lefts, then it makes false

5.Message Class:

By this class, user can create new messages.

Instance Variables:

1. message: String – This is the variable text of the message is kept in.

2. time: String – This is the sent time of the message.

3. owner: User – This is the user who sent this message.

Public Methods and Constructors:

1.Message(String message, String time, User owner): void – You can initialize a new message by providing message, time and an owner.

6.Course Class:

Due to this class, users can create any courses in Bilkent University

Instance Variables

1.department: String - This is the department of the course(**CS**-102/1)

2.courseCode: int - This is the courseCode of the course(CS-**102**/1)

3.section: int – This is the section of the course(CS-102/**1**)

Public Methods and Constructors:

1.Course(String department, int courseCode, int section): void – You can initialize the new course by providing department, course code and section number.

2.3 Task Assignment

Model

Request and User Class: Yavuz Faruk Bakman

RequestCycle Class: Hilmi Barut

ChatRoom Class: Alperen Oziş

Message Class: Atay Kaylar

Course Class: Furkan Ahi

UI

Login and Register Page: Atay Kaylar

Chatroom and make request page: Alperen Oziş

Main page, myRequest page: Furkan Ahi.

Information page: Hilmi Barut

Settings Page: Yavuz Faruk Bakman

Controller

Controller and database interaction: Yavuz Faruk Bakman and Alperen Oziş

2.4 Inspiration

We use creately.com to draw diagrams. [1].

The database system of the program is. [2].

We get information about the UML and definitions of UML relationship from the website. [3].

3. Summary & Conclusions

The program will be created Java and related libraries. Also, online database of MongoDB will be the significant part of the program by providing message, authentication and storing information system. All parts of the system interact with each other in a proper way.

References

1. https://creately.com/ Accessed at 02/12/2018.
2. https://www.mongodb.com Accessed at 02/12/2018.
3. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/ Accessed at 02/12/2018