CS102	**2021/22 Spring**	Project Group	2H
Instructor:	**Aynur Dayanık**		
Assistant:	**Haya Khattak**		

~ New Calendar App ~

1,2,3…

Kutay Tire, Alper Yıldırım, Alp Batu Aksan, Melih Rıza Yıldız, Süleyman Yağız Başaran

Criteria	TA/Grader	Instructor
Presentation		
Overall		

Detailed Design Report

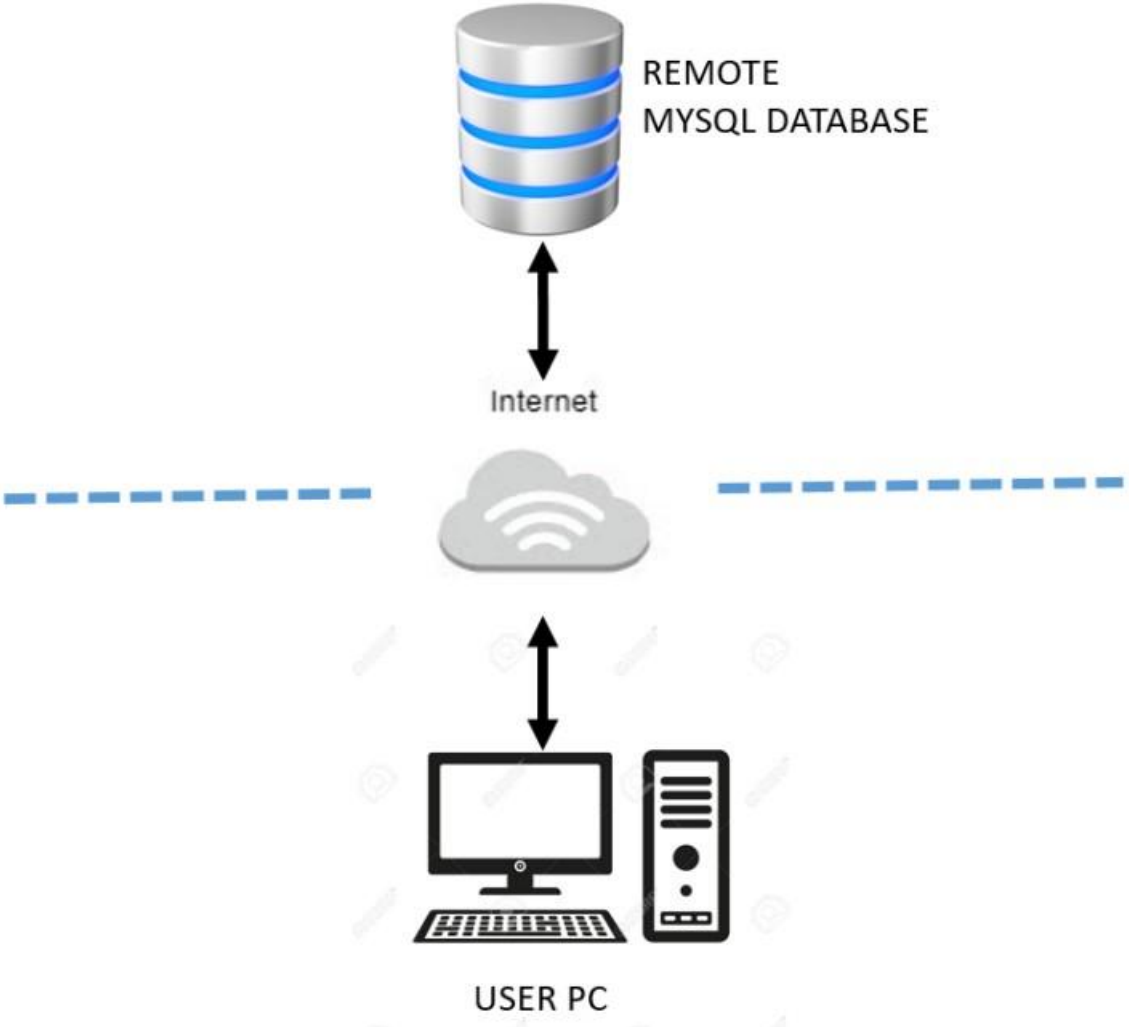
(Version 1.0)

