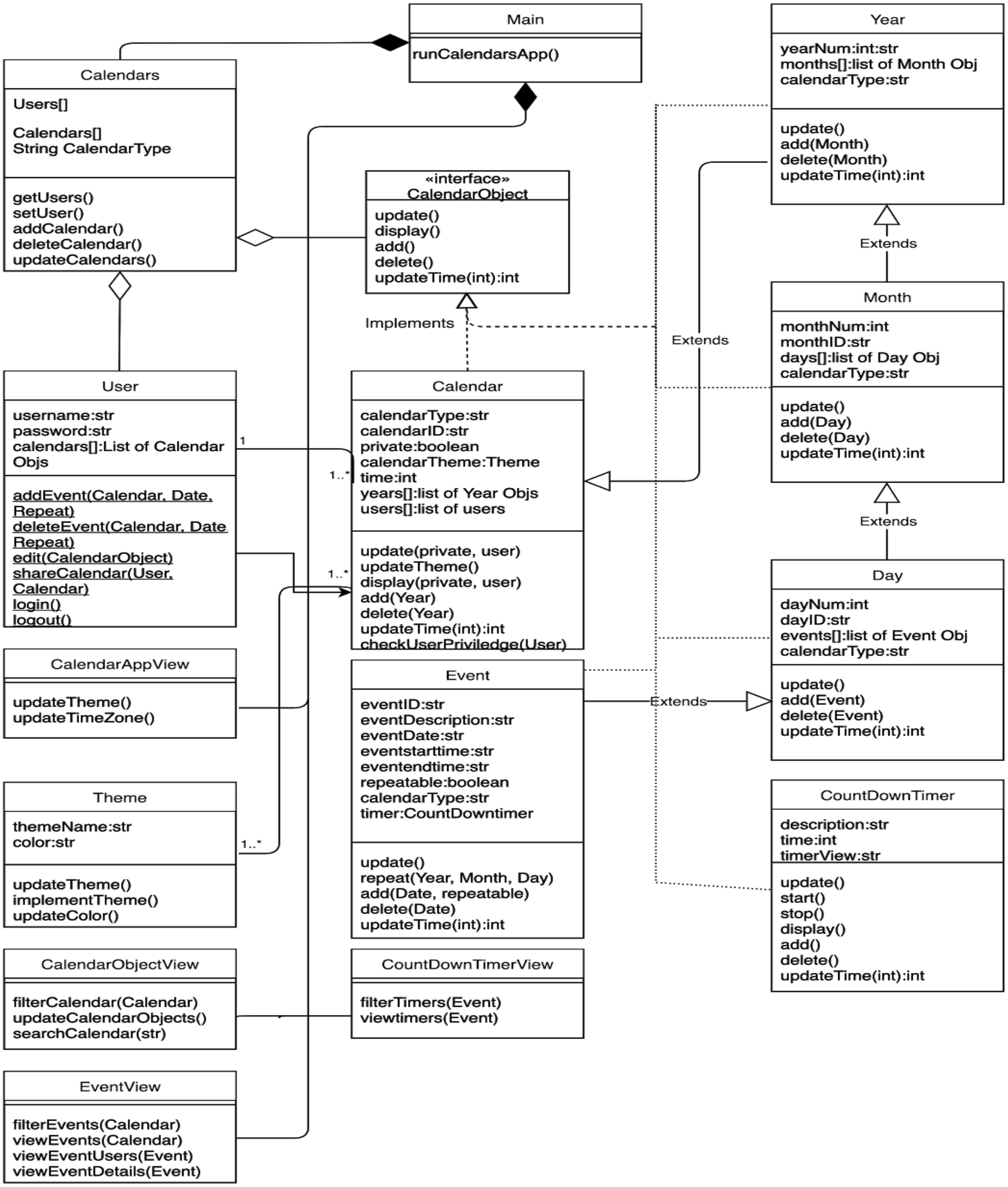
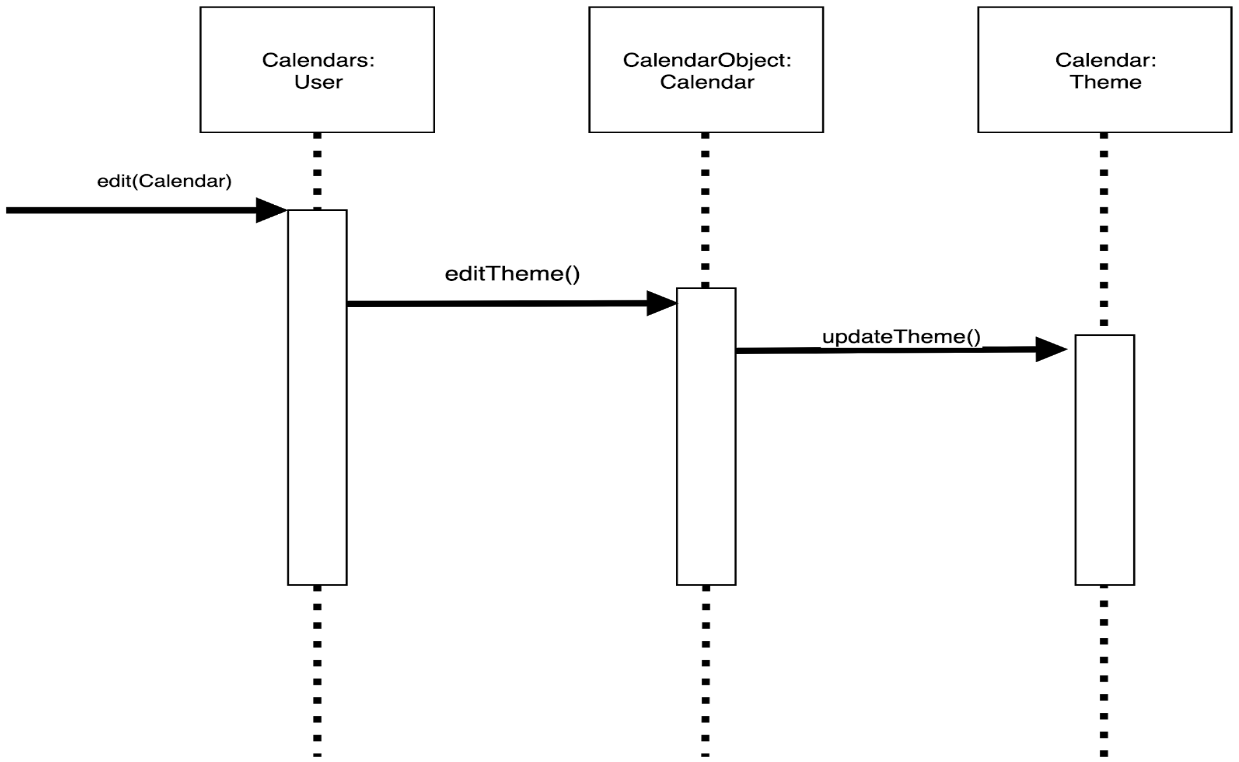
