CS 210, Fall 2012
Programming Assignment #3: Birthdays (50 points)
Due on or before: 11:59PM Sunday, November 4, 2012

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`.

This program involves date comparisons in 2012, which is a "leap year." Leap years have an extra 366th day, which occurs as February 29th. The 366 days are 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	29	31	30	31	30	31	31	30	31	30	31

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

```
This program compares two birthdays
and displays which one is sooner.
Today is 1/29/2012, day #29 of the year.

Person 1:
What month and day were you born? 10 17
10/17/2012 falls on day #291 of 366.
Your next birthday is in 262 day(s).
That is 71.6 percent of a year away.

Person 2:
What month and day were you born? 11 23
11/23/2012 falls on day #328 of 366.
Your next birthday is in 299 day(s).
That is 81.7 percent of a year away.

Person 1's birthday is sooner.

<< your birthday fact here >>
```

```
This program compares two birthdays
and displays which one is sooner.
Today is 2/14/2012, day #45 of the year.

Person 1:
What month and day were you born? 1 10
1/10/2012 falls on day #10 of 366.
Your next birthday is in 331 day(s).
That is 90.4 percent of a year away.

Person 2:
What month and day were you born? 2 28
2/28/2012 falls on day #59 of 366.
Your next birthday is in 14 day(s).
That is 3.8 percent of a year away.

Person 2's birthday is sooner.

<< your birthday fact here >>
```

```
This program compares two birthdays
and displays which one is sooner.
Today is 1/1/2012, day #1 of the year.

Person 1:
What month and day were you born? 1 1
1/1/2012 falls on day #1 of 366.
Happy birthday!

Person 2:
What month and day were you born? 12 31
12/31/2012 falls on day #366 of 366.
Your next birthday is in 365 day(s).
That is 99.7 percent of a year away.

Person 1's birthday is sooner.

<< your birthday fact here >>
```

```
This program compares two birthdays
and displays which one is sooner.
Today is 2/28/2012, day #59 of the year.

Person 1:
What month and day were you born? 2 29
2/29/2012 falls on day #60 of 366.
Your next birthday is in 1 day(s).
That is 0.3 percent of a year away.

Person 2:
What month and day were you born? 2 29
2/29/2012 falls on day #60 of 366.
Your next birthday is in 1 day(s).
That is 0.3 percent of a year away.

Wow, you share the same birthday!

<< your birthday fact here >>
```

Your program asks for two users' birthdays and prints information about them. The program prompts for the birthday month and day of the two users. For both birthdays, the program prints the absolute day of the year on which that birthday falls, the number of days until the user's next birthday, and the percentage of a year (percentage of 366 days) away. Next the program shows which user's birthday comes sooner in the future. If it is a user's birthday today, or if the two birthdays are the same, different messages are printed.

Lastly, the program prints a fun fact about your own birthday. Try searching Wikipedia and Google for interesting facts about your date of birth. For example, if yours is Sep. 19, you could print the following fact:

```
Did ye know?  9/19 be International Talk like a Pirate Day.
Arr, me mateys, arr!
```

Since this is an interactive program, it behaves differently when given different input. The examples above may not show all possible cases. Please **think and evaluate all possible outputs** and do your own testing.

You may assume that all user input is valid, and that the program is being run between 1/1/2012 and 2/28/2012.

Absolute day of the year:

One major task in this program is computing the **absolute day of the year** on which each user's birthday falls in 2012. Jan 1 is day #1. Jan 2 is #2. Jan 31 is #31. Feb 1 is #32. And so on, up to Dec 31, which is #366.

Date (month/day)	1/1	1/2	1/3	...	1/31	2/1	2/2	...	2/28	2/29	3/1	3/2	...	12/30	12/31
Absolute day	1	2	3	...	31	32	33	...	59	60	61	61	...	365	366

For reference, the following are the absolute days for the first day of each month:

Date (month/day)	1/1	2/1	3/1	4/1	5/1	6/1	7/1	8/1	9/1	10/1	11/1	12/1
Absolute day	1	32	61	92	122	153	183	214	245	275	306	336

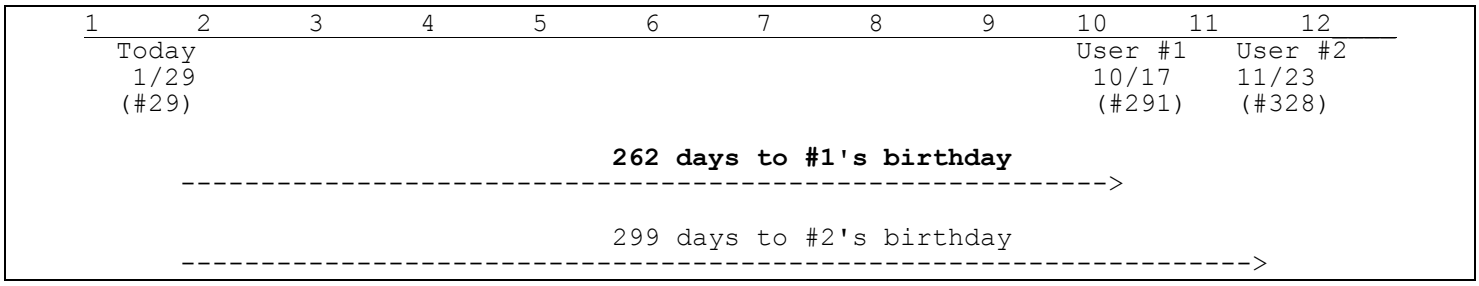
You must compute absolute days of the year by writing a particular static method. This method should accept parameters representing a month and day and should return the absolute day of the year 2012 that is represented by those parameters. For example, calling this method with the parameter values of 2 and 13 (representing February 13th, 2012) should return 44, and calling it with parameter values of 9 and 19 (representing September 19th, 2012) should return 263. This method should not produce any console output, though the result it returns can be printed by code elsewhere in your program.