1. Introduction

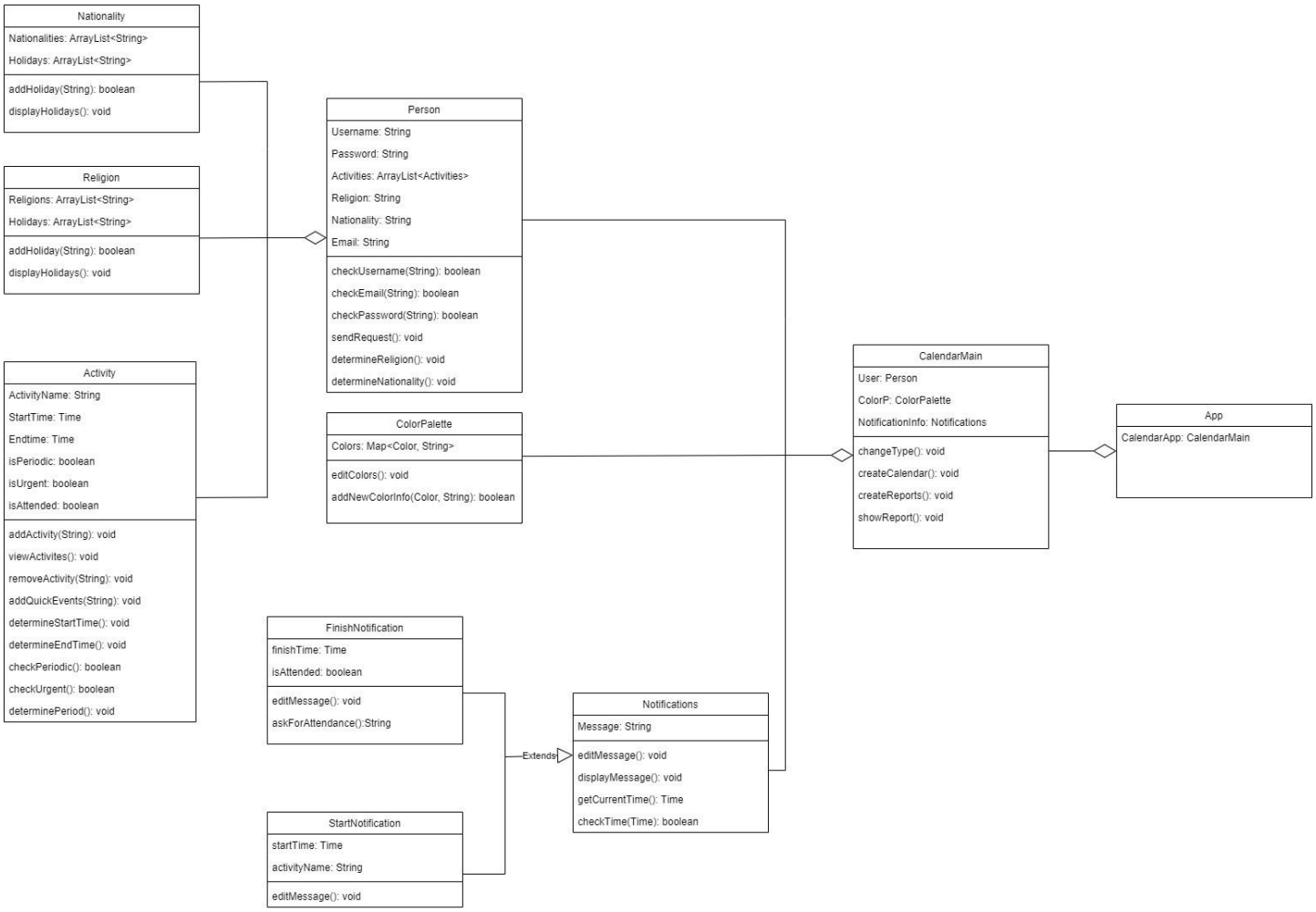
All of the classes that will be implemented are given in the below UML diagrams. There, the key methods and their functionalities are also explained briefly. To hold information, the MySQL database will be used with appropriate interfaces. From this database, the necessary information will be fetched. Furthermore, the task assignments of each group member are given in this report. The work is divided equally among the group members so that everyone gets an opportunity to both learn new things and rehearse what has been covered throughout the semester. Finally, the concerns regarding the project are given.

2. System Overview

In this app, MVC pattern is going to be used. View classes are going to be implemented separately by using the Java Swing class. Therefore, in this report, model and control classes are mostly shown. For the implementation of these classes, JavaFX will be used as it is also going to be a desktop-based app. For the database, MySQL is going to be used to hold the values of the user such as his/her nationality, religion, or the added activities on his/her calendar. Below, there is a simple diagram about how the database is going to work.



3. Core Design Detail



Nationality Class

Instance Variables:

- **Nationalities:** String array list that holds the predetermined list of nationalities that user can choose from.
- **Holidays:** List of holidays belonging to each nationality. For a specific nationality, specific indexes will be given. For instance, indexes from 0-5 will be Turkish holidays, 6-10 will be German holidays, etc.

Methods:

- **addHoliday(String):** This method enables the user to add a specific holiday if they choose. The index of this new holiday will be arranged based on the nationality of the user.
- **displayHolidays():** This method shows the table of the cultural holidays.

