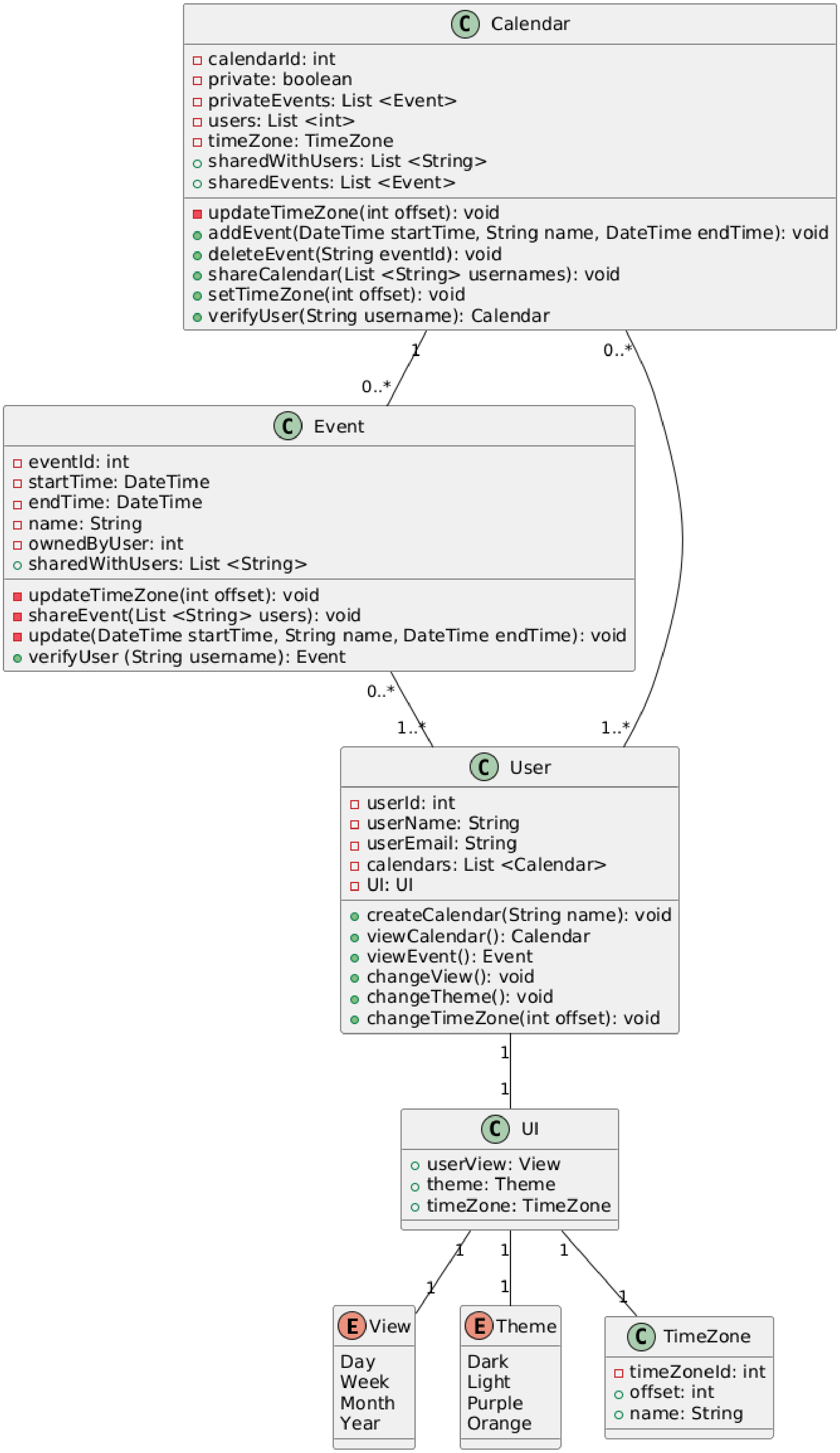
Class UML Diagram

```
@startuml
' Define enums
enum View {
    Day
    Week
    Month
    Year
}

enum Theme {
    Dark
    Light
    Purple
    Orange
}

class UI {
    + userView: View
    + theme: Theme
    + timeZone: TimeZone
}

class Calendar {
    - calendarId: int
    - private: boolean
    - privateEvents: List <Event>
    - users: List <int>
    - timeZone: TimeZone
    - updateTimeZone(int offset): void
    + sharedWithUsers: List <String>
    + sharedEvents: List <Event>
    + addEvent(DateTime startTime, String name, DateTime endTime): void
    + deleteEvent(String eventId): void
    + shareCalendar(List <String> usernames): void
    + setTimeZone(int offset): void
    + verifyUser(String username): Calendar
}

class Event {
    - eventId: int
    - startTime: DateTime
    - endTime: DateTime
    - name: String
    - ownedByUser: int
    - updateTimeZone(int offset): void
    + sharedWithUsers: List <String>
    - shareEvent(List <String> users): void
    - update(DateTime startTime, String name, DateTime endTime): void
    + verifyUser (String username): Event
}

class User {
    - userId: int
    - userName: String
    - userEmail: String
    - calendars: List <Calendar>
    - UI: UI
    + createCalendar(String name): void
    + viewCalendar(): Calendar
    + viewEvent(): Event
    + changeView(): void
    + changeTheme(): void
    + changeTimeZone(int offset): void
}

class TimeZone {
    - timeZoneId: int
    + offset: int
    + name: String
}

UI "1" -- "1" View
UI "1" -- "1" TimeZone
UI "1" -- "1" Theme
User "1" -- "1" UI
Calendar "0..*" -- "1..*" User
Calendar "1" -- "0..*" Event
Event "0..*" -- "1..*" User
@enduml
```


