CSE 142, Summer 2010
Programming Assignment #4: Birthday (20 points)
Due: Tuesday, July 20, 2010, 11:30 PM

Program Description:

This interactive program focuses on `if/else` statements, return, `Scanner`, and computing a cumulative sum. Turn in a Java file named `Birthday.java`.

The program prompts for today's date and the user's birthday. Each prompt lists the range of values from which to choose. Notice that the range of days printed is based upon the number of days in the month the user typed.

The program prints the "absolute day of the year" for today and the birthday. This is a date's place within the year from 1 to 365. January 1st is absolute day #1 and December 31st is #365. Lastly the program prints the number of days until the user's next birthday. Different messages appear if the birthday is today or tomorrow.

The following are four runs of your program and their expected output (user input is bold and underlined):

```
This program tells you how many days
it will be until your next birthday.

Please enter today's date:
What is the month (1-12)? 7
What is the day   (1-31)? 24
7/24 is day #205 of 365.

Please enter your birthday:
What is the month (1-12)? 11
What is the day   (1-30)? 6
11/6 is day #310 of 365.

Your next birthday is in 105 days.

<< your birthday fact >>
```

```
This program tells you how many days
it will be until your next birthday.

Please enter today's date:
What is the month (1-12)? 9
What is the day   (1-30)? 22
9/22 is day #265 of 365.

Please enter your birthday:
What is the month (1-12)? 3
What is the day   (1-31)? 17
3/17 is day #76 of 365.

Your next birthday is in 176 days.

<< your birthday fact >>
```

```
This program tells you how many days
it will be until your next birthday.

Please enter today's date:
What is the month (1-12)? 2
What is the day   (1-28)? 14
2/14 is day #45 of 365.

Please enter your birthday:
What is the month (1-12)? 2
What is the day   (1-28)? 15
2/15 is day #46 of 365.

Wow, your birthday is tomorrow!

<< your birthday fact >>
```

```
This program tells you how many days
it will be until your next birthday.

Please enter today's date:
What is the month (1-12)? 12
What is the day   (1-31)? 25
12/25 is day #359 of 365.

Please enter your birthday:
What is the month (1-12)? 12
What is the day   (1-31)? 25
12/25 is day #359 of 365.

Happy birthday!

<< your birthday fact >>
```

At the end (regardless of what input the user typed), print a fun fact of your choice about your own birthday. Search the web for fun facts about your date of birth. For example, if yours is July 11, you could print:

```
Did you know July 11, 2010 Spain won the 2010 FIFA World Cup?
It was the first time Spain won the World Cup and became the eighth team to ever win a
World Cup.
```

Since this is an interactive program, it behaves differently when given different input. The examples above do not show all possible cases. Please **look at all example outputs on the web site** and do your own testing. We are picky about output matching *exactly*, including spacing; use the web Output Comparison Tool to be sure.

You may assume that all user input is valid; that when prompted for a date, the user will enter an integer in a valid range. You may also assume that your program is not run in a leap year, so February always has 28 days.

Implementation Details:

The 365 days of the year are divided into twelve 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

The Section 4 handout contains a `daysInMonth` method that, when passed a month parameter, returns the number of days in that month. Use this method to help you write your program and to avoid redundancy.

Absolute day of the year:

One major task in this program is computing the **absolute day of the year** on which each given date falls. Jan 1 is absolute day #1. Jan 2 is #2. Jan 31 is #31. Feb 1 is #32. And so on, up to Dec 31, which is #365.

Date (month/day)	1/1	1/2	1/3	...	1/31	2/1	2/2	...	2/28	3/1	3/2	...	12/30	12/31
Absolute day	1	2	3	...	31	32	33	...	59	60	61	...	364	365

You must compute each absolute day using a cumulative sum loop as shown in textbook section 4.2. A given date's absolute day is the sum of all prior months, plus the days within the current month. For example:

	Jan	*Feb*	*Mar*	*Apr*	*May*	*Jun*	*Jul*	*total*
• *absolute day for July 24* =	31	+ 28	+ 31	+ 30	+ 31	+ 30	+ 24	= **205**

Part of your grade will come from decomposing your solution into a coherent set of methods. You could write a parameterized method that computes the absolute day for a given date. We suggest that you use the `daysInMonth` method from section to help compute your cumulative sum.

