# CS& 141

## Project 3: "Make a Date"

Name:

A Date program is a staple for those who use a command line interface. You will now get to write your own. In order to do this, you will need to create two related classes (in two files) that contain at least the following::

- **DateDriver:** Main class

This is the "Driver Class" that tests out the **DateAD** class described below.

**methods:**

- **public static void main** -- takes 0, 1 *(a day-of-the-month)*, 2 *(a day-of-the-month and a month)* or 3 *(a day-of-the-month, a month and a year)* COMMAND LINE arguments, creates a DateAD object *(see class definition below)* and prints :
  - the given date
  - the day before
  - the day after
  - the current date
  - an indication if the given date is in the past, present or future

  **0** parameters should make the date be the current date
  **1** parameter should make the date be the current date, with day changed if valid; current date otherwise
  **2** parameters should make the date be the current date with day and month changed if both are valid; current date otherwise
  **3** parameters should make the date match input if all are valid; current date otherwise
  non-numeric arguments should be ignored *(as well as all arguments after non-numeric)*

- **DateAD**: contains information about a single date -- this class will be re-used in further assignments. Instead of just using the built in "**Date**" class *(the one in java.util)*, you will write and use your own.

**properties:**

- **public static finals (constants)** for
  - **lowest year allowed** *(1760 works well -- a leap year just after Gregorian calendar starts in 1752)*
  - **starting day-of-the-week for counting** *(January 1 of 1760 was a Tuesday)*
  - **months in a year** *(12)*
  - **days in a week** *(7)*
  - **an array of month-names** *( [0] is "January", etc.)*
  - **an array of day-of-week-names** *( [0] is "Sunday", etc.)*
- **private short** year -- the full year *(1760+)*
- **private short** month -- *(0 - 11)*
- **private short dayOfMonth** -- *(1 - <numberOfDaysInTheMonth>)*
- **private short dayOfYear** -- *(1 - 365 or 366)*
- **private short dayOfWeek** -- *(0 - 6 -- 0 is Sunday)*

**methods:**

- **4 constructors** -- one default, and ones with **1** *(month)*, **2** *(month, year)*, and **3** *(dayOfMonth, month, year)* parameters. All four constructors should begin by calling **setCurrentDate()**. Any out-of-range parameter will cause the object to remain as the current date.
- **public static boolean isLeapYear** -- takes in a year. The rule is: A year is a leap year if it is divisible by **4** unless it divisible by **100** except if it is also divisible by **400**.

- **public static short daysInMonth** -- takes in a month number and a year and uses a switch to send back the proper number of days.
- **private static short countLeaps** -- takes in a year and returns the number of leap years from the base year to it. In general, if you take the difference a number of years, the number of leap years in that range is the number of years ÷ **4** -the number of years ÷ **100** + the number of years ÷ **400**. Be careful if you go in negative years! An easy algorithm is to start at **1760** *(which was a leap year) and loop through all of the years to the year sent in, adding one to your count if* `isleapyear(yearTested)`.
- **Accessors and Mutators** *(sets and gets)* **for the appropriate properties**. Make sure that mutators change values ONLY if the values are valid and that all properties remain consistent with each other. See the descriptions for **setDayOfYear** and s**etDayOfWeek** below.
- **public DateAD getYesterday** -- returns a DateAD representing the day before this day.
- **public DateAD getTomorrow** -- returns a DateAD representing the day after this day.
- **public boolean lessThan** -- returns a true if this day comes before the DateAD passed in as an argument; false otherwise.
- **public boolean equals** -- returns a true if this day is the same day as the DateAD passed in as an argument; false otherwise.
- **public String toString** -- returns the day-of-week, day-of-month, month, year (as text), e.g., "Tuesday, 2 January, 2007"; overrides Object.toString()
- **public void setCurrentDate()**
  *(requires an* `import Java.util.*;`*)*
  *// **N.B.:** The set of code given here is the **ONLY** place you are allowed to use the*
  *// GregorianCalendar class.*

```
{
   GregorianCalendar cal = new GregorianCalendar();
   month = (short)cal.get(GregorianCalendar.MONTH);
   year = (short)cal.get(GregorianCalendar.YEAR);
   dayOfMonth = (short)cal.get(GregorianCalendar.DAY_OF_MONTH);
   // add your own additional code to properly set dayOfYear and dayOfWeek
   // DO NOT use the GregorianCalendar class to do this--do it yourself
}
```

- **private void setDayOfYear** -- properly sets dayOfYear. *(January 1 is day 1).*
- **private void setDayOfWeek** -- properly sets dayOfWeek.
  *One algorithm is to start at January1, 1760 (which was a Tuesday) and loop through all of the years to the year sent in, adding one to your count for each year (one more if* `isleapyear(yearTested)`*. Then add in dayOfTheYear and mod by DAYS_IN_A_WEEK.*

A sample run of the program might look like:


**sample executable:** Date.jar (*your program should work the same way*). You can run the program by changing to the proper directory and typing (*at the command prompt*): `java -jar Date.jar`
You may use this program to compare answers in your test plan.

---

## Deliverables:

### Physical:

- *The project should be turned in inside a clear plastic Deluxe Locking Project File Folder DOCU Manager or equivalent. This folder should have a simple flap to hold paper in place--NO buttons, strings, velcro, etc. Pages should be in order, not stapled.*
- *Assignment Sheet (printed from the web), with your name written on it, as a cover sheet.*
- *Printed Source Code with Comments (including heading blocks. Describe parameters, no line wrapping)*
- *Sample Input and Output (printed)*
- *A simple test plan including explanations of any discrepancies and reasons for each test. Make sure to test non-numeric and out-of-bounds values. Show actual input and ALL values output as well as ALL expected output.*

### Electronic:

- All .class, .jar, .html (javadocs), and .java files ***zipped*** together. Do not use ***rar*** or any archive format other than ***zip***. Rename the file: "<YourName>_p3.zip".
- Submit this single ***zip*** file by going to **Canvas**, select this class, select the **Assignment** tab on the left, select the ***Assignment 3*** , select the **submission** tab at the top, find the file, and **Submit**.

**Due:** Tuesday, November 12, 2013, 6:00 p.m. *(beginning of class)*
Resubmittal due: Two weeks from the day the project is returned in class (*beginning of class*).