CSE 142, Autumn 2006
Programming Assignment #2 (Space Needle)
Due: Wednesday, October 11, 2006, 9:00 PM

Program Description:

This program tests your understanding of expressions and variables, `for` loops, and class constants, and also reinforces your previous material on static methods and `println` statements. You should write a Java class named `SpaceNeedle` that is saved into a file named `SpaceNeedle.java`. Your program should produce the following figure as output (which is supposed to look like Seattle's Space Needle):

```
            ||
            ||
            ||
            ||
         __/||\__
      __/:::||:::\__
   __/::::::||::::::\__
__/:::::::::||:::::::::\__
|""""""""""""""""""""""""|
\_/\/\/\/\/\/\/\/\/\/\/\_/
  \_/\/\/\/\/\/\/\/\/\_/
    \_/\/\/\/\/\/\/\_/
      \_/\/\/\/\/\_/
            ||
            ||
            ||
            ||
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
        |%%||%%|
         __/||\__
      __/:::||:::\__
   __/::::::||::::::\__
__/:::::::::||:::::::::\__
|""""""""""""""""""""""""|
```

You should exactly reproduce the format of this output. This includes having identical characters and spacing.

One way to write a Java program to draw this figure would be to write a `System.out.println` statement that prints each line of the figure. However, this solution would not receive full credit. A major part of this assignment is showing that you understand `for` loops.

Therefore, in lines that have repeated patterns of characters that vary in number from line to line, represent the lines and character patterns using appropriate nested `for` loops. (The complex figure in Chapter 2's case study is a good example of this.) It may help you to write pseudo-code and tables to understand the patterns in the output, as described in the textbook and lecture.

Another significant component of this assignment is the task of generalizing the program using a class constant that can be changed to adjust the size of the figure. See the next page for a description of this constant and how it should be used in your program.

The course web site will contain expected output files that show you the expected output if your constant height is changed to various other values.

Submission and Grading:

Turn in your Java file electronically from the Assignments section of the course web page. Your program will be graded on its "external correctness" (whether the program compiles and produces exactly the expected output) and its "internal correctness" (whether your source code follows the stylistic guidelines specified in this document).

Stylistic Guidelines:

The following are the stylistic criteria on which you will be graded:

- Use of `for` loops (nested as appropriate)

  This program is intended to test your knowledge of Chapters 1 and 2, especially nested `for` loops. If you are interested, you may use the Java language features from Chapter 3, although you are not required to do so and you will receive no extra credit for doing so. You may not use any programming constructs that are not in Chapters 1 through 3 of the textbook.

- Use of static methods for structure and elimination of redundancy

  Continue to use static methods to structure your solution. Structure your program in such a way that the methods match the structure of the output itself. Avoid significant redundancy; make sure that no substantial groups of identical printed lines or other statements appear in your code. No non-blank `println` statements should appear in your `main` method.

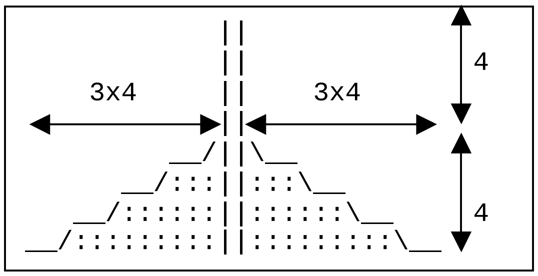
- Source code aesthetics (commenting, indentation, spacing, identifier names)

  You are required to properly indent your code and will lose points if you make significant indentation mistakes. See the textbook for an explanation and examples of proper indentation.

  Give meaningful names to methods and variables in your code. Follow Java's naming standards about the format of `ClassNames`, `methodAndVariableNames`, and `CONSTANT_NAMES`. Localize variables whenever possible -- that is, declare them in the smallest scope in which they are needed.

  Include a comment at the beginning of your program with basic information and a description of the program **and include a comment at the start of each method**. Your comments should be written in your own words and not taken directly from this document.

- Class constant for figure's size

  ```
              ||
              ||
              ||
              ||
           __/||\__
        __/:::||:::\__
     __/::::::||::::::\__
  __/:::::::::||:::::::::\__
  ```

  You should create a class constant to represent the height of the various pieces of the figure. The various subfigures in the middle of this output have a height of 4, so 4 should be the value of your constant. These subfigures have the property that their height determines their width; therefore, you do not need a second constant to represent the width. Your figure must be based on a constant value of 4 to receive full credit.

  On any given execution your program will produce just one version of this figure. However, you should refer to the class constant throughout your code, so that by simply changing your constant's value and recompiling, your program would produce a proportional figure of a different size. Your program should work for any constant value of 2 or greater.

How to Get Started:

This program is best completed in stages. We strongly recommend that you do not worry about the constant at first. Write an initial version of the program without a constant, focusing on each section of the figure individually. Use loop tables and/or pseudocode, as shown in class, to help you deduce the patterns in the output. After your figure looks correct at the default size, begin a second version of your program that adds the constant. See Chapter 2's case study in the textbook for an example of a program that uses a constant while drawing a complex figure.