

# CS 205 - Programming for the Sciences

## Spring 2008 - Programming Assignment 5

20 points

**Out: March 27, 2008**

**Due: April 3, 2008**

Consider an application like an appointment book. One of the types of objects that such an application would need is a Date object. In this assignment, you will write a Date class that models a calendar date. (C# has a built-in DateTime class that models both date and time together in one object, but we'll make up our own Date class that just models a date.) The Date class consists of three private integer attributes representing the day, month, and year of a Date and the following public methods:

- Properties **Day**, **Month**, and **Year** that return the day, month, and year, respectively, of a Date. Since Dates, like numbers, are immutable values, the property is to be read-only and have only a get method.
- Empty constructor that creates a default date of January 1, 2008.
- Explicit-value constructor that receives three integers, **d**, **m**, and **y** (in this order), that are the day, month, and year values of the Date to be created. A valid Date must meet the following criteria (also called the class **invariant**):
  - year is greater than 0
  - month is between 1 and 12, where 1 represents January, 2 represents February, etc., to 12 represents December
  - day is in the correct range for the month represented: January, March, May, July, August, October, and December have 31 days; April, June, September, and November have 30 days; February has 28 days unless y is a leap year, then it has 29 days. A leap year is divisible by 4, but not by 100, unless it is also divisible by 400. E.g., 2008 is a leap year, 2100 is not a leap year, 2000 was a leap year.

If any of year, month, or day do not meet these criteria, the constructor should **throw** an exception with an appropriate message.

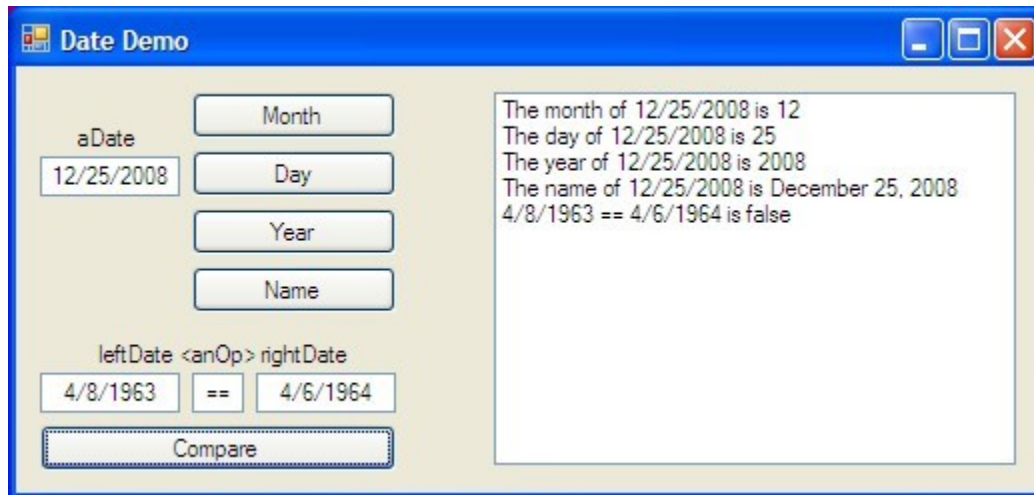
- **Static** method **Parse** that returns the Date equivalent of a received string. Since there are two slashes in the string representation of a Date, we need to use the version of the string **IndexOf** method that has the following syntax:

```
targetIndex = s.IndexOf(targetChar, startIndex, count);
```

where **s** is a string, **targetChar** is the character to search for, **startIndex** is the index to start the search at, and **count** is the number of positions to search. As before, **IndexOf** returns -1 if the target character is not found. If the received string is not in the correct format, an exception should be thrown with an appropriate message.

- **Overridden** method **ToString** that returns the string equivalent of the Date in the form "**m/d/y**".
- **Name** method that returns a string equivalent of the Date in the form "name-of-month day, year". E.g., this method would return "**January 1, 2008**" for 1/1/2008.
- **Static operator** methods for the comparison operators **==**, **!=**, **<**, and **>**. Given two Dates **d1** and **d2**, **d1 == d2** is true, if their day, month, and year are the same; **d1 != d2** is true if any of their day, month, or year is not the same; **d1 < d2** is true if **d1** is before **d2**; **d1 > d2** if **d1** is after **d2**.

For this assignment, you are given a compressed solution folder DateDemo.zip that contains a project solution with the following GUI that will allow you to test a Date class meeting the specification above. A sample of the results that should be displayed by the GUI is also shown.



### Assignment

Download the compressed solution folder DateDemo.zip from the course webpage.

Complete the Date Demo program in the same manner as was done for the Rational number demo in-class exercise. To get you started, declaration of the attributes and the implementations of the properties and the empty constructor have been provided in **Date.cs**. You need to complete the rest of the Date class as specified above. (Ignore the warnings about Equals and GetHashCode.) The handlers for the GUI buttons are to do the following:

- Month, Day, and Year buttons - display the Month, Day, or Year property, respectively, of the Date given in the aDate textbox.
- Name button - display the result of calling the Name method of the Date given in the aDate textbox.
- Compare button - display the result of comparing the Dates given in the leftDate and rightDate textboxes using the operator in the anOp textbox. The handler should throw an exception with an appropriate message if the operator is not valid. (Parse and the explicit-value constructor will take care of malformed input.)

### What to submit

For full credit, a compressed (zipped) solution folder containing a C# project must be submitted as an attachment to an email to Dr. Hwang ([hwang@evansville.edu](mailto:hwang@evansville.edu)) no later than 4:30pm on the due date.

To create a compressed solution folder find the solution folder via Windows Explorer. Right-click on the solution folder, select Send To, then select Compressed (zipped) folder. This will create a compressed folder of the same name as the solution folder with extension .zip and an icon of a folder with a zipper.