Calendar
calendarId: int
private: boolean
privateEvents: List <Event>
users: List <int>
timeZone: TimeZone
sharedWithUsers: List <String>
sharedEvents: List <Event>
updateTimeZone(int offset): void
addEvent(DateTime startTime, String name, DateTime endTime): void
deleteEvent(String eventId): void
shareCalendar(List <String> usernames): void
setTimeZone(int offset): void
verifyUser(String username): Calendar
1
0..*
0..*
Event
eventId: int
startTime: DateTime
endTime: DateTime
name: String
ownedByUser: int
sharedWithUsers: List <String>
updateTimeZone(int offset): void
shareEvent(List <String> users): void
update(DateTime startTime, String name, DateTime endTime): void
verifyUser (String username): Event
0..*
1..*
1..*
User
userId: int
userName: String
userEmail: String
calendars: List <Calendar>
UI: UI
createCalendar(String name): void
viewCalendar(): Calendar
viewEvent(): Event
changeView(): void
changeTheme(): void
changeTimeZone(int offset): void
1
1
UI
userView: View
theme: Theme
timeZone: TimeZone
1
1
1
1
1
1
View
Day
Week
Month
Year
Theme
Dark
Light
Purple
Orange
TimeZone
timeZoneId: int
offset: int
name: String

Explanation

There are three main classes in this diagram Calendar, Event, and User. Users have the ability to create Calendar and Events. Events are associated with one Calendar, but Calendars can store many events. Calendars and Events are associated with the user who origically created them, although other users may be able to view and edit them if they are in the sharedWithUsers list.

Users have a unique userName associated with their account which is how Calendars keeps track of which Calendars and Events Users have created and have access to. Users can edit any Calendars and Events that they have created or have access to.

