Assignment Purpose

For this assignment, you will apply what you have learned to build an application to maintain a personal appointment calendar. You can work in your groups (2 members). If your partner is not present in class, each will need to complete the assignment on your own by the due date/time.

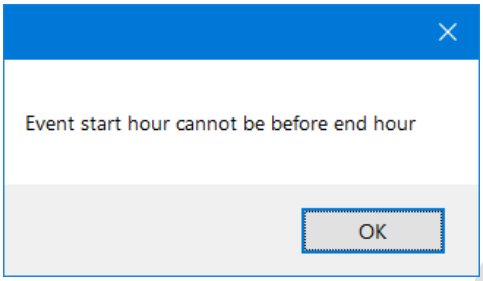
Mandatory Instructions

Step 1: Create an Event class

Information about each event on the calendar will be held in an object of type Event. The constructor will accept integer values for month, day, year, starting hour, starting minute, ending hour, ending minute, and event title. We assume that the start time and the stop time are both on the same day. Save the start time and the stop time as private DateTime values, and provide a public read-only property for each. Save the event title as a private string value, and provide a read-only property.

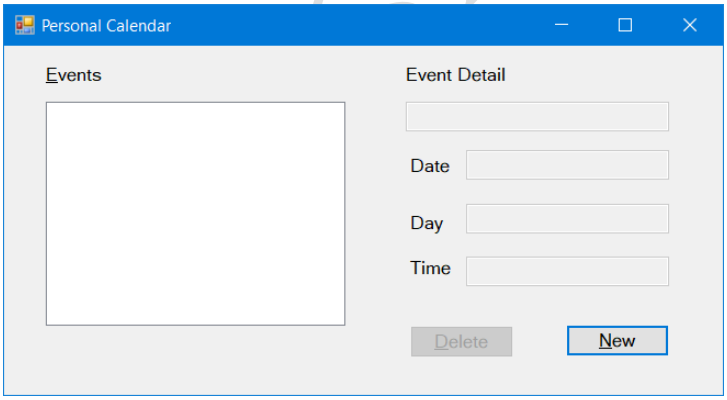
Override all three of the standard methods inherited from the Object class. For the ToString method, the string returned should be in the following format: "yyyy-mm-dd hh:mm" (for example, "2017-03-05 15:30"). Decide for yourself a reasonable interpretation of when two events are equal and what to return for GetHashCode().

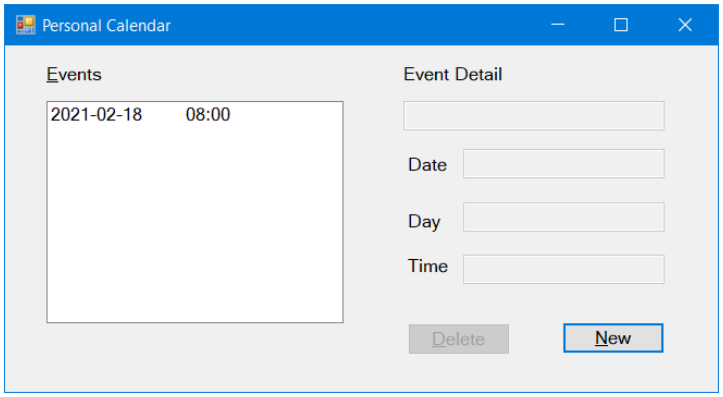
The start time must proceed with the stop time, and the event title cannot be empty. If either of these constraints is not satisfied, an exception should be thrown with a clear message indicating the problem. A message box should be shown to the user. Make your message box contain an appropriate title and icon.



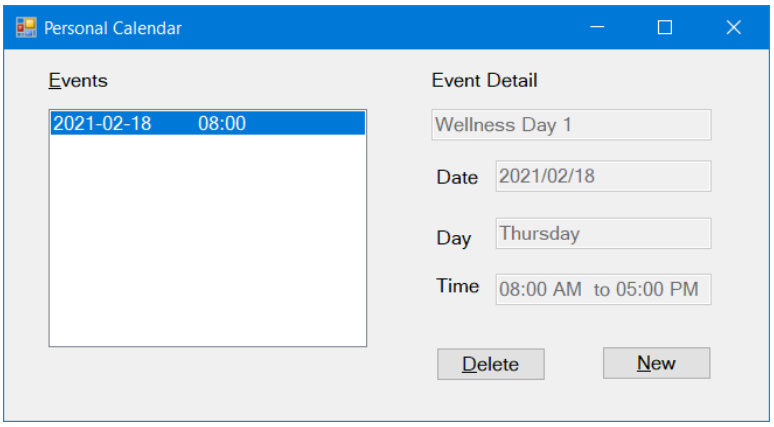
Step 2: Design the main form

Your form should look like the below example on the left initially and on the right after an event is added.





