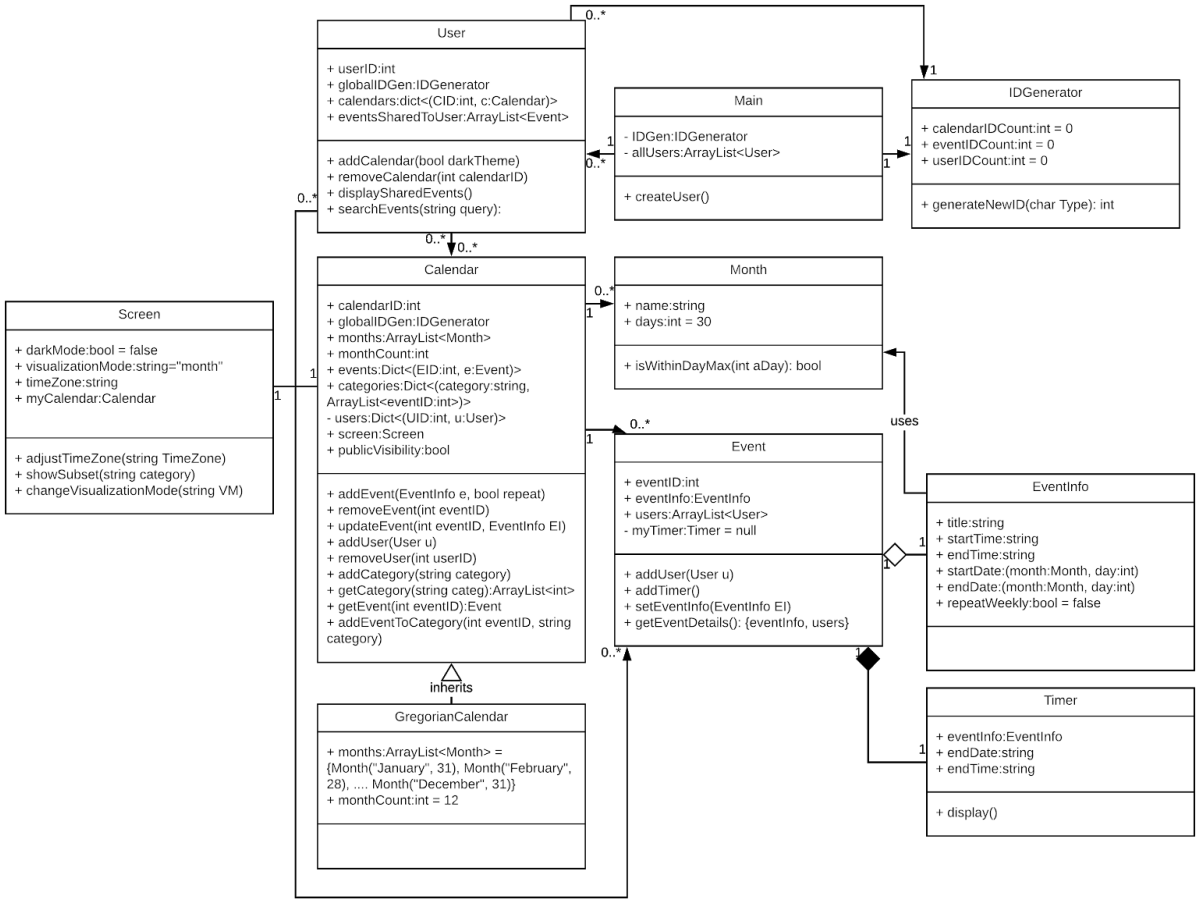
Maha Sheikh
11099501
INF 122 HW 1-1

Class Diagram



Class Diagram Documentation

Class Information:

Main: 'Main' is the overall controller class of the program. It instantializes one IDGenerator for the entirety of the program, which keeps track of all the unique IDs, and sends the same IDGenerator to any instances of the User class. It also has an arraylist of all users throughout the program.

