CSE 142, Autumn 2007
Programming Assignment #4: Birthdays (20 points)
Due: Tuesday, October 23, 2007, 4:00 PM

Program Description:

This assignment will give you practice with `if/else` statements, interactive programs with `Scanner`, cumulative sum, and `return` statements. Turn in a Java file named `Birthdays.java`. The following are two runs of your program and their expected output (user input is bold and underlined):

```
Today's date (month and day)? 10 17

Person #1 of 2:
What month and day were you born? 11 23
11/23/07 falls on day #327 of 365.
Your next birthday is in 37 day(s).

Person #2 of 2:
What month and day were you born? 10 17
10/17/07 falls on day #290 of 365.
Happy birthday!

Person #2's birthday is sooner.
```

```
Today's date (month and day)? 12 25

Person #1 of 2:
What month and day were you born? 1 10
1/10/07 falls on day #10 of 365.
Your next birthday is in 16 day(s).

Person #2 of 2:
What month and day were you born? 5 1
5/1/07 falls on day #121 of 365.
Your next birthday is in 128 day(s).

Person #1's birthday is sooner.
```

Your program asks for two users' birthdays and prints information about them. The program initially prompts for the current date (month and day), then prompts for the birthday month and day of the two users. After each birthday is read, the program prints the absolute day of the year on which that birthday falls, and the number of days until the user's next birthday. Lastly the program shows which user's birthday comes sooner in the future.

Since this is an interactive program, it behaves different ways when it is given different input values. Please carefully examine all of the expected output files on the course web site as well as conducting your own testing. **You may assume that all user input is valid and that no user is born on February 29th.**

This program involves a lot of date comparisons. Let's recall some basic facts about calendar dates. There are normally 365 days per year, divided among the 12 months as follows:

Month	1 Jan	2 Feb	3 Mar	4 Apr	5 May	6 Jun	7 Jul	8 Aug	9 Sep	10 Oct	11 Nov	12 Dec
Days	31	28 *	31	30	31	30	31	31	30	31	30	31

* One factor that makes this program more complicated is that 2008 is a leap year. Leap years have an extra 366th day, which occurs as February 29th.

One of the major tasks of this program is computing each user's absolute day of the year on which his or her birthday falls in the year 2007. For example, January 1st is absolute day #1. January 2nd is absolute day #2. January 31st is #31. February 1st is #32. February 28th is #59, and March 1st is #60; and so on, up to December 31st, which is #365. You do not need to worry about leap years here, since we are always considering the year 2007 for this part of the program.

Another major task in this program is computing how many days remain until each user's next birthday. There are several important cases to consider. If the user's birthday falls after today's date, it is within the year 2007. However, if it falls before today's date, the next birthday occurs in the year 2008, so you'll need to think of a way to count the days that "wrap around" between today and the end of 2007, and then the start of 2008 to the user's birthday in 2008. Making it more complicated is that if the user's birthday is after February 28th (but before today's date), you'll need to account for the extra leap year day.

Stylistic Guidelines:

To receive full credit on this assignment, you are required to have at least 3 non-trivial static methods in your program besides `main`. One of these must be the particular method described below.

Method to get the absolute day of the year, for a given month/day:

This method should accept parameters representing a month and day and should return the absolute day of the year 2007 that is represented by those parameters. For example, calling this method with the parameter values of 2 and 13 (representing February 13th) should return 44, and calling it with parameter values of 9 and 19 (representing September 19th) should return 262. This method should not produce any console output, though the result it returns can be printed by code elsewhere in your program.

In grading, we require the preceding method, and at least 3 total non-trivial methods besides `main`. Continue to use static methods for structure and to reduce redundancy, utilizing parameters and return values appropriately. Each method should perform a coherent task and should not do too large a share of the overall work by itself.

Much of your code will involve conditional execution with `if` and `if/else` statements. Part of your grade will come from using these statements appropriately. You may want to review section 4.2 of the textbook about nested `if/else` statements and section 4.3 about factoring `if/else` code.

Unlike in past assignments, you may have `println` statements in your `main` method on this program. In fact, you'll probably want to do this for certain parts of your output that don't fit well into other methods. The `main` method should still be concise and should represent a reasonable summary of the overall program's execution.

For this assignment you are limited to the language features in Chapters 1 through 4, in addition to the logical operators described in book section 5.2. Give meaningful names to methods and variables, and use proper indentation and whitespace in your code. Follow Java's naming standards as specified in Chapter 1. Localize variables when possible; declare them in the smallest scope needed. Include meaningful comment headers at the top of your program and at the start of each method.

How to Get Started, and Hints:

As usual, you should write your code incrementally, repeatedly making small improvements. **We strongly suggest that you completely ignore leap years until the very end of your work.** It is possible to get the entire program running without worrying about this aspect, except that some birthday estimations will be off by 1 day. Once the rest of your program works, you can make small changes to account for this case. The leap year only affects certain parts of the program and only in certain particular cases.

When you write the absolute day method described above, test it by calling it with several fixed values. To compute the absolute day value, it may help you to review the idea of a cumulative sum as described in section 4.1 of the textbook. You will also need code to evaluate the number of days in each particular month. (You might put this code into a method that accepts a month parameter and returns the number of days in that month.)

Computing the number of days until the user's birthday seems very difficult at first. If you have other parts of the program finished, you can base this computation on those parts. Don't forget to take advantage of the methods you have already written, and that methods can call other methods to help perform their overall task.

For reference, our solution to this assignment occupies 80 lines including comments and has 4 methods other than `main`, though you do not need to match these amounts exactly to receive full credit.

Don't forget to place the following statement at the top of your Java file:

```
import java.util.*;   // so that I can use Scanner
```