CSC 110: Fundamentals of Programming I
Assignment #3: Static Methods, Parameter Passing, & Return Values

Due date

Friday, October 7th, 2016 at 5:00 pm via submission to conneX.

How to hand in your work

Submit the requested files for part (b) (see below) through the Assignment #3 link on the CSC 110 conneX site. Please make sure you follow all the required steps for submission (including confirming your submission). Part (a) is not marked.

Learning outcomes

When you have completed this assignment, you should understand:

- How to use for-loops and nested for-loops to generate output for structured text.
- How to use a simple if-statement..
- Use methods to organize your program.
- Use method parameters and method return values to direct the computation performed in your solution.

Part (a): Problems from the Textbook

Complete Chapter 2 self-check problems 21 to 35; complete the Chapter 3 self-check problems 1 to 11 and compare your answers with those available at:

`http://www.buildingjavaprograms.com/self-check-solutions-4ed.html`

Part (b): Calendars

In this assignment you will write a program named *CalendarCanada.java* producing a month's calendar as output. The program will:

- Ask the user for the number of days in the month.
- Ask the user for the date on which the first Sunday falls.
- Print out the resulting calendar.

For this particular program you need not worry about legal calendars, i.e., assume the user provides sensible values for number of days and starting Sunday.

Here is an example of the program's execution (which should produce a calendar looking like that for October 2016):

```
$ java CalendarCanada
Number of days in month? 31
Date of first Sunday? 2
   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 |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+
```

Notice the program is named *CalendarCanada*; in other countries (e.g., Germany) the days of the week on a calendar line begin with Monday (or, rather, "Montag") than Sunday.

Your program must have at least three methods (besides *main()*):

- *public static void printSeparator():* This takes no parameters and returns no value. It simply outputs the "horizontal" separator lines at the top and bottom of the calendar (i.e., those with "+-----+", etc.)

- *public static String rightJustify(int value, int spaces):* The method returns a new string of *width* spaces where *value* appears at the end of the string. For example, *rightJustify(10, 5)* will produce a string starting with three spaces and ending with "10" (i.e., five characters in the string in total, or "␣␣␣10" where '␣' is denotes a space character.) Hint: The bottom of page 193 in the textbook suggests one way of writing such a method.

- *public static void outputCalendar(int numDays, int startingSunday):* Contains the Java statements needed to output the calendar corresponding to the values passed in as parameters. The method will, of course, make use of the other methods describe above. For example, the result appearing on the console given a call to ~~*outputCalendar(31, 3)*~~ *outputCalendar(31, 2)* is what appears in the sample execution up above. (Other examples are shown in the provided file named *examples.pdf*.) Amongst other problems to solve, you will need to carefully consider how best to print the "blank days" (i.e., those in the starting and ending lines of each calendar) and to do so such that blank days only appear if they are needed given the number of days in the month or the starting Sunday or both.

We expect your program to prompt the user as shown in the example with code contained in the main() method. *Please ensure your prompts (i.e., the "questions" asked of the user) use the same text and order as in the example.*

File to submit: CalendarCanada.java

Grading: What the marker will look for

- *Documentation*: Documentation in the implemented code must include the author's identification and the purpose of the program. Each group of instructions must be preceded with a comment describing their common purpose. Each method must be preceded with a comment describing its purpose as well as an indication of the input to and the output from the method.
- *White Space and Indentation* in the code and adherence to Java naming conventions.
- Your methods should *accept input parameter values and return an output* as described above. Your program should use the methods wherever it is appropriate. Other suitable methods can be created, of course, or no other methods, if that is appropriate.
- *Compiles and produces correct output:* The code constitutes a valid Java program (i.e. compiles *and* runs without error on the ECS 250 lab machines). It must accept input and print output as described above.
- Prompts are provided as described in this document.

Grading scheme

- "A" grade: An exceptional submission demonstrating creativity and initiative going above and beyond the assignment requirements. The program runs without any problems using the specified methods. Any extra work appears in the file named *CalendarExtra.java*, and identified within the file (i.e., Class comment) is how you have extended the assignment to demonstrate creativity and initiative.

- "B" grade: A submission completing the requirements of the assignment. The program runs without any problems, uses the specified methods.

- "C" grade: A submission completing most of the requirements of the assignment. The program runs with some problems but has the specified methods, and yet might not have the expected output.

- "D" grade: A serious attempt at completing requirements for the assignment. The program runs with major problems, or is missing specified methods.

- "F" grade: Either no submission given, the submission does not compile, or submission represents very little work.