Calendar App Implementation Changes

While the original UML class diagram was very helpful in creating a strong foundation for the Calendars app implementation structure, I found that several components were not necessary and others needed some small changes in order for the app to actually work and connect seamlessly. I tried my best to adhere to the original design layout, keeping most of the structure the same, but the major changes are described below:

- I did not include the Year, Month, and Day classes outlined in the original design because having 3 separate objects to represent one instance of a date was inefficient and unnecessary. Instead, I saved a date string in the Event class and parsed through the string to get each month and year (the day is already stored in the date string). I also created 3 Map objects in the Calendar class (dayEventMap, monthEventMap, and yearEventMap) that efficiently insert an event into the correct day, month, and year key each time a new event is created. They also get removed dynamically if an event is deleted. This allowed for a very seamless transition into the view functionality, as all I needed to do was display the map data for each type of view the user could select (day, month, year) by iterating over the map and printing it out, because each possible view option was already mapped to an array of Event objects that matched the events' day, month, or year. I believe this design choice was an efficient one.

- I did not include the Theme class outlined in the original design because it was unnecessary for a command-line interface, and even for a GUI interface. The Theme class in the original design only has 2 attributes: themeName and color. Though it is good practice to modularize functionality, Calendar theme is just more of an attribute itself than a whole class object. A Calendar theme only has 2 options: light mode or dark mode. Therefore, I implemented the theme as an attribute in the Calendar class. My app handles updating the theme in the same manner as updating any other Calendar setting: in the updateCalendarSettings() method in the CalendarApp class.

- I decided to rename the "Calendars" class "CalendarsApp" instead to prevent confusion when comparing it to the "Calendar" class, as well as making it a distinct app object that directly handles the main and most general functionality of the app. This class follows the same structure as the original design, the only differences being that it only stores User attributes as opposed to Calendar attributes in the original design, and that a Scanner type input is stored as well. I think that it is a better design choice to separate Users from Calendars in the CalendarsApp class in order for better modularity. The Calendars class handles Events, the Users class handles Calendars, and the CalendarsApp class handles

Users. It is a succinct hierarchical structure that allows for less confusion and more isolated changes.

- I chose to save unique userIDs, calendarIDs, and eventIDs as integers instead of strings as outlined in the original document. This ultimately was a much more helpful and useful design choice because it allows for easy information retrieval by using the IDs as indexes instead of just identification strings that can't be used to easily fetch array data. Also, I added a public static integer variable to each class (User, Calendar, and Event) that keeps track of the current ID of the object and automatically increments each time a new instance of each of the classes is created (and decremented when deleted).

- I chose to implement collections of objects as ArrayLists instead of regular Java arrays as outlined in the original document because it is a much more beneficial and efficient way to store and retrieve data, especially since most of the objects created need to be stored in dynamic way because there is much addition and deletion of these objects and can get very large with many Events, Calendars, and Users.

- I did not include login and logout functionality designed in the original document because it was not within the scope of this project to include such functionality. The requirements stated that each session is terminated completely when ended.

- I did not implement the Countdown Timer component as it was frankly quite too complicated to be implemented for the first iteration of this app. Handling just Calendars, Users, and Events alone through the command-line interface was a major challenge and I would say getting this functionality to work is a good first step in itself. After refactoring the code and improving on the existing foundation, it would be easier and more realistic to add a Countdown Timer, especially with a GUI implemented first.