AP CS: Lesson 4: Building a Ziggurat, a Scalable Complex Figure

Name: ________________________ Period: ___

Java Suggestions:

Recommendations for managing complexity:

1. Design the program (think about steps or methods needed).
   - write an English description (Psuedocode) of steps required
   - use this description to decide the methods needed and loops

2. Determine the Nested Loops: (these are just guidelines for figures, actual use may vary)
   - Outer (top) loop to determine the number of lines / rows, remember to have a newline (println) at the end of the outer loop
   - If a character's number does not vary (i.e. only appears once), then it simply needs to be printed (print)
   - If a character's number varies per line, then you need a for loop and require its formula

3. Create a table that count out the patterns of the characters that vary to find their equations
   - use a `for` loop starting at 1 going to the desired length, determined by…
   - for each set of characters use a table to determine their pattern and then the linear equation
   - In General (but not always):
     - For patterns that increase, the constant in the linear equation does not change based on the Scaling Size.
     - For patterns that decrease, the constant in the linear equation typically change based on the Scaling Size and you need to determine its linear equation.

Building your code incrementally:

Write a small portion of code for to start the program, (for example: the first method, a couple loops that build the first couple characters of the figure, etc) compile it, check the results and get that working before writing more code and the whole program. Why?

1. It is easier to find a problem in a small piece of code than a large one, especially if there are multiple mistakes.
2. If you have a repeated mistake you can correct it once and not need to fix it multiple times.
3. This manages complexity, starts small and builds up, learning along the way.
4. You can copy & paste working code to add further occurrences, but keep in mind...
5. Look for redundant code to better decompose your final code; in general if lines of code occurs more than twice, you should consider turning them into their own method.

Use what you already have:

If you get stuck, go back to your lab worksheet problems, our previous lectures & sample code, using them as starting points & examples for your new program. A good way to get unblocked is to start by editing a program that is already working. *Qualifier: eventually, the goal is for you to be able to write code without using previous examples, but it's a good way to get started.*

AP CS: Lesson 4: Building a Ziggurat, a Scalable Complex Figure

Class Notes:

Exit Ticket - Only Return this to or Speak with Mr. Bergquist if you Answer "No" to any of these. Thanks.

1. I understand how to nest for Loops to make Simple Figures: __ Yes __ Somewhat __ No
2. I understand how to use loops to make Complex Figures images: __ Yes __ Somewhat __ No
3. I understand how to use Class Constants to Scale figures: __ Yes __ Somewhat __ No

Additional comments, especially if you answered "No" to any of the above what would you like to more details on? Thanks