CSE 142, Autumn 2007
Programming Assignment #2: ASCII Art / Book (16 points)
Due Wednesday, October 10, 2007, 4:00 PM

This two-part assignment tests your understanding of expressions, variables, `for` loops, and class constants, and also reinforces previous material on static methods and `println`. Turn in two Java programs described below.

Part A: ASCII Art Contest (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 named `AsciiArt` in a file named `AsciiArt.java`. Your program can produce any text picture you like, with the following restrictions and details:

- The picture should be your own work, not an ASCII image you found on the Internet or elsewhere.
- The number of lines should be at least 3 but no more than 100, and no more than 100 characters per line.
- The picture should not include hateful, offensive, or otherwise inappropriate images.
- The code to produce the picture should use at least one `for` loop.
- The picture must not be identical to your solution for Part B or consist entirely of reused Part B code.
- Your code should not use material beyond Ch. 3 of the book and should run without user interruption.
- 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").

Student pictures will be posted anonymously on the course web site at a later date as part of a voting contest.

Part B: Book (14 points):

The second part of your assignment is to produce a specific text figure that is supposed to look like an advertisement with our course textbook, *Building Java Programs,* sitting on its side. Write a Java class named `Book` in a file named `Book.java`. Your program should produce the following output:

```
           +------------------------------+
          /                           ___/
         /                        ___/__//
        /                     ___/__/__///
       /                  ___/__/__/__////
      /               ___/__/__/__/__/////
     /            ___/__/__/__/__/__//////
    /         ___/__/__/__/__/__/__///////
   /      ___/__/__/__/__/__/__/__////////
  /   ___/__/__/__/__/__/__/__/__/////////
 /___/__/__/__/__/__/__/__/__/__//////////
+------------------------------+//////////
|    Building Java Programs    |//////////
|    Building Java Programs    |////////
|    Building Java Programs    |//////
|    Building Java Programs    |////
|    Building Java Programs    |//
+------------------------------+
Now only $90!
```

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 nested `for` loops. (Chapter 2's case study is a good example of this.) It may help you to write pseudocode and tables to understand the output patterns, 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 size constant is changed to various other values. You can use our Output Comparison Tool to measure numbers of characters.

Stylistic Guidelines:

Use of `for` loops (nested as appropriate)

This program is intended to test your knowledge through Chapter 2, especially nested `for` loops. If you are interested, you may use the Java 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 constructs that are not in Chapters 1 through 3.

Use of static methods for structure and elimination of redundancy

Continue to use static methods to structure your solution in such a way that the methods match the structure of the output itself. Avoid significant redundancy; use methods so that no substantial groups of identical statements appear in your code. No `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.

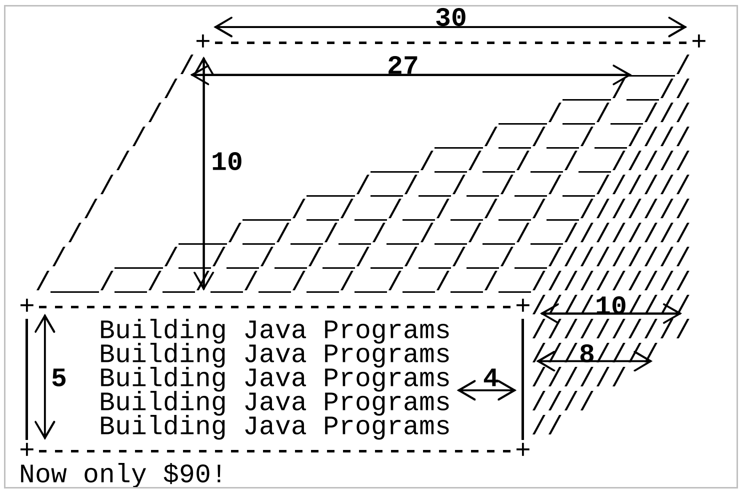
Give meaningful names to methods and variables in your code. Follow Java's naming standards about the format of `ClassNames`, `methodAndVariableNames`, and `CONSTANT_NAMES`. Localize variables whenever possible; that is, declare them in the smallest scope in which they are needed.

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.

Class constant for figure's size

You should create one (and only one) class constant to represent the size of the various pieces of the figure. Use **10** as the value of your constant. 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. Your program should scale correctly for any even constant value of 8 or greater. (We will not grade with an odd value or value below 8.)



```
Building Java Programs
Building Java Programs
Building Java Programs
Building Java Programs
Building Java Programs
Now only $90!
```

How to Get Started:

This program is best completed in stages. We recommend that you do 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 an example program that uses a constant while drawing a figure.

Turn in your two Java files electronically from the Homework section of the web site. Part B 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 in this document). As a point of reference, our solution to this program occupies **95 lines** including comments and blank lines, though you do not have to match these totals exactly.