Religion Class

Instance Variables:

- **Religions:** String array list that holds the predetermined list of religions that user can choose from.
- **Holidays:** List of holidays belonging to each religion. For a specific religion, specific indexes will be given. For instance, indexes from 0-5 will be Islamic holidays, 6-10 will be Christian holidays, etc.

Methods:

- **addHoliday(String):** This method enables the user to add a specific holiday if they choose. The index of this new holiday will be arranged based on the religion of the user.
- **displayHolidays():** This method shows the table of the religious holidays.

Activity Class

Instance Variables:

- **ActivityName:** Name of the activity
- **StartTime:** Starting time of an activity in terms of Time class in JavaFX.
- **EndTime:** Ending time of an activity in terms of Time class in JavaFX.
- **isPeriodic:** Variable indicating whether the activity is periodic or not. This variable is given a value during the addActivity (String) method.
- **isUrgent:** Variable indicating whether the activity is urgent or not. This variable is given a value during the addActivity (String) method.
- **isAttended:** Variable indicating whether the user attended this activity or not. This information will be reached from the “FinishNotification” class.

Methods:

- **addActivity(String):** This method is the method called when the user wants to add an activity. In this method, the user also specifies the starting and the ending times of the activity.
- **viewActivities():** The user can see the list of his/her daily activities with this method.
- **removeActivity(String):** The user can remove an activity from the schedule with this method.
- **addQuickEvents(String):** This is the method that enables the user to add quick events. Later, the user can choose these quick events from the "Quick Events" page.
- **determineStartTime():** The method that is called to choose the starting time of an activity. This method will automatically be called inside the addActivity (String) method.
- **determineEndTime ():** The method that is called to choose the ending time of an activity. This method will automatically be called inside the addActivity (String) method.
- **checkPeriod ():** Method that checks whether the activity is periodic or not.
- **checkUrgent ():** Method that checks whether the activity is urgent or not. **determinePeriod ():** Method that is called to specify the period of an activity if it is periodic.

Person Class

Instance Variables:

- **Username:** String variable that holds the name of the user.
- **Password:** String variable that holds the password of the user.
- **Activities:** ArrayList of activities that holds the list of the activities of the user that will be displayed on the "View Activities" page.
- **Religion:** String variable that holds the religion of the user.
- **Nationality:** String variable that holds the nationality of the user.
- **Email:** String variable that holds the email of the user to send verifications if the password is forgotten.

Methods:

- **checkUserName(String):** This method checks whether the entered username is the same as the one in the database so that the user can log in.
- **checkEmail(String):** This method checks whether the entered email is the same as the one in the database so that new password requests can be sent to the mail address if the user forgets his/her password.
- **checkPassword(String):** This method checks whether the entered password is the same as the one in the database so that the user can log in.
- **sendRequest():** This method sends a new password request to the mail address of the user.

- **determineReligion():** This method enables the user to choose a religion from the array list of religions.
- **determineNationality():** This method enables the user to choose a nationality from the array list of nationalities.

Color Palette Class

Instance Variables:

- **Colors:** A map that matches every information such as attended, missed, or urgent with a specific color.

Methods:

- **editColors():** This method enables the user to arrange the mapping of information with a specific color if s/he doesn't want to use the default configuration.
- **addNewColorInfo(String, Color):** This method is called when a user wants to add new information matched with a different color.

Notifications Class

Instance Variables:

- **Message:** String variable that holds the message to be displayed.

Methods:

- **editMessage():** The method that edits the message if the user wants to change the default one.
- **displayMessage():** The method that is automatically called when the starting time of an activity comes. With this method, a notification is sent to the user.
- **getCurrentTime():** The method that gets the local time from the computer.
- **checkTime (Time):** The method that checks whether the starting time of an activity is the same as the local time so that a notification can be sent.

FinishNotification Class

Instance Variables:

- **finishTime:** Time variable that holds the finishing time of an activity so that the necessary notification can be sent when the right time comes.
- **isAttended:** Boolean variable indicating whether the user has attended the activity or not. Later, this information will be used in the reports.

Methods:

- **editMessage():** The method that edits the message if the user wants to change the default one. It is inherited from the superclass.
- **askForAttendance():** The method asks the user whether s/he attended the activity or not. The response of the user is saved in this method.

StartNotification Class

Instance Variables:

- **startTime:** Time variable that holds the starting time of an activity so that the necessary notification can be sent when the right time comes.
- **activityName:** String variable that holds the name of the activity which will be displayed in the notification.

Methods:

- **editMessage():** The method that edits the message if the user wants to change the default one. It is inherited from the superclass.

