General Structure Changes:

Change: Visibility Enum, I created an enum for Visibility because although the original diagram shows a Visibility type, there is no accompanying type reference. This was needed because or else there would be an error regarding the lack of a Visibility type

CalendarsApp Class Changes :

Change: Added a method on CalendarApp Class to createUser(name: string). Currently in the design there is no way for a User to be created. This was needed because without a createUser() method, there will never be a User to manage, and thus no app.

Change: Added a currentUser attribute to the CalendarApp class. I made this change because there is currently no way in the UML design to track which user is accessing the application. This was needed because otherwise the CalendarsApp would have no way of tracking the current user, a nd thus no way of showing specific events based on the user.

Change: modified method getUser(user: User) to be getUsers(): List<User> because the original method requires you to pass in the User object you are looking for into the method to get it. This makes no sense as you already have the User. Instead we required a getUsers method because there was no way to get a single User for the application let alone all the users.

Change: added method loginUser(name: string): boolean to add User login functionality. This change was necessary because the UML diagram previously provided no way of supporting multiple users aside from the storage collection.

Change: added method logoutUser() to add User logout functionality. This change was necessary because the UML diagram previously provided no way for the application to switch between Users.

User Class Changes :

Change: modified addCalendar(calendar: Calendar): void to be addCalendar(title: String): void, I made this change because the current UML digram requires you to create a Calendar Class and pass it into the User class to “Create a calendar”, however the User class should be responsible for this task alone it should only take initialization values and create the Calendar Class by itself. This was needed in order to complete the implementation while keeping the design authors original intentions in mind.

Change: modified removeCalendar(calendar: Calendar): void to be removeCalendar(title: String): void. I made this change for the same reason as **(Change Above)**

Change: removed method updateCalendar(calendar: Calendar) because it was unnecessary. Modification to the calendar happens with getter methods, setters method as well as the create and removeEvent methods within the Calendar Class itself.

Calendar Class Changes :

Change: removed deletedEvent: Event attribute because there was no functionality to support its implementation. This was needed because it was not implemented in the final code.

Change: removed TimeZone and Theme attributes because there was no functionality to support their implementation. This was needed because the attributes were not present in the implementation.

Change: removed updateCalendar(event: Event) method because the method is redundant, updating the calendar happens with getter methods, setters method as well as the create and removeEvent methods within the Calendar Class itself.

Event Class Changes :

Change: added name attribute to the event so that it has a unique identifier when retrieving or removing the event. This change was needed because otherwise after an event is created, the only unique identifier for the event is the Object ID from Java, and its start and end times. Which would not be enough when searching for an event in a sea of many.

Change: Replaced DateTime class (which doesn't exist in java) with the Date class that java implements by default. This was needed because. DateTime doesn't exist in java and the Date Class allows us to set and get the Date and Time.

Change: Removed the shareUser method because there was no way to store or retrieve shared Users in the diagram. This change was needed because the implementation does not provide a shareUser method to the Event Class.

Configuration Class Changes :

Change: Removed the Configuration class as there was no way mentioned in the diagram on how to apply the configuration to the application. The Class also was being saved as an attribute nowhere and had no purpose.

Theme Class Changes :

Change: Removed the Theme class as there was no way mentioned in the diagram on how to apply the theme to the application. The association between the Theme class was also unclear and caused confusion when implementing.