Fall 2016 Programming Assignment 4: Simple GUI Calendar

Instructor: Dr. Kim

Soft Copy Due: Saturday, November 19 11:59pm.

Hard Copy Due: Monday, November 21 in class.

In this assignment, you will design and implement a simple version of GUI calendar application. To represent and manipulate a calendar (in terms of day, month, year, time, etc), use the GregorianCalendar class from the library that provides a standard calendar system. (The API of GregorianCalendar comes with useful examples.)
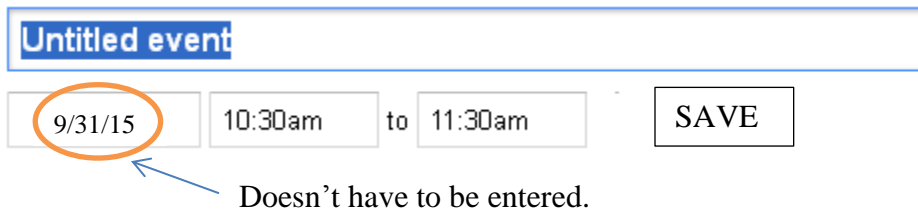
Minimum Functional Requirements

1. A sample initial screen of the calendar is shown here:
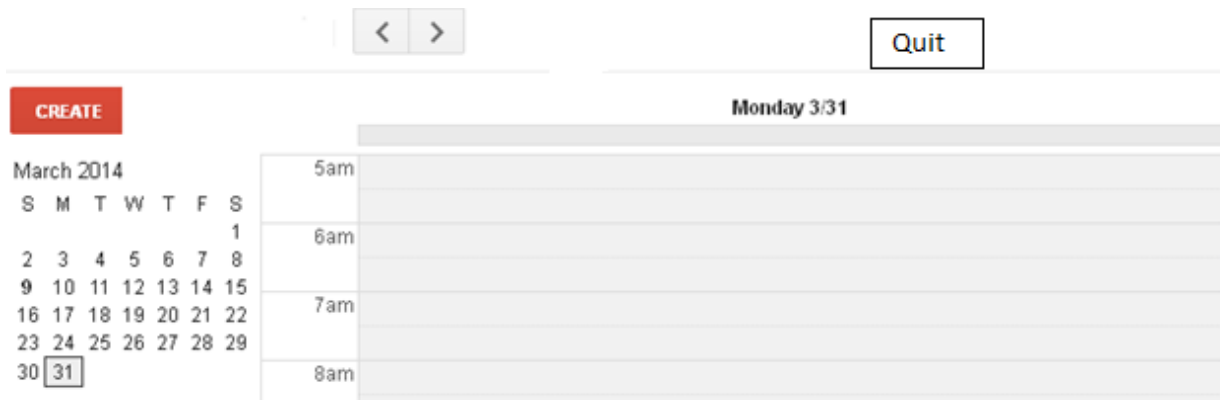
2. Follow the MVC pattern to design the calendar application.
3. If events.txt exists, starting program will load events from the events.txt to the Calendar. (You may use Deserialization for this purpose.)
4. The initial screen presents the current month view and the current day view. Each day on the **month** view is clickable. When a user clicked on a day on the month view, the day view (at the right side) is changed to represent the day the user clicked on.
5. The calendar comes with previous and next buttons, ,to move the calendar backward and forward, respectively. Each click on the button will change the calendar one

day at a time. For example, when the user clicked on button from the calender shown above, the rectangle surronding 31 will now highlight 30, and the day view at the right side will be changed to the day view of March 30. If the user clicked on button from the given calendar, the month view (at left side) will present the view of April, the rectangle will highlight the day 1, and the day view will change to the view of April 1.

6. A user should be able to move the calendar back and forth as far as the GregorianCalendar can go.

7. The calendar comes with CREATE button through which a user can enter events to be scheduled in the calendar. To schedule an event on a particular day, a user first clicks on the day of the monthe view, and then clicks on the CREATE button. The CREATE button pops up input box(es) through which the user can enter an event. The following is a sample screen shot of input boxes. Your application doesn't have to show exactly the same input boxes. You may simplify the way to get an event from the user as long as the calendar can take an event name, starting time and ending time. (Since the user clicked on the day before choosing the CREATE option, the user does not have to re-enter the day information.)

Doesn't have to be entered.

We assume a user always correctly enters an event in the format shown in the above sample. After the user saves the event, the event is stored in the calendar. When the calendar present a day view representing the day with event(s), All scheduled event should be presented on the day view.

8. The calendar should be able to check a time conflict. When a user enters an event schedule that conflicts with an existing event, the calendar generates an error message, asking the user to enter an event without any time conflict.

9. The Quit button terminates the Calendar program and saves the events in a file called events.txt. (You may use Serialization for this purpose.)

Deliverable

1. Soft copy: hw4.zip containing all java files.

   Name a class that contains the main method "SimpleCalendar" and save it in a file called "SimpleCalendar.java".

2. Hard copy:

   - Write a report that contains the following

   1. Brief description about the Model, View, and Controller of your application.

   2. Take the following screen shots of the calendar

     Screen 1: After starting the application (initial screen)

     Screen 2: After clicking on the previous button on Screen 1

     Screen 3: After clicking on the next button 20 times from Screen 2

     Screen 4: After clicking on a particular day on Screen 3 and creating an event on that day.

     Screen 5: After clicking on the same day you choose in the previous step and creating an event with a time conflict.

     Screen 6: After entering an event on the same day you choose in the previous step and creating an event without a time conflict.