IDGenerator: An instance of 'IDGenerator' keeps a tally of the last generated ID counts for the whole program. generateNewID(char Type) takes a character, 'C', 'E', or 'U' and returns an integer based on the member variables 'calendarIDCount,' 'eventIDCount', and 'userIDCount.'

Essentially, it returns the increment of these member variables, to ensure no ID is used more than once for their individual type.

User: 'User' is passed its user ID and the global IDGenerator, by the Main class which created it. A User can create & add calendars to their 'calendars' member variable dictionary using the addCalendar(bool darkTheme) method. The 'calendars' dictionary's keys are calendarIDs and value is the Calendar.

The removeCalendar method takes a calendarID, and removes the (ID, Calendar) pair from the 'calendars' dictionary by matching its key.

A User also has an arraylist of shared Events, (these do not exist on any of this User's calendars but rather the original owner's calendar), which can be displayed using its corresponding method, 'displaySharedEvents'. Finally, the 'searchEvents' method uses a string query and goes through each Calendar and its Events to sort through related events, as well as the User's shared events arraylist to display them.

Calendar: The 'Calendar' class is passed the global IDGenerator and its generated calendarID by the User class. It contains an arraylist, 'months,' of associated Months, and a 'monthCount' to represent however many and what kinds of months a specific Calendar can have. It has a 'screen' member variable of type Screen that deals with any and all display functionalities.

The 'addEvent(EventInfo e, bool repeat)' takes an instance of EventInfo and whether or not the event repeats, checks if the EventInfo's Month(s) (in the 'startDate' & 'endDate') correspond to Calendar's own 'months', and creates a brand new Event to its 'events' dictionary. The 'events' dictionary has the event ID as a key and the calendar as a value. Calendar's 'removeEvent(int eventID)' uses the eventID, as the key, to find the ('eventID', Event) pair in the 'events' dictionary and removes it. The 'updateEvent' method identifies the Event in the 'events' dictionary and assigns it a new EventInfo. The 'getEvent' method returns the Event using the eventID, by utilizing the dictionary.

The 'addUser' method allows for a User to be added to the 'users' Dict, with the user ID as the key and the User as the value. The method allows this to happen only if the 'publicVisibility' member variable is true. The 'removeUser' method, simply matches the userID parameter to the 'users' dictionary key values, and deletes its corresponding User.

The 'categories' dictionary keeps a string category name as its key and an ArrayList of eventIDs as its value. 'addCategory' creates a category and adds it to the dictionary accordingly. 'getCategory', mostly used by the Screen class, allows the ArrayList of eventIDs to be returned. 'addEventToCategory' uses its parameter details (eventID and category string) to add an event to the 'categories' dictionary.

GregorianCalendar: The 'GregorianCalendar' class inherits all of Calendar's functionalities but simply sets a constant arraylist of Months, associated with the appropriate Gregorian Calendar

months and days, as well as a constant ‘monthCount’ of 12. This class helps easier create a GregorianCalendar specifically instead of keeping track of which Calendars have the same kind of Months & monthCount.

Month: The ‘Month’ class simply keeps track of its string name, and its integer number of days. It has a single method ‘isWithinDayMax’ which returns a boolean if any particular day fits within this Month’s day count.

Screen: ‘Screen’ handles all visual elements of a Calendar. It adjusts the time zone of time elements in the Calendar through the ‘adjustTimeZone’ method, a well as changes the visualization mode, whether in day view, month view or year view, through the changeVisualizationMode method. The ‘showSubset’ method takes a category string and utilizes Screen’s Calendar member variable to retrieve all events associated to by the specific ‘category string,’ using Calendar’s categories dictionary.

Event: The ‘Event’ class is given its eventID member variable by the Calendar class, and its eventInfo. It has a list of Users associated to this Event, these include users whom this Event has been shared to and any users who have this Event through the parent Calendar. (Users can be added through its corresponding method, ‘addUser’). The Event can add a Timer, saved in its ‘myTimer’ member variable, and can return event details with its getEventDetails method.

EventInfo: The ‘EventInfo’ class stores all relevant event details as member variables for easy passing and reference.

