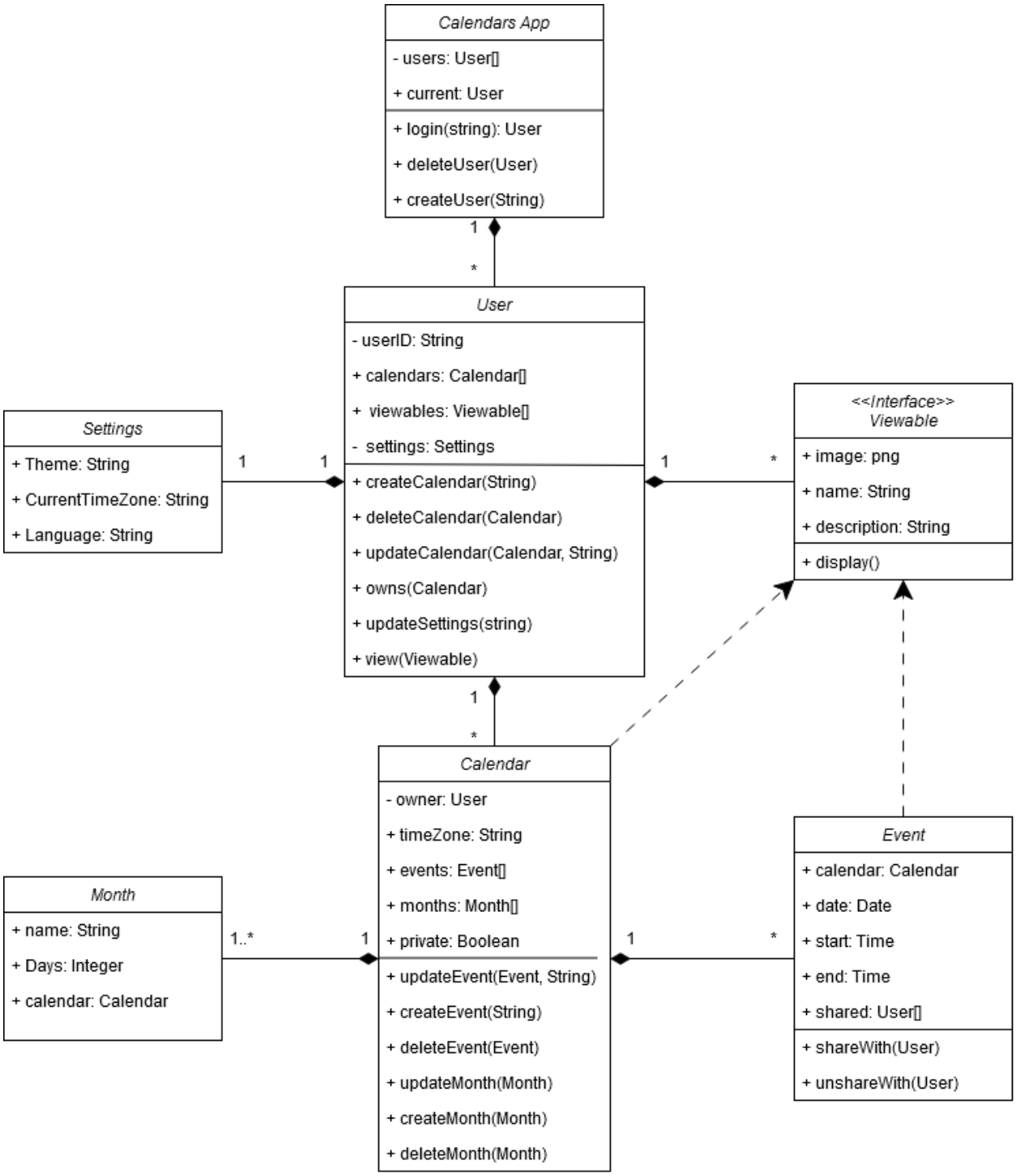
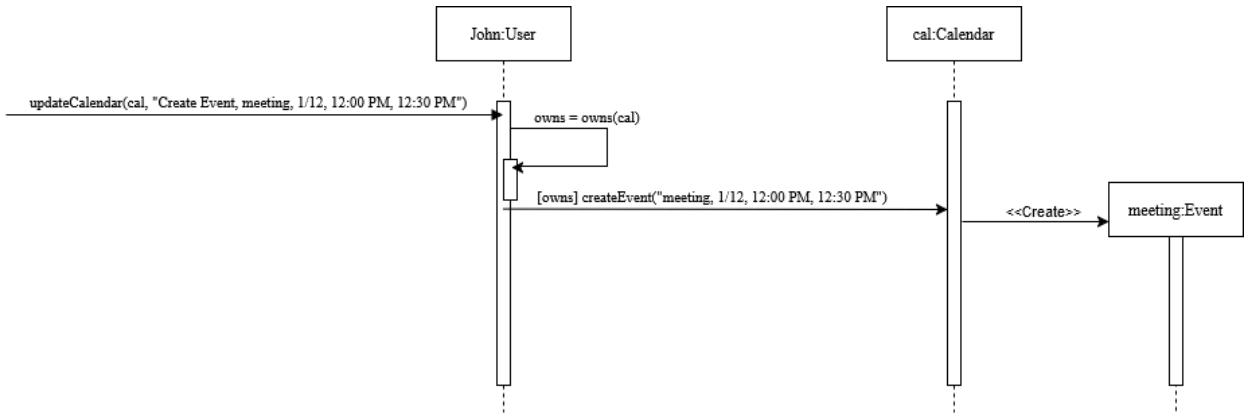
UML Class Diagram