Your method must compute its value using a cumulative sum as described in section 4.1 of the textbook. To perform this calculation, 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.

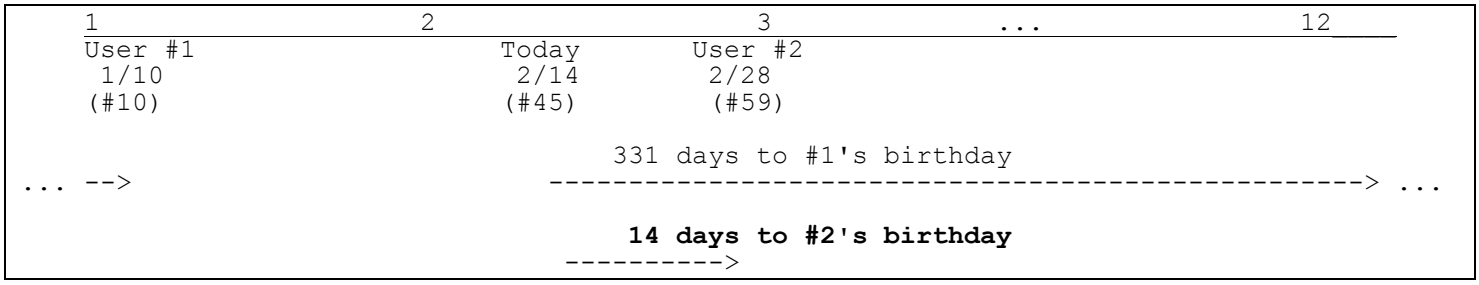
Days until next birthday:

Another major task is computing how many days remain until each user's next birthday. There are several cases to consider. If the user's birthday falls after today's date, it is within the year 2012. However, if it falls before today's date, the next birthday occurs in the year 2013, so you'll need to think of a way to count the days that "wrap around" between today and the end of 2012, and then the start of 2013 to the user's birthday in 2013.

For example, consider the first example output on the previous page. The current day is 1/29, and the two users' birthdays are 10/17 and 11/23. User #1's birthday is 262 days away, and User #2's is 299, so #1 is sooner:



Now consider the second example output. The current day is 2/14, and the two users' birthdays are 1/10 and 2/28. In this case, User #1's next birthday "wraps around" to the next year, 2013, and is 331 days away.



It may be helpful to notice that it takes 331 days to move forward from 2/14 to 1/10, and it takes 35 days to move forward from 1/10 to 2/14, and that these two values add up to 366.

Implementation Guidelines:

Don't forget to place the following statement at the top of your Java file, to be able to read user input:

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

Part of your program's output shows the percent of a year remaining until each user's next birthday. These percentages are rounded to the nearest tenth of a percent. You can achieve this rounding using a rounding method similar to the `round2` method shown in lecture, or by using the `System.out.printf` command. See section 4.4 of the textbook for a description of this command.

Stylistic Guidelines:

Your program should contain **two class constants** representing today's month and today's day. Your program will print this information at the start and will use these values as a point of reference when counting how many days until each user's birthday and which birthday is sooner. Changing these constants should cause your program to modify its behavior. The example outputs shown in this document use several different values for these constants to show behavior with different notions of "today." As stated previously, you can assume that these constants will be used only to represent dates between January 1st and February 28th, so you will not need to worry about a user whose next birthday occurs after February 28, 2013. (This way, you can safely assume that all years are leap years for the purposes of this program.)

To receive full credit on this assignment, you are required to have at least **4 non-trivial static methods** in your program besides `main`. One of these must be the absolute-day-of-year method described previously. 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. The `main` method should be a concise summary of the overall program execution. The `main` method should not contain `println` statements.

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.

For this assignment you are limited to the language features in Chapters 1 through 4, in addition to the logical operators such as `&&` and `||` described in book section 5.2. You are especially forbidden from using any of Java's built-in facilities for manipulating dates, such as the `Date` or `GregorianCalendar` classes.

Give meaningful names to methods and variables, and use proper indentation and whitespace. 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. You might want to start by writing the absolute day method described previously, and test it by calling it from `main` with several fixed values. Do not worry about leap year issues; assume that the year always has 366 days.

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.

Figuring out which user's birthday comes sooner is a task that depends on many of the other pieces of your program, so we suggest you handle it last.