CalendarMain Class

Instance Variables:

- **User:** Person variable that holds the attributes of the Person class.
- **ColorP:** ColorPalette variable that holds the attributes of the ColorPalette class.
- **NotificationInfo:** Notifications variable that holds the attributes of the Notifications class.

Methods:

- **changeType():** This method changes the type of the calendar if the user chooses to do so.
- **createCalendar():** This method creates the main calendar with appropriate days, months, and buttons on it.
- **createReports():** This method creates the weekly report with necessary information about the activities displayed in the report.
- **showReport():** This method displays the weekly report if the user pleases to do so.

4. Task Management

Kutay

- Implementation of the Nationality and Religion classes.
- Implementation of Person class. Moreover, creating the login page and figuring out how to send new password requests to the email of the user.
- Learning about MySQL and creating MySQL interfaces.
- Creating “Religion”, “Nationality” and “Main Calendar” windows with Java Swing.

Yağız

- Implementation of Activity class.
- Learning about how to send notifications by comparing Time variables and local time.
- Creating “Activity View”, “Add Quick Events”, “Add Activity” and “Remove Activity” windows with Java Swing.

Melih

- Implementation of changeType () method by learning necessary conversions and attributes of other calendars (Islamic and Chinese).
- Creating “Change Type”, “About”, and “Contact” windows with Java Swing.

Batu

- Working with Yağız on, implementation of the Activity class. More specifically, focusing on the methods of periodic activities like the “determinePeriod ()” method.
- Implementation of createReports () and showReport () methods to create the weekly reports.
- Creating the “Reports” window with Java Swing.

Alper

- Implementation of ColorPalette class and its method.
- Learning about the “ColorChooser” class in Java to enable the user to edit the colors.
- Implementation of the “Notifications” class and its subclasses.
- Creating notification screens and the “Color Palette” page with Java Swing.

5. Database Relational Classes

In this project, some of the classes are connected to the database (MySQL) to store information. This information is contained in "Model" (according to MVC) part of the app. These classes are Activity, Person, Holiday and UserColor.

Activity Class

- In this class, all of the information about activities coming from users are being saved/stored by MySQL database. To explain why this is the case, every user is going to have different types of activities related to their hobbies. Therefore, the app needs to save these specified activities to users and show them without problems whenever user wants even after logging out.

Person Class

- There are 5 important properties in this class, these are: userid, email, password, nationality and religion. UserID, email and password are classic properties that every software uses and stores in their databases, and also in this app, the system is same.
- Nationality and religion properties also being saved by the app with the help of MySQL database because these information helps marking colors connected to the users' choices.

Holiday Class

- This class is also connected to the Nationality and Religion properties. Some days on the calendar are being marked by app depending on the choices of users and app saves this information on the database.

UserColor Class

- Users can customize colors of different situations such as "attended", "missed", "periodic", "upcoming" and etc. These colors are special to each user because there is no limit to picking colors. Therefore, color information is being stored in the database.

6. Class Interactions

This app is designed according to MVC rules/patterns; thus, there are three main parts. Model, View and Controller. Classes on the "Controller" side will take information from the user. Then, it will send these data to classes of "Model". After getting the response information, "Controller" will send these data to the "View" classes. Classes of "View" will create a last response information and send these to the user. To sum up, every class is working together in order to achieve best efficiency in the MVC system.

7. Concerns

There are some specific concerns regarding each class separately. For instance, for Religion and Nationality classes, override problems may occur if both a religious holiday and a cultural holiday coincide. Another concern regards the "Person" class. Keeping the username

and the password of a user correct is important and requires the use of the database. Also, this database needs upgrading for each activity of a person. Keeping track of these is crucial for the app to function properly. One final concern is about the Notifications class. The synchronization between the local time and starting/finishing times in the app is really important for the notifications. Otherwise, they will be sent at random times which is unwanted.

8. Summary & Conclusions

As a result, this report outlines how the basic classes are going to be implemented. Naturally, these classes aren't the final classes as some changes during the "Implementation" stage can occur. Still, they will be designed in parallel with the MVC pattern no matter what the changes may be. Furthermore, equal amounts of work are given to each member to be fair. Also, concerns regarding this design are given as a final remark.

9. Reflections

First of all, we have mostly accomplished our goals in terms of the desired implementation. Nevertheless, some features like adding the Chinese calendar was too difficult the implement. Project work was a little bit troublesome, as it was difficult to gather at a suitable time. However, we have learnt how to cooperate. The most difficult aspect was choosing the correct time for everyone. If we could start over, the planning of the project would be better as we have more experience now. We have spent approximately 1 week for the project and we are proud of the quality of work.

10. References

- Alder, Gaudenz. diagrams.net. 2000. JGraph Ltd. Accessed April 24, 2022. < https://www.diagrams.net/>