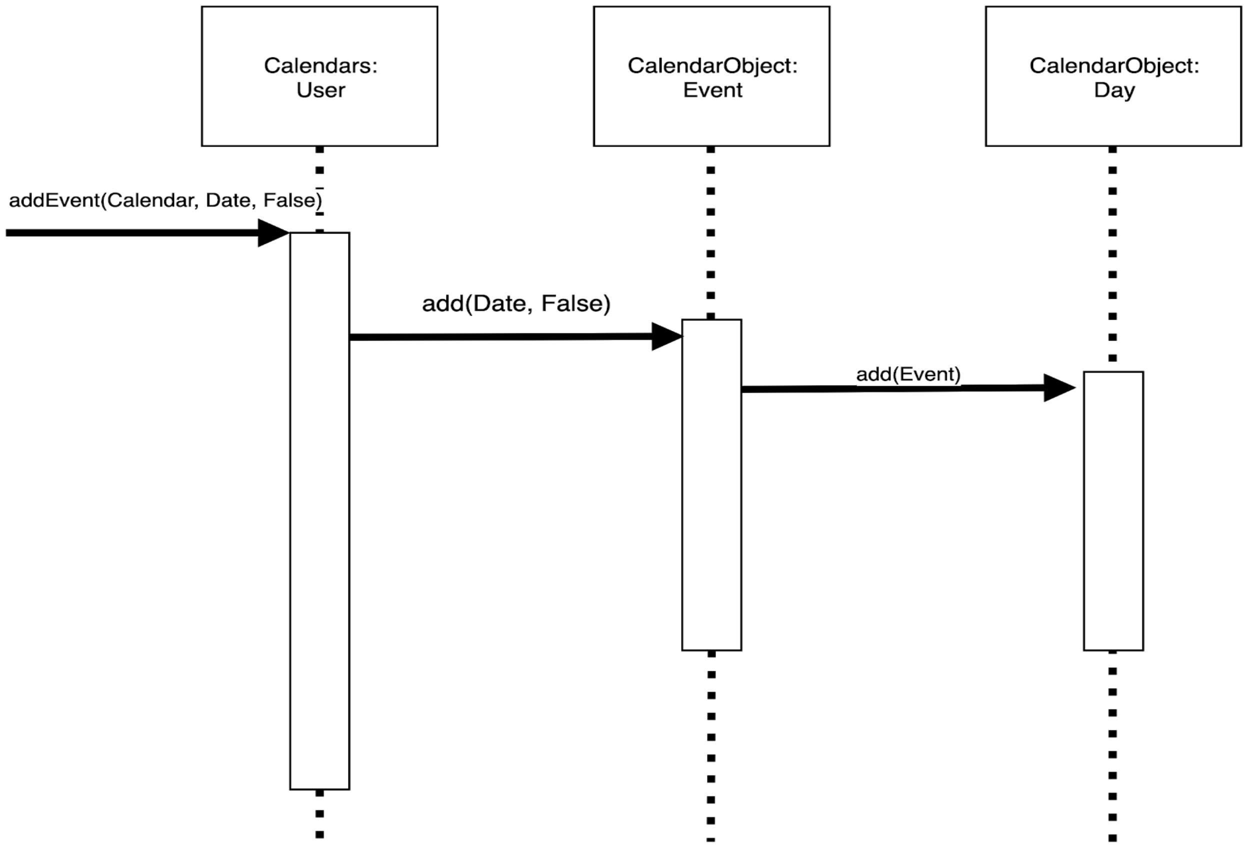
Assignment 1-1