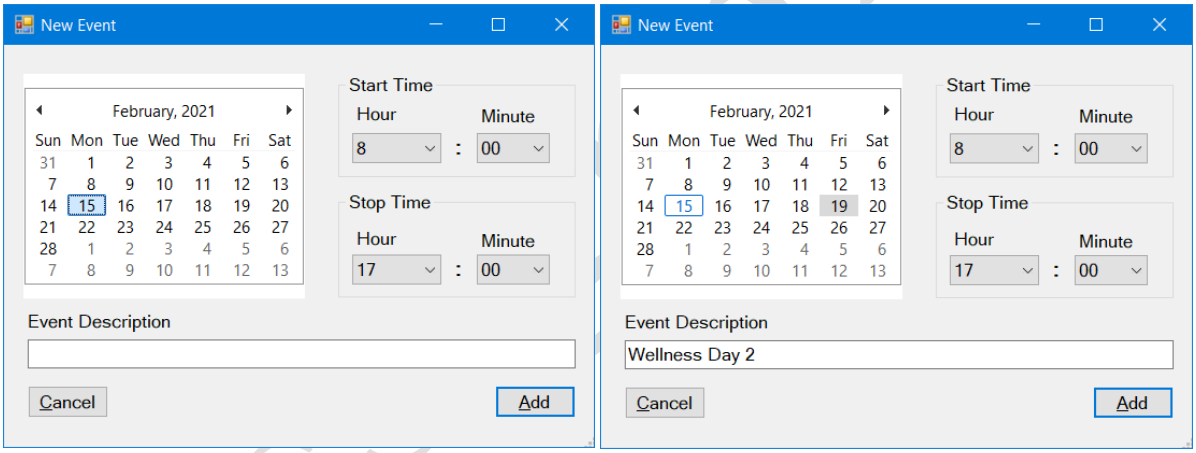
When the user selects an event shown in the Events Listbox, details should then be displayed in the event details text boxes (title, date, day, time). See example below.



I suggest that you set the Font property for the form to be 10-point **before** adding any controls. That font setting then becomes the default for controls on the form. Be sure to set the Sorted property of the list box equal to true. The Delete button should be initially disabled and enabled when an event is selected from in the list box. When the Delete button is clicked, the selected event should be removed from the list box, all text boxes should be cleared, and the Delete button should be disabled again.

Each entry in the list box should be an Event object. The values from that Event object should be used to fill the text boxes in the format shown. (This is pretty easy if you take advantage of the DateTime structure features, so check out the documentation before getting started.)

Step 3: Design the event form



Add a new form, called EventForm, to your project. It can also be called Form2.

This form is displayed when the user wants to add an event to the calendar. Set the AcceptButton property for the form to be the Add button, and set the CancelButton property of the form to be your Cancel button.

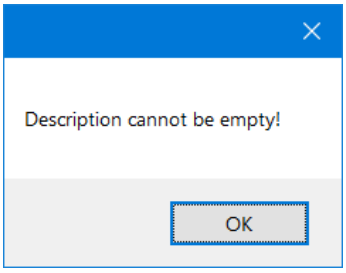
For this form, you will need to create a private static variable of the type list box that will refer to the main form's list box. You will also want a static property that sets/gets that value. Now go back to the main form (Form1) and add a line in the Load event that saves a reference to the list box in the EventForm class.

For each of the four combo controls on this form, set the type to be DropDownList. This will prevent the user from trying to type in an invalid hour or minute value. The valid hour values are integers from 0 to 23. The minute values are 00, 15, 30, 45. We will use a 24-hour clock. Be sure to set the start hour to 8 am and the end hour to 5 pm initially, i.e., business day hours.

The Load method for this form should restrict the calendar so that the earliest valid date is the current date (i.e., the date the program is being run), and the latest possible date is December 31 of the current year. It should also set a default value for each of the combo boxes.

Add a private Event variable to the form to represent the newly created event, and define a read-only property for this value. If the user clicks Cancel on this form, the event variable should be null, and no event should be added to the list box.

When the Add button is clicked, an Event object is created using the data from the form. Any exception is caught and handled correctly. For example, no event title/description provided…



Also, go through the current events in the list box to ensure that the new event doesn't have a time conflict with some event that already exists. For this, you might use a statement like the following:

```
foreach (Event otherEvent in mEventList.Items)
```

When you have fully debugged your code, upload a compressed solution folder to Canvas. Each group member should upload the solution and provide their grade sheet for this project.

Code Documentation

Each source file should contain a documentation header with a description, author name, and class information (CS3160, Fall/Spring 20xx). Each function should have a documentation header with, minimally, function name, description, parameters, return value. Use proper program style at all times, which means indentation, alignment, and whitespace. Utilize self-documenting code which means use mnemonic, i.e., meaningful variable/function/namespace/class names, etc.

What to turn in?

Submit a compressed (.zip) solution folder via Canvas.
Make sure that all necessary files to compile/link/execute your projects are provided in your solution folder.
Here are some files that may be required:

1. All required C# source files (.cs)
2. makefile (if the solution requires it)
3. The entire Visual Studio 2019 solution folder (if the solution requires it)