Samhitha Tarra
INF 122
January 19, 2022

Assignment 1-1 Original Design and Documentation

The calendar application I am designing utilizes various different classes that will handle different functionalities. The main class to begin with is the **User Class** which will contain user specific attributes such as username, all their user specific calendars they can view, what timezone they are in, and how they would like to view their calendar. This leads to the next class which is **AllCalendars Class** which is an entity that will essentially hold information about the various calendars that the user has access to or has created. This class will also contain certain additional arrays that will be updated to hold search results of events or filtered calendar types and accordingly update the list of relevant calendars to be displayed. In close relation to the AllCalendars class is the **Calendar Class** itself which will have to be assigned to a category upon creation for simplicity of filtering. The calendar class will contain attributes that specify whether it is public or private and an array that stores all the events within that calendar. The calendar will be able to be edited through its update function and also will allow the user to share an event that exists in the calendar if it is a public calendar. As previously mentioned, the Calendar class is composed of many events represented by the **Event Class** which hold details such as name of the event, details, start/end time, duration, etc. Events will be able to be created, edited, and removed from a calendar. Here is where event specific countdown timers are calculated as well. A function called calculateTimeUntilEvent() will calculate the time left until the event occurs based on current date/time. By storing this in each event, if a user ever wants to see how much time is left before the event occurs we can easily grab and display the data from that event object. Another type of the event class that will require more information is the **RepeatableEvent Class** which I extended for simplification and the elimination of adding unnecessary NULL fields to the Event Class as a whole. This way if a user specifies that they would like to create a repeating event then and only then will they be asked for a start date and an end date for which the event will keep repeating. Lastly is the **View Class** which is the overarching class that incorporates either monthly, daily, weekly, or yearly views. This class accounts for how the user would like to see their events and also contains an attribute for whether the display theme should be “Light” or “Dark”. By distributing the View Class amongst four different subclasses we can store different arrays of events and display the corresponding events when the user toggles to see a specific view type.

This design also contains flexibility to add and incorporate various future changes such as:

Adding a GUI (if you don't already have one):
Adding a GUI will not take much change as I already include a display() method for my classes so the most that needs to be altered is how I am showing the events/calendar.

Allowing users to add images to represent calendars or events:
This would be a relatively simple addition to the Calendar or Event class as they already contain various specific attributes. In order to implement this in the future we would simply need to add a function/attribute that will update the image attribute of a Calendar object or Event object and then update the display method to show it on the frontend.

Notifications are shown to a particular user of when a calendar event is shared by another user:
This change could actually very simply be implemented within the shareEvent() function that I already event to another user's event array however it could be modified to also send them an email/notificatio could be called as a helper function within shareEvent() to avoid complicating the class's code.

Email notifications to users about an event, which are sent a certain amount of time specified by a user:
This future change can be implemented by adding a function and attributes to the Event class. For example, upon a user creating an event we can prompt for whether they would like a notification for this event, mark that boolean as True, and if True then we prompt asking how many minutes before the event they would like to be notified. Once all this information is stored and updated within the Event object, we can write and call a function that will send the user an email when the current time is set to equal the notification time or some similar logic.

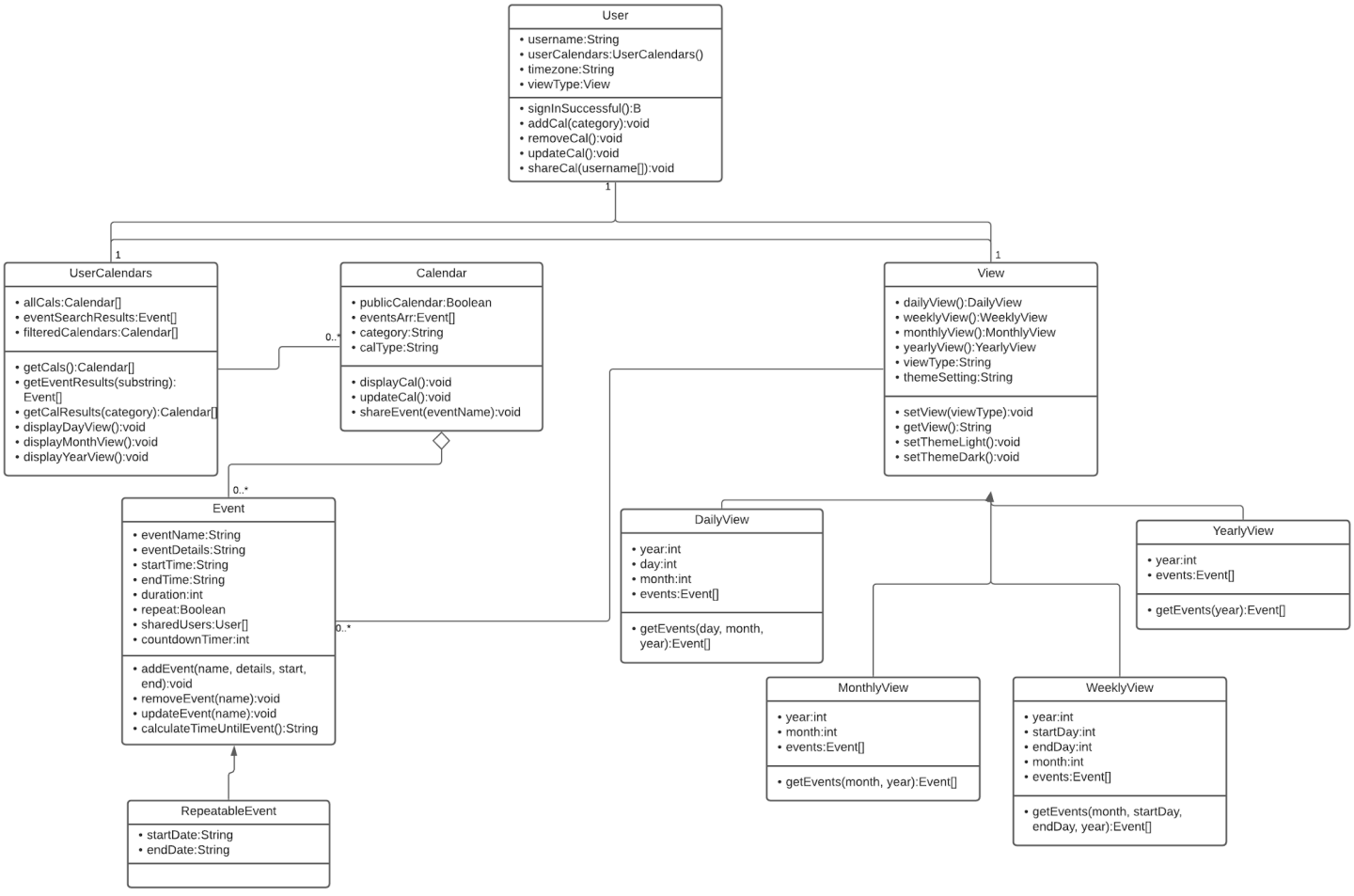
The ability to add notes or a description for a particular event or calendar:
I have already accounted for this in my class diagram by having a details attribute for the event class however if we wanted to be able to add further notes then we can always create another String variable that allows for this. Additionally, if we wanted to add notes/description for a calendar event it would be the same process of storing a variable in that object as well.