UML Sequence Diagram (User Adds Event to Calendar)



UML Documentation

UML Class Diagram

My calendars design focuses on three classes that have singular primary responsibilities of managing objects they “own”.

- The Calendars App class manages only users
- The User class manages only calendars
- The Calendar class manages only months and events.

Additionally, most classes have compositional relationships with one other class. I reached this conclusion as in a calendar app, the lifetimes of objects would depend on other managing objects to exist. An Event or Month must be a part of some Calendar, a Calendar must be created and owned by a User, and a User is dependent on the lifetime of the app. I also implemented a viewable interface that is implemented by objects that can be displayed to the User. This design makes it that responsibility and calls are distributed across classes and avoids a single large manager class.

- **Class Descriptions**
    - Calendars App
        - Manages User objects and stores them
        - Allows for selecting the current user by logging in
    - User
        - Manages Calendars that belong to the user and Viewable objects that have been shared with the user
        - Can view Viewable objects
        - Can set settings to modify the application
    - Calendar

        - Manages Events and Months
        - Contains attributes for whether it is public or private and timeZone
        - Should the timeZone attribute be changed through updateCalendar(), corresponding updateEvent calls will be made
    - Settings
        - Object unique to each user that remembers their settings
    - Month
        - Represents a custom month that can be part of a Calendar
    - Event
        - Represents event objects that can be part of a Calendar
        - Possesses a shared attribute to list which Users can see the event.

UML Sequence Diagram

My UML Sequence Diagram illustrates a call on John.updateCalendar(...). This call first checks if cal is within the calendars that John owns before calling cal.createEvent(...). This will result in the creation of a new Event object that will be stored in cal. Information about what type of update is occurring is passed down in a string parameter that is parsed.

Future Flexibility

1. My design has the flexibility to support descriptions as it implements a description attribute to Viewable objects (Events and Calendar) which will be displayed on calling the objects display(). The description is set on object creation and can be changed by calling the update functions.
2. My design has the flexibility to support images as it implements an image attribute to Viewable objects (Events and Calendar) which will be displayed on calling the objects display(). This is set in the same way as the description attribute.
3. My design has the flexibility to support different languages by implementing a language option that will be saved per user in the settings and can be updated through updateSettings().
4. My design has the flexibility to support custom calendars besides Gregorian through the creation of custom Months, this allows users to specify how many “Months” the calendar has and the days in each Month