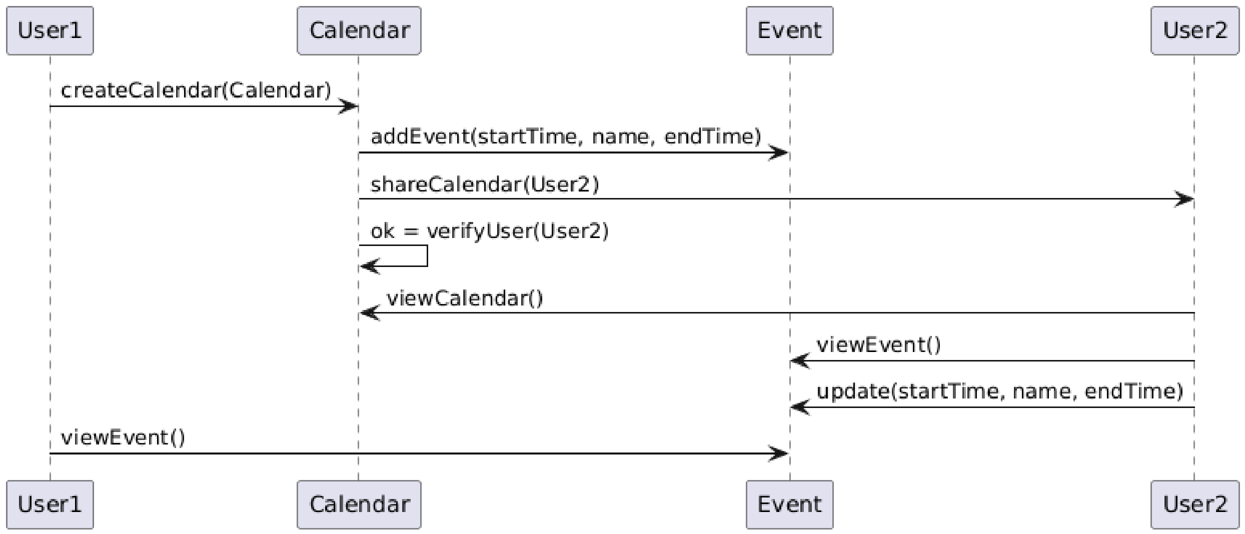
When a User creates a Calendar they must choose a name for it. A User can create an event within a calendar. When a User creates a Event they must enter a startTime and a name, and they may optionally add an endTime. If an endTime is not added, it will default to an hour after startTime. All Events in a Calendar will be created in the TimeZone of the User. If Events are shared between Users, they will appear to each User in the User's local time.

Users may share singular Events or entire Calendars with each other. If Users share single Events, only that Event from the Calendar will be shared. If Users share a Calendar, all Events within that Calendar will be visible and editable to the parties it is shared with. Users may also make their Calendars public, which means all Events in their Calendar can be viewed by any other Users on the Calendars app. Whether a Calendar is public or private is indicated by the private boolean variable.

Users have a UI attribute which stores UI settings such as TimeZone, Theme, and View. The User's TimeZone determines which timezone they view Events in, their Theme determines the application theme in terms of colors, and the View determines if the User can see all of their Events on their screen for the day, week, month, or year without scrolling.

Sequence UML Diagram

```
@startuml
User1 -> Calendar: createCalendar(Calendar)
Calendar -> Event: addEvent(startTime, name, endTime)
Calendar -> User2: shareCalendar(User2)
Calendar -> Calendar: ok = verifyUser(User2)
User2 -> Calendar: viewCalendar()
User2 -> Event: viewEvent()
User2 -> Event: update(startTime, name, endTime)
User1 -> Event: viewEvent()
@enduml
```



Explaination

This is an example of two Users interacting with the same Calendar and Event. A User, User 1 creates Calendar called Calendar. Within the Calendar User1 creates an Event called Event. Then, User1 shares the entire Calendar and all the Events within it with User2. Now User2 attempts to view the Calendar. Calendar first has to verify that User2 is on its approved sharedWithUsers list. When Calendar determines that User2 has permission to view it, User2 is views the Calendar. User2 then tries to view the Event within the Calendar. Since, User2 has permission to the Calendar, the Event within the Calendar does not need to check User2's permissions. User 2 then updates the Event by changing one or more of the startTime, name, and endTime. This action will update the Event so that when User1 views it, they will see the changed event.

Flexibility

"The ability to add notes or description for a particular event or calendar": This future change can be easily added to my design by adding attributes called "description" and "notes" to both the Calendar and Event class. It would also be easy to add to a GUI display by creating room to display a certain number of characters. The addEvent and createCalendar can have added arguments to allow users to add descriptions and notes upon creation of Events and Calendars.

"Expansion of settings and configurations of the app": This future change can be implemented via the IU class that is stored in every User instance. The UI class already stores the User's TimeZone, Theme, and View. Additional settings such as text size, color, or notifications can be stores in this IU class. The UI class is unique to each user, which means if many users are sharing one device as stated in the assignment details, they can all maintain their preferred exeperience without additional work.

“The ability to add simple to-do lists that interface with calendars”: This future change could be implemented with an additional To-Do class. To-Do instances can be either stored in specific Calendars, or stored in a specific To-Do Calendar which does not hold other events depending on what our users want. The To-Dos would act similarly to Events, but may not need start and end times, and instead perhaps a due date. To-Do instances may also need additional features such as the ability to mark them as complete. Depending on the approach, the Calendar class can be modified to hold To-Dos as well as Events and display To-Dos in a separate box in the timeframe that they are associated with.