CSE 142, Summer 2013
Programming Assignment #2: ASCII Art / Rocket Ship (16 points)
Due Tuesday, July 9, 2011, 11:30 PM

This assignment will give you practice with `for` loops, static methods, `print`/`println` statements and a class constant. This assignment is worth 16 points instead of the normal 20 points.

Part A: ASCII Art (2 points):

The first part of your assignment is to write a program that produces any text art (sometimes called "ASCII art") picture you like. Write a Java class called `AsciiArt` in a file called `AsciiArt.java`. Your program can produce any text picture you like, with the following restrictions and details:

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

If your Part A program compiles and runs successfully and meets the above constraints, it will receive the full 2 points. Part A will not be graded on style ("internal correctness"). However, we still encourage you to practice good style and commenting.

Part B: Rocket Ship (14 points):

SIZE = 3

```
     /**\
    //**\\
   ///**\\\
  ////**\\\\
 /////**\\\\\
+=*=*=*=*=*=*+
|../\..../\..|
|./\/\../\/\.|
|/\/\/\/\/\/\|
|\/\/\/\/\/\/|
|.\/\/..\/\/.|
|..\/....\/..|
+=*=*=*=*=*=*+
|\/\/\/\/\/\/|
|.\/\/..\/\/.|
|..\/....\/..|
|../\....../\..|
|./\/\../\/\.|
|/\/\/\/\/\/\|
+=*=*=*=*=*=*+
     /**\
    //**\\
   ///**\\\
  ////**\\\\
 /////**\\\\\
```

In this portion of the assignment, you are to **exactly** reproduce the image of the rocket ship to the left, including characters and spacing. Turn in a class named `DrawRocket` in a file named `DrawRocket.java`.

One way to write a Java program to draw this figure would be to write a single `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.

In lines that have repeated patterns of characters that vary in number from line to line, represent the lines and character patterns using nested `for` loops (see Chapter 2's case study). It may help to write pseudocode and tables to understand the patterns, as described in the textbook and lecture.

Another significant component of this assignment is the task of generalizing the program using a single class constant that can be changed to adjust the size of the entire 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 files that show you the expected output if your size constant is changed to various other values. You should use our Output Comparison Tool on the course web site to measure numbers of characters and to verify the correctness of your output for various values of the size constant.

(continued on back)

Style Guidelines (for Part B):

Use nested for loops and static methods for structure and elimination of redundancy

While the bulk of this program is intended to test your knowledge of `for` loops, you should continue to use static methods to structure your solution in such a way that the methods match the structure of the output itself. No `println` statements should appear in your `main` method. Avoid significant redundancy; use methods so that no substantial groups of identical statements appear in your code. You do not need to use methods to capture redundancy in partial lines, such as the two groups of slashes in the following line:

```
|./\/\../\/\.|
```

To produce such a line, you may have one or more pairs of identical loops in a method. This is acceptable for this assignment. Though you are permitted to eliminate this redundancy using Java features from Chapter 3 such as parameters, you are not required to do so and will receive no extra credit for doing so. You may not use any Java constructs beyond Chapter 3.

Class constant for figure's size

You must create one (and only one) class constant to represent the size of the pieces of the figure. Use **3** as the default value of your constant to produce the figure shown on page 1. Your figure must be based on that exact value to receive full credit.

On any given execution your program will produce just one version of the 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 figure of a different size. You should thoroughly test your program by comparing your output with the expected output for the various constant values provided via the Output Comparison Tool on the course website.

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 examples of proper indentation. No line of your code should be over 100 characters long.

Give meaningful names to methods and variables in your code. Follow Java's naming standards about the format of `ClassNames`, `methodAndVariableNames`, and `CONSTANT_NAMES`.

Include a comment header at the beginning of your program with basic information and a description of the program. **Also include a comment at the start of each method**, describing its behavior. Your comments should be in your own words.

Development Strategy (How to Get Started):

This program is best completed in stages. We strongly recommend the following development strategy:

1. **Tables**: Examine the output and write tables to discover the patterns of repeated characters on each line.
2. **Code w/o Constant**: Completely write the Java code to draw the rocket ship at its default size of 3.
3. **Code w/ Constant**: Add a constant to your code so that the rocket can scale to different sizes.

To summarize, you should not worry about the constant at first. Write an initial program without a constant, using loop tables or pseudocode to help you deduce the patterns in the output. After your figure looks correct at the default size, begin a second version with the constant. See Chapter 2's case study for a good example program.

Turn in both files electronically via the link on the Homework page of the course website.