Expansion of the settings or configurations of the app:
This could be accounted for within the View Class as it would simply be an expansion very similar to setting the theme of the app to dark versus light. If we add the

attributes/functions we needed to the View Class and make any week/month/day/year specific changes to the subclasses then it should configure the application accordingly.

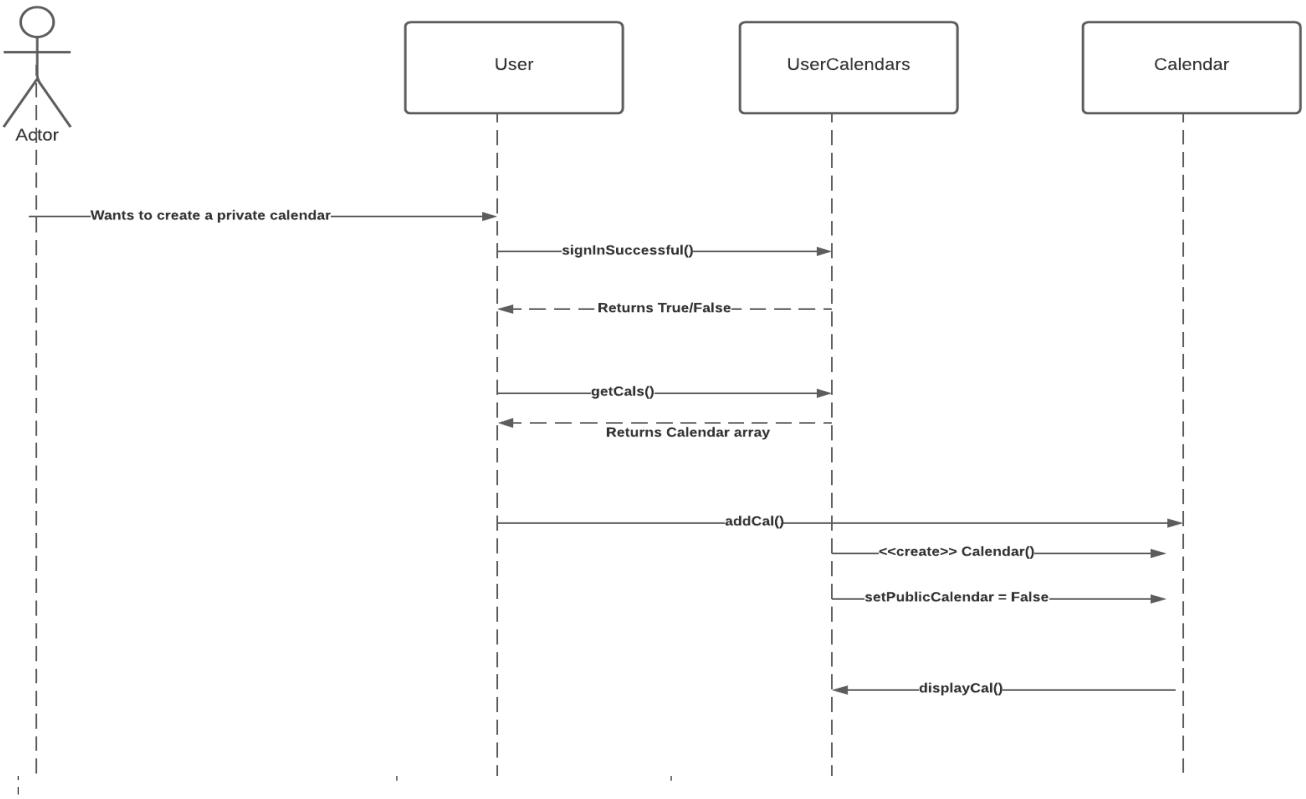
The ability to support different languages:
While currently the application will display all text in English, when building a Java application we can offer the user the ability to select a language upon account creation. By offering a locale choice to the user upon logging in, using Java libraries we can easily display all following text in that set language. In the View Class again we can even add a variable that keeps track of the language to display text in.

UML CLASS DIAGRAM:



SEQUENCE DIAGRAMS:

Create Private Calendar



Update an Event