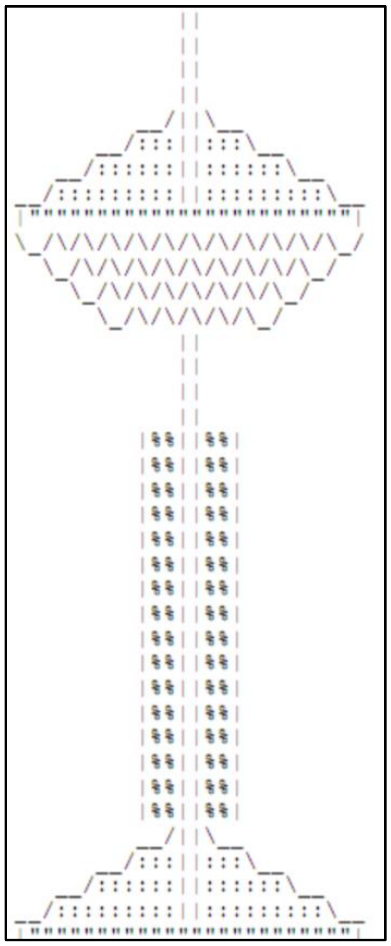
Project 2: Scalable Space Needle

High Level Description:

The purpose of this project is to practice using nested for loops, input parameters, print statements, and static methods in Java.

There are two parts to this project. In part one, you will create a method called "printSpaceNeedle" that takes an input parameter called size and prints an ASCII representation of the Seattle Space Needle to the console. In part two, you will find or design another ASCII pattern and create a method that will prints your pattern to the console.

Both of your methods should be in a single class called ScalableSpaceNeedle.java. Make sure you write clean, neat code and test your program with different input sizes to make sure it works properly. See below for detailed instructions on each part.



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

Part 1 – printSpaceNeedle(int size) { … }

Create a method that returns nothing and takes an input of an integer called size. Your method should print an ASCII image of the space needle that scales up or down depending on the size. In the Teams assignment for this project, you will find text files with examples of the image at sizes 2, 3, 4 and 7. Below is an example at size 4.

I would recommend splitting the design into smaller pieces, writing some pseudocode, then assembling your pieces together. Remember, you can have your static methods call other static methods!

Part 2 – printMyDesign(int size) { … }

Find or design another work of ASCII art and create a method that will print your artwork at different sizes. Your ASCII art can be whatever you want, so long as you can create it using nested for loops. Let me know if you need help finding examples of different ASCII art or if you think the design you chose needs approval.

Here is a link to some ASCII art examples: ASCII Art Archive

Extensions – How can I go above and beyond?

1.) Create a program that will generate different art at different sizes depending on user input (Scanner).
2.) Create a landscape in ASCII art with multiple different landmarks that can be scaled to multiple sizes.

Grading:

Your grade on this project will be based around the following categories:

1. **External Correctness (Output)**

   Does your code produce the exact desired result. When run, will your code print the correct lyrics to the console without any typos, errors, or misspellings?

2. **Internal Correctness (Application of Unit 2 Material)**

   Your code must include the required components from the chapter. Here are your specific constraints on your code:

   a. You may not include any print or println statements in your main method. *(You must call static methods instead)!*
   b. Your static methods must include more than one print(ln) statement. *(It is bad programming style to have a method print just one line to the console. Instead, look for groups of lines that appear in multiple places and create methods to represent those groups.)!*
   c. You may not use content outside of unit 1. *(if/else statements, switch statements etc. unless you are doing an extension)!*

3. **Coding Style**

Your code must be neat, concise, and ***readable***! Refer to the following coding conventions for this project:

- Use conventional indentation and whitespace.
- Avoid lines that go over 100 characters in length – you can check this at the bottom of your Eclipse window.
- Give meaningful names to all methods in your code.
- Follow Java's naming standards about the format of ClassNames and methodNames.
- Include a blank line followed by meaningful comments at the start of each method other than the main method, describing its behavior. This may be javadoc or regular block comments. This should not be how your method works, but an overview of what it is supposed to do.
- Include and complete the header at the top of all your submitted .java files (except the ones marked as "DO NOT MODIFY"). A template is included in your starter code.
- Opening curly braces must be either at the end of the line beginning the block or on their own new line.
    - *For example:*

```
public static void main(String[] args) {
   or
public static void main(String[] args)
{
```

- Closing curly braces must be on their own line.