COSC102 **Spring 2015**

OOP : Implementing a `Date` class — Homework 3

Due date: **Wednesday, February 18th, at 11:55 p.m. on Moodle**

Overview

In this assignment, you implement a class that stores a date (year, month, and day) and performs display and accounting operations of its content. A `Date` object

- prints its values in either American or English order format, using either a abbreviated or full month name, and
- computes different relations, such as
  - how many days have passed since the start of its year,
  - how many days in the year are remaining or
  - how many days until another date.

Be sure to use the **same method names** as the one given in this document. You should write your code in the file called `Hw3.java`, stored in a `03` folder under your `cosc102\hw` path.

Provided Code

In the file `Hw3.java`, we defined

- a `Hw3` class that is the client code containing a `main` method and
- a public class `Date`.

Your task is

1. to complete the `Date` class and
2. to test it by creating `Date` objects and calling instance methods to check updates are correct from the `Hw3` client `main` method.

Format

Two different formats are used to display dates. The American date format is Month-Day-Year; the English date format is Day-Month-Year:

```
JAN-8-2013      vs.      8-JAN-2013
```

The abbreviated version uses 3 uppercase letters months while the long version writes the month entirely:

```
8-JAN-2013      vs.      9-January-2013
```

Required Instance Methods

The required methods are listed below. We recommend that you read through them and the provided code before to start implementing. In some cases, **extra methods** (which may be private "helper" methods of the object or of the class) simplify your code. You have to think about them.

Some *optional methods* are indicated in the next section, feel free to start by them as a warm-up exercises. They are not required because they are not the purpose of the `Date` class, but as accessor and getter methods are often part of a class definition you can go ahead and include them.

- **constructors** One with three parameters; the other with four including the format to use.
- **`toString()`** Return a string in the form "D-M-Y" or "M-D-Y" depending of the format value using an abbreviated month.
- **`toStringLong()`** Same as `toString` using an unabbreviated month. (Note this method has to be explicitly called.)
- **`setFormat(int f)`** Set the format to use either to American or English (the value of the integer `f` is either equal to the constant `AM` or `EN`).
- **`isLeapYear()`** Return `true` if the date object is a leap year, `false` otherwise.
- **`dayNumber()`** Return the day number in the year (out of 365/366) this date is.
- **`dayRemaining()`** Return the number of days remaining in the year from this date.
- **`advanceDay()`** Advance this date forward by one day (31 JAN 2012 becomes 1 FEB 2012)
- **`compare(Date d)`** Compare this date to the passed argument `Date` d:
  - if this date is earlier than d `compare` returns -1
  - if this date is later than d `compare` returns 1
  - if the two dates are the same (logical equivalent) `compare` returns 0
- **`equals(Date d)`** Return `true` if this date has the same values than the passed argument `Date` d.
- **`daysTo(Date d)`** Return the number of days that elapses between this date and the `Date` object, d
  - if this date comes after d, the value returned is a positive number
  - if this date comes before d, the value returned is negative
  - if the two objects are equal, the value returned is zero

Beware: All the methods are intended to be leap-year aware.

Your code only has to work for dates in the official Gregorian calendar era (it doesn't have to work before 15 October 1582).

Important: Use the names and methods' signatures we provided: order, spelling, lower/upper case have to be followed. It is important for our testings. You may lose points if you do not respect the instructions.

Optional Methods

- **`getYear`, `getMonth`, `getDay`** Accessor methods that return the requested component of the date.
- **`setYear`, `setMonth`, `setDay`** Getter methods to change a component of the date.

Submission

Submit only one zip file `hw3username.zip` containing only your `Hw3.java` file, where the methods described above are implemented.

Remember that you should declare and use helper functions in your code.

In your `main` method write tests (giving the output as comments) to show us you **fully** tested your implementation. You might find this webpage and this calculator useful.

WARNING: Code that does not compile will automatically receive a zero. Code that doesn't compile is impossible to test, let alone reason about. You should compile and submit often so that you always have a running, even if partially complete, version of your code uploaded.

Challenge Problem: for fun

Add an instance method that returns the day of the week for that date. To determine the day of the date implement this formula.