Assignment 4
CSc 210 Fall 2017
Exercises Due September 25th, 8:00 pm MST
Programs Due September 28th, 8:00 pm MST

Introduction

At this point in the course, you know the basics of programming in Java, including input/output, control structures, and writing methods. It is time to put those skills to work and implement some software!

NOTE: THIS ASSIGNMENT HAS TWO DUE DATES.
There are two parts (and two due dates!) to this assignment, exercises done via CodeStepByStep and programs to be turned in via github classroom. The exercises will be due Monday night while the actual programs will be due Thursday night.

You may follow this link to receive your private repository for this assignment:
https://classroom.github.com/a/KgiHXZHH

For each Java program assigned, you will turn in a .java file for that program. We should be able to run "javac *file*.java" and then "java *file*" on lectura and your program should compile and run successfully.

Your Java programs should adhere to our style guidelines we will give to you. Here is the link: https://www2.cs.arizona.edu/classes/cs210/fall17/StyleGuidelines.pdf

Specification Part I: CodeStepByStep

Complete the following exercises on CodeStepByStep by Monday, September 25th, at 8:00 pm.

- Console output->CalculateLine
- Console output->evenSumMax
- Console output->numberSquare
- Console output->printTriangle
- Strings->repeat

Specification Part II: Calendar Programs

When was George Washington born? Was it February 22, 1732? Or, was it February 11, 1731? From the US National Archives, “George Washington was born in Virginia on February 11, 1731 according to the then-used Julian calendar. In 1752, however, Great Britain and all its colonies adopted the Gregorian calendar which moved Washington’s birthday a year and 11 days to February 22, 1732.”

Calendars are used frequently enough that computer systems provide utilities to track and display dates and times. The Unix command

```
$ cal 9 1752
```

prints a calendar for September 1752—note the jump from September 2 to September 14:

```
September 1752
Su Mo Tu We Th Fr Sa
1 2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

The Python calendar module produces a different answer for September 1752:

```
>>> import calendar
>>> calendar.prmonth(1752,9)
September 1752
Mo Tu We Th Fr Sa Su
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

The Unix cal utility reflects the change in calendar systems, from the Julian to the Gregorian, on September 14, 1752 in Britain and its colonies. The Python calendar module shows Gregorian dates. It also follows the European convention of showing Monday as the first day of the week.

For this project you will be working on programming exercises and implementing a Gregorian calendar system that has several functionalities. You will be turning in three different calendar programs that facilitate these functionalities. These programs are detailed below.

Program 1: StartCalendar.java

For this program, you will print the calendar for a month whose characteristics will be given as input. You will first read in a length, which is an integer between 28 and 31 for the length of a month to be printed. This will be followed by another integer representing the first weekday which could be an integer between 1 and 7, where 1 represents Sunday, 2 represents Monday, and so on, through 7 for Saturday. If input does not follow these conditions, exit your program with a status of 1 and print an error message to stderr (Try using System.exit() and System.err.println() for this, google them for more details). The error message just needs to be informative as long as it is printed to stderr. Once this input is read, print out a visualization of a month following those conditions in the style shown below. Here is a transcript of the desired interaction, where file StartCalendar.java contains the program:

```
% javac StartCalender.java
% java StartCalender
31  // User entered length of month
3   // User entered start date
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

Once the calendar is printed, your program may exit with a status of 0. Make sure to turn in your solution to this problem in a file called StartCalendar.java.

Program 2: MonthCalendar.java

This program behaves similar to the last one in that it prints one month out at a time. However, instead of reading in specifics about a month to be outputted, you will read in a month and a year and print out the dates for that specific time out based on the Gregorian calendar.

As stated above, you will read in two integers: First, an int representing what year we are dealing with, and second, an int representing what month out of the year we are looking at (must be 1 - 12). If either of these values is invalid, exit your program with a status of 1 and print an error message to stderr. If the values are correct, you will print out that month. In addition to the format of the first program, the month printed out will also have a label at the top which is the month's name.

Example execution:

```
% javac MonthCalendar.java
% java MonthCalendar
2016    // User inputted year
2       // User inputted month
      February
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29

% java MonthCalendar
1758
12
      December
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

Once the calendar is printed, your program may exit with a status of 0. For this program, you will turn in a file called MonthCalendar.java.

Program 3: YearCalendar.java

This program will take in a year between 1753 and 3000 and print out a calendar with Gregorian dates for that year. You should first read in an integer that is the year to print out. If this is an invalid year, exit your program with a status of 1 and print an error message to stderr. Once given a valid year, you must compute what a calendar would look like for that year and print it out.

Example execution:

```
% javac YearCalendar.java
% java YearCalendar
2017         // year inputted by user
      January               February               March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4            1  2  3  4
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   5  6  7  8  9 10 11
15 16 17 18 19 20 21  12 13 14 15 16 17 18  12 13 14 15 16 17 18
22 23 24 25 26 27 28  19 20 21 22 23 24 25  19 20 21 22 23 24 25
29 30 31              26 27 28              26 27 28 29 30 31


       April                May                 June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1      1  2  3  4  5  6               1  2  3
 2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30
30

        July                August           September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1         1  2  3  4  5                  1  2
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   3  4  5  6  7  8  9
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  10 11 12 13 14 15 16
16 17 18 19 20 21 22  20 21 22 23 24 25 26  17 18 19 20 21 22 23
23 24 25 26 27 28 29  27 28 29 30 31        24 25 26 27 28 29 30
30 31

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4                  1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
```

Once the calendar is printed, your program may exit with a status of 0 and print an error message to stderr. For this program turn in a file called YearCalendar.java.

Some things to consider (for all Calendar programs on this assignment):

1. When does the Gregorian calendar start? Given a specific month, how could I calculate what day of the week it starts at? (Hint, Jan 1, 1753 was a Monday)
2. Leap years may affect some calculations of dates. Leap years give an extra day in February in every year divisible by 4 except if the year is also divisible by 100. A further exception is that if the year is divisible by 400, then it is a leap year. For

example, 1996, 2000, 2004 are all leap years. 1999, 1900, 1902 are NOT leap years.

3. Notice the month labels printed above. What is the spacing when the length of a month name is odd? What about when even? Make sure your spacing matches ours exactly (we will give you a test case to help with this).

Do note that all of these problems have an order to them and should build on each other. If done correctly, you should be able to use large chunks of or even build off your solution for 1 to solve 2 and ditto for 2 and 3.

Miscellaneous

This assignment will be submitted through github classroom. Make sure all of your code you would like to submit is in your repository when the due date arrives.

Note: Your output must match what is defined here in the spec. We will give you a small selection of test cases so you can make sure you have the right format. Do not print extraneous output, you may lose a lot of points (This includes prompts when reading in input!).

Remember, do not cheat! Refer to the syllabus and first lecture for more information.