Timer: The ‘Timer’ class is used and created by ‘Event;’ the ‘display’ method handles all of the Timer countdown functionality using its endDate and endTime method variables.

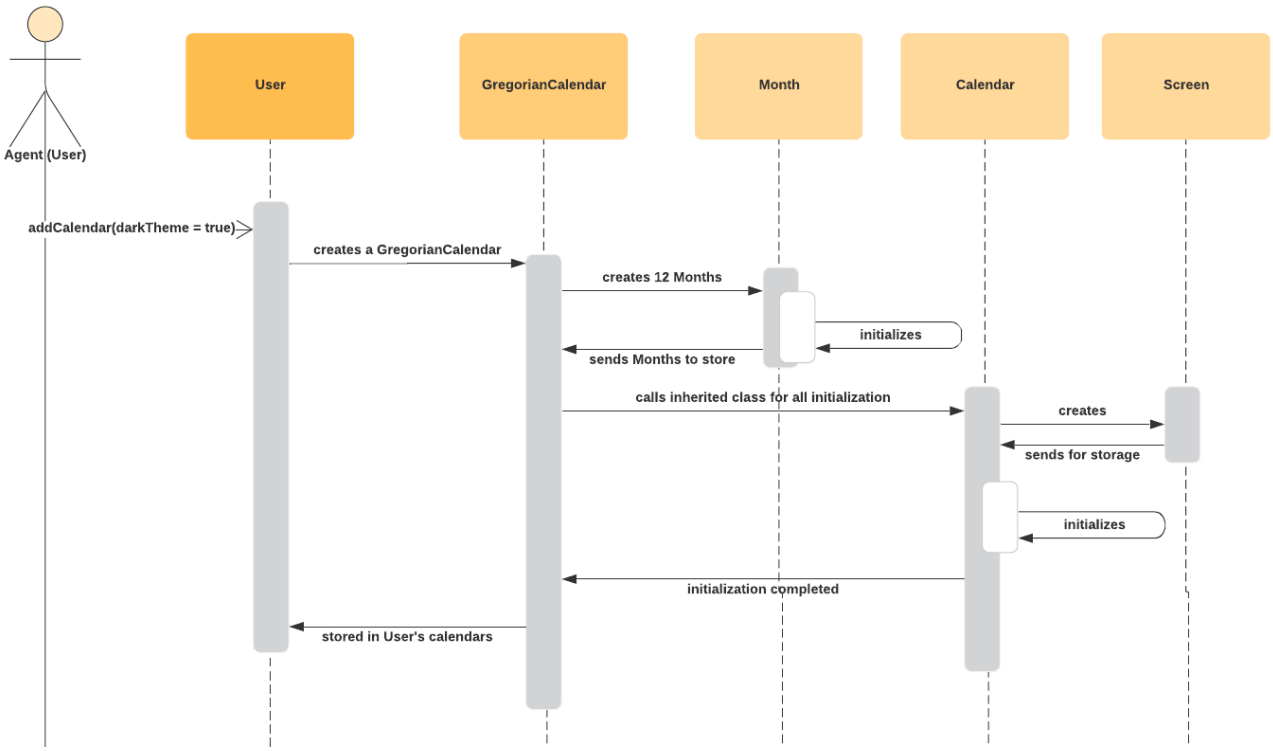
Accounting for Future Changes:

Other types of calendars, Julian calendar, or TamrielCalendar can easily be inherited from the Calendar class and made into another class, just as the GregorianCalendar is currently. They would make use of the flexibility of the Months list in Calendar and Month class, which allows for any amount of days and any string title. A different kind of event or different kind of user would simply need to inherit from the Event and User class.