Calendars: User
CalendarObject: Calendar
Calendar: Theme
edit(Calendar)
editTheme()
updateTheme()


Calendars: User
CalendarObject: Event
CalendarObject: Day
addEvent(Calendar, Date, False)
add(Date, False)
add(Event)

- **Main**
    - Java main object to run the application, object, and views.
- **Calendars** compose Main
    - Application object that governs the entirety of calendar objects and user interaction
    - Contains lists of users and calendars w/ calendar objects
    - Methods given for updating, editing, and displaying
- **User** aggregates from Calendars App
    - Username and password attributes for future authentication
    - Record of calendars shared and owned by user
    - Methods for adding and viewing calendar objects (Specific dates and Events)
    - Methods for user login and logout for future features
- **CalendarObject interface**
    - Abstract methods for calendar objects (Calendar, Year, Month, Day) to be implemented
    - Each CalendarObject can update, change, delete, and update its time.
- **Calendar** implements CalendarObject
    - Stores calendar type (Gregorian, etc.), unique identifiers, Theme object for UI, and lists of years and users, who have permission to view and edit.
    - Aside from the overloaded methods from CalendarObject interface, there is a unique method for checking if a users with permission to edit the calendar.
- **Year** implements CalendarObject
    - Inherits from Calendar Obj to gain access to Calendar attributes
    - Overrides methods to update Year's own attributes
        - Can add or delete Months by user or system itself
- **Month** implements CalendarObject
    - Inherits from Year Obj to gain access to Year attributes
    - Overrides methods to update Month's own attributes
        - Can add or delete Days by user or system itself
- **Day** implements CalendarObject
    - Inherits from Month Obj to gain access to Month attributes
    - Overrides methods to update Day's own attributes
        - Can add or delete Events by user or system itself
- **Event** implements CalendarObject
    - Inherits from Day Obj to gain access to Day attributes
    - Overrides methods to update Event's own attributes
        - Can add or delete Events by user or system itself
    - Unique methods for Event that allows itself to be repeatable using timer and time attributes

- **CountDownTimer** implements CalendarObject
  - Stores time and timerView for display as well as a string description for title and details, which can be viewed from CountDownTimerView
  - Methods for updating, starting, stopping, adding, and deleting
  - UpdateTime method used for updating the view of CountDownTimer
- **Theme** associates with Calendar Object
  - Attributes store theme name and color.
  - Methods created for updating Theme and color.
  - ImplementTheme for triggering UI to change CalendarAppView
- **CalendarAppView** compose Main
  - View Object that only stores methods for changing the UI and time zone of user's application
- **CalendarObjectView** compose Main
  - View Object that allows the User to search and filter for Calendar Objects from the UI.
  - User can also manually force update all Calendar Objects.
- **EventView** compose Main
  - Specific view object that can search, filter, and view for details of events within a Calendar Object or Event Object
- **CountDownTimerView** compose Main
  - View Object for filter and viewing timers of specific Event Objects.

Flexibility:

The flexibility shown in my UML class diagram has objects separated from each other for easier maintainability of specific Calendar Objects or the overall application itself. The future changes specified in the assignment can be easily handled by adding attributes or methods to different objects required from the changes. Each object is not too interdependent on one another, but they share attributes for accessibility to changes within the overall structure of a calendar. Overall, I find that breaking down the high level objects into smaller components allows for flexibility to specific functionalities from future requirements.