Days until next birthday:

Another task is computing how many **days until a person's next birthday**. There are two cases to consider:

1) When the birthday falls after today's date: This simpler case means that the next birthday is within the current year. Look at the first example output on the previous page. "Today" is 7/24, and the birthday is 11/6.

```
1       2       3       4       5       6       7       8       9       10      11      12
                                                Today                   Birthday
                                                7/24                    11/23
                                                #205                    #310

                                                    ---------------------->
                                                105 days to birthday
```

2) When the birthday falls before today's date: In this case you need to "wrap around," counting the days from today to the end of the year, and then from the start of next year to the user's birthday.

In the second example output, "today" is 9/22 and the user's birthday is 3/17:

```
    1       2       3       4       5       6       7       8       9       10      11      12
                    Birthday                                        Today
                    3/17                                            9/22
                    #76                                             #265
          76                                                                100
... ------------->                                                  ------------------> ...
                                                                    176 days to birthday
```

It may help you to use the number 365, the number of total days in a year, in your wrapping computations.

Style Guidelines:

Many of the style points on this assignment are related to how well you procedurally decompose the problem into a set of well-chosen static methods. To receive full credit, you must have at least **4 non-trivial methods** in your program besides `main`. Use methods for structure and to reduce redundancy. You should also utilize parameters and return values appropriately to pass information throughout your code.

See the textbook (pgs. 283 – 288) for information about *"procedural design heuristics"* to help understand what constitutes a good method. Each method should perform a single clear and coherent task. No one method should do too large a share of the overall work. As a very rough estimate, a method whose body (excluding the header and closing brace) has more than 15-20 lines is likely too large and may lose points.

In this assignment it is okay to have a few `println` statements in the `main` method, though you should limit this as much as possible. The contents of `main` should be a concise summary of the overall program; the majority of your program's behavior should come from its other methods. You should avoid "chaining" long sequences of method calls together without returning to `main`, as described in the Heuristics reading.

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 4.1 of the textbook about nested `if/else` statements and section 4.4 about methods with conditional execution.

For this assignment you are limited to language features in Chapters 1 through 4, in addition to logical operators such as `&&` and `||` described in book section 5.3. You are especially forbidden from using Java's built-in date libraries such as `Date` or `GregorianCalendar`.

Give meaningful names to methods and variables, and use proper indentation and whitespace. Avoid long lines over 100 characters in length. 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.

Don't forget to place the following declaration at the top of your program, to be able to read user input:

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

Development Strategy and Hints:

On this program, many students encounter **"cannot find symbol"** compiler errors related to scope, parameters, and return values. Common mistakes include forgetting to pass or return a needed value, forgetting to store a returned value into an appropriate variable, and referring to a variable by the wrong name.

You should write your code incrementally, repeatedly making small improvements and testing it. You might want to start by writing code to compute the absolute day of the year as described previously. Test this code by running it with several fixed values and debug it until it works, then move on to the rest of the program.

Computing the number of days until the user's birthday can be challenging. But 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. Methods can call other methods to help perform their overall task.

Caution: We are very picky about output matching exactly. This includes identical spacing, such as the extra spaces after the phrase, "What is the day." Many students lose points for minor output formatting errors. Please run the web Output Comparison Tool on several test cases to make sure it matches.

For reference, our solution to this assignment (excluding the "birthday fact" code) occupies 90 lines including comments (53 "substantive" lines according to our Indenter tool), and has 6 methods other than `main`, though you do not need to match these amounts exactly.