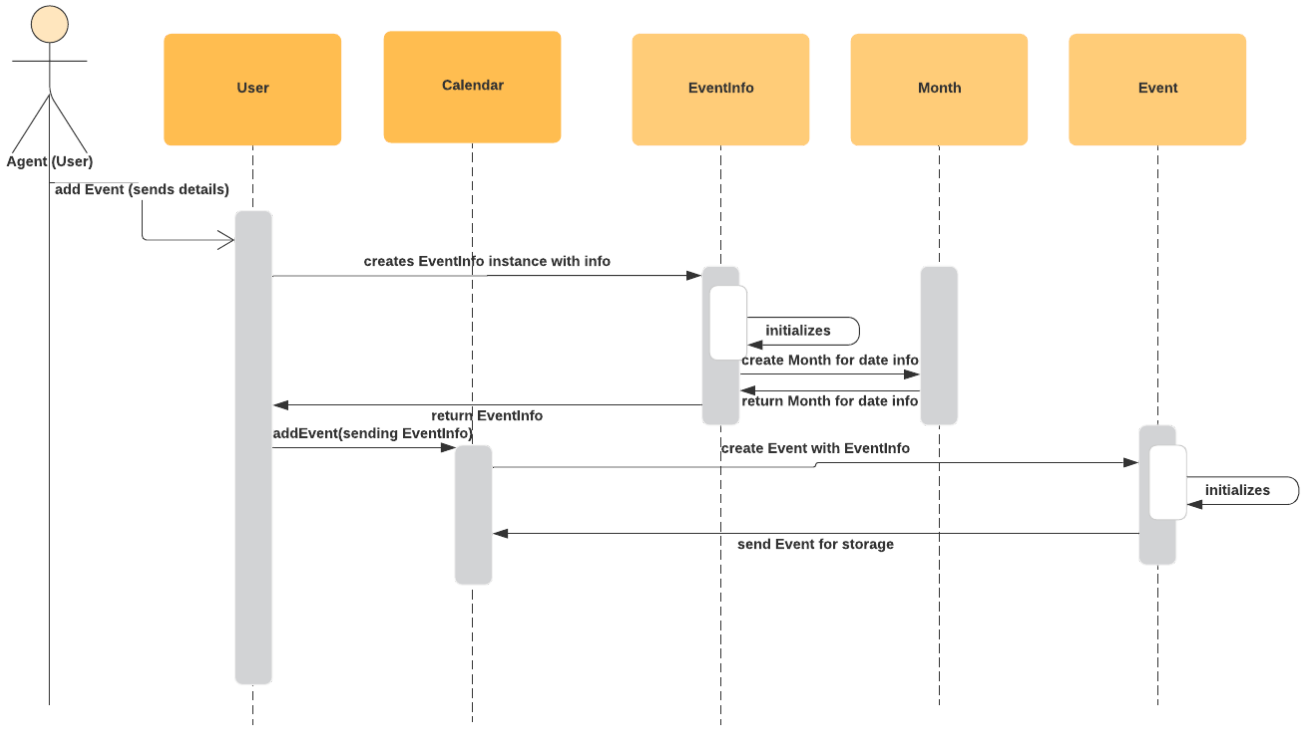
Allowing for undoing deletion or editing of calendar or event would be enacted by a simple pastCalendars, pastEvents, pastEventInfos member variables in the User, Calendar, and Event classes respectfully.

Adding a GUI would simply utilize these classes and add a new GUI class that interacts with Main. Images to represent calendars or events could be member variables in Calendar or Screen. To-do lists could be a separate class, but a member variable in Calendar, as well as Notifications. Notes/descriptions is an easy string member variable that can be added to EventInfo or Calendar. A different language could be set in either Main or the Screen class.

Sequence Diagrams



The above sequence diagram represents the Agent(User)'s creating a Calendar process. It utilizes the User class to create a GregorianCalendar, which utilizes Month to ensure the respective kind of Month for the Gregorian Calendar exist. Then, the Gregorian Calendar calls the inherited class Calendar to initialize all member variables, (one such is initializing the Screen class).



The agent (user) decides to add an Event with its appropriate details to a particular Calendar, so the User class first, creates an EventInfo class to store its details (which then uses Month too store the date info), then, sends the EventInfo instance to the Calendar's addEvent method. The calendar creates the Event, the Event initializes, and it is sent to the Calendar for storage.