Take-home Assessment 2: Space Needle

due October 13, 2020, 11:59pm

This assignment will assess your mastery of the following objectives:

- Write a functionally correct Java program to produce specified console output.
- Write arithmetic expressions in Java.
- Write `for` loops, including nested loops, to repeat code and manage control flow.
- Use class constants to represent and modify important values in a program.
- Follow prescribed conventions for spacing, indentation, naming methods, and header comments.

Expected Output (size 4)

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

Program Behavior

Part A: ASCII Art

The first part of your assessment is to write a program that produces any text art (sometimes called "ASCII art") picture you like. Your program can produce any picture you like, with the following restrictions:

- The picture should be your own creation, not an ASCII image you found on the Internet or elsewhere.
- The picture should not include hateful, offensive, or otherwise inappropriate images.
- The picture should consist of between 3 and 200 lines of output, with no more than 100 characters per line.
- The picture must not be substantially similar to your solution for Part B, consist entirely of reused Part B code, or be substantially similar to a related CSE142 assignments from a previous quarter.
- The code must use at least one for loop or static method but should not contain infinite loops.
- The code must not use material beyond Ch. 3 of the textbook.

This part of the assessment will only contribute to the Behavior dimension grade. It will not factor in to grading on the other dimensions.

Part B: Space Needle

The second part of your assessment is to produce a specific text figure that is supposed to look like Seattle's Space Needle. Your program should ***exactly*** reproduce the format of the output to the left, including characters and spacing. The Mark button in Ed will be helpful in confirming you've produced the correct output.

As on assessment 1, you must **exactly** match the output.

In addition to producing the default Space Needle shown below, your program will need to be able to be easily modified to produce similar images of different sizes. You will still only turn in one program, and your program will only produce one size of output on any given run. However, we should be able to make a single change to your code and recompile to produce a different size. See below for more details.

You should be able to change a **single** value at **one** point in the code to scale the figure.

Development Strategy

Since this is a more complex program, we suggest you approach the program in stages rather than trying to implement everything at once. We specifically recommend following these steps:

(1) **Tables:** Utilize loop tables as demonstrated in class to find the patterns and expressions for repeated sequences of characters.

(2) **Default size:** Implement the code necessary to produce the Space Needle using the default size of 4. DO NOT think about other sizes yet.

(3) **Scaling:** Once your default size completely works, add your constant (see below) and modify your code so the output can be scaled to different sizes.

```
Expected Output (size 3)
```

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

Implementation Guidelines

Like all CSE 142 assessments, Part B of this assessment will be graded on all four dimensions defined in the syllabus. Be sure to adhere to the following guidelines:

Using `for` loops

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 be redundant and inflexible. Instead, you are required to use `for` loops to create a more generalized version of the program. Specifically, in lines that have repeated patterns of characters that vary in number from line to line, your code should print the lines and character patterns using nested `for` loops. You may find it helpful to write pseudocode and tables to understand the patterns, as described in the textbook and lecture.

Scaling with a class constant

As described above, your program should be able to be easily changed to produce a figure of a different size. For example, a Space Needle of size 3 is shown to the left. To achieve this, your program should include one (and only one) class constant that represents the size of the figure. (The default size of the figure is 4.) *To ensure our testing and grading scripts work correctly, you **must** name your constant SIZE.* Throughout your program, any values that are related to the size of the figure should refer to this constant so that a different size of figure can be produced by changing **only the value of the constant**. Your program should work correctly for any size greater than or equal to 2.

Be sure to name your constant SIZE or you won't be able to pass our tests!

The height of the Space Needle's body grows with the **square** of the figure's size. For example, in our size 4 default, the body is 16 (4 x 4) lines tall.

The course website and Ed lesson will contain files that show you the expected output if your size constant is changed to various other values. You can use the Mark button in Ed to verify the correctness of your output for various values of the size constant.

Permitted Java Features

For this assessment, you are restricted to Java concepts covered in chapters 1 and 2 of the textbook. In particular, you MUST use `for` loops and class constants (see above) and you **may not** use parameters. You also still may not use the `\n` escape sequence.

Code Quality Guidelines

In addition to producing the desired behavior, your code should be well-written and meet all expectations described in the grading guidelines and the Code Quality Guide. For this assessment, pay particular attention to the following elements:

Capturing Structure

As in assessment 1, you should use static methods to accurately capture the structure of the output in your program. In particular, your `main` method should be a concise summary of the program and reflect the structure of the figure being produced. Your program should not include any `System.out.println` or `System.out.print` statements or any `for` loops in `main`.

Try to identify the large chunks of the output and write a method for each, as in the tapestry example from lecture.

Reducing Redundancy

You should continue to reduce redundancy as much as possible in your program by making good use of static methods. Specifically, you should not have any redundant code to produce the same full line of output. Instead, use methods to remove this redundancy. Similar to assessment 1, this only applies to full-line redundancy. You are not expected to deal with partial-line redundancy, such as the two groups of colons in this line of output:

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

This will result in *some* redundant code. This is OK, but **only** if that code is producing partial-line redundancy as described here. All other redundant code should be fixed.

Code Aesthetics

Your code should be properly indented, make good use of blank lines and other whitespace, and include no lines longer than 100 characters. Your class, methods, variables, and constant should all have meaningful and descriptive names and follow the standard Java naming conventions. (e.g. `ClassName`, `methodOrVariableName`, `CONSTANT_NAME`) See the Code Quality Guide for more information.

Commenting

Your code should include a header comment at the start of your program, following the same format described in assessment 1. Your code should also include a comment at the beginning of each method that describes that methods behavior. Comments should be written in your own words (i.e. not copied and pasted from this spec) and should not include implementation details (such as describing loops or expressions included in the code). See the Code Quality Guide for examples and more information.

Getting Help

If you find you are struggling with this assessment, make use of all the course resources that are available to you, such as:

- Reviewing relevant examples from lessons, section, and lab
- Reading the textbook
- Visiting office hours
- Posting a question on the message board

Collaboration Policy

Remember that, while you are encouraged to use all resources at your disposal, including your classmates, **all work you submit must be entirely your own**. In particular, you should **NEVER** look at a solution to this assessment from another source (a classmate, a former student, an online repository, etc.). Please review the full policy in the syllabus for more details and ask the course staff if you are unclear on whether or not a resource is OK to use.

Reflection

In addition to your code, you must submit answers to short reflection questions. These questions will help you think about what you learned, what you struggled with, and how you can improve next time. The questions are given in the file `SpaceNeedleReflection.txt` in the Ed assessment; type your responses directly into that file.

Running and Submitting

You can run your program by clicking the "Run" button in Ed. This will compile and execute your code and show you any errors, or the output of your program if it runs correctly. If you believe your output is correct, you can submit your work by clicking the "Mark" button in the Ed assessment. You will see the results of some automated tests along with tentative grades. **This grade is not final until you have received feedback from your TA.**

You may submit your work as often as you like until the deadline; we will always grade your most recent submission. Note the due date and time carefully—**work submitted after the due time will not be accepted**.