Assignment 07: Docs and Recursion

CS 140 with Dr. Sam Schwartz

Due: Sunday, November 2 at 11:59pm via Canvas Upload

1 Purpose

The purpose of this assignment is to build confidence using recursion and using JavaDoc.

2 Tasks

In this assignment, you will:

- Create methods that will print at least two different shapes that we have constructed in class or on assignments before (e.g., Scottish flag, checkerboard, empty rectangle, etc.)
- Create at least one new shape/figure of your choosing. Get creative!
- Do so only through the use of recursive functions. You are not allowed to use loops in this assignment.

The size of these shapes should be passed as a parameter to the method which creates them.

Your code should contain a main function which calls each shape you created, so that when your code is run it outputs all of them to the console all at once.

Example:

```
// TODO: Write JavaDoc
public static void main(String[] args) {
 printCheckerboard(4, 8, 8); // Prints an 8x8 checkerboard, where each square is 4x4 characters.
 printEmptySquare(5); // Prints a 5x5 square with a border and a whitespace interior.
 // TODO: Write at least one more method.
}
```

Be sure to document everything with JavaDoc.

Submit screenshots and a single file called RecursiveShapes.java to Canvas.

3 Grading Criteria

This assignment is a bit subjective in it's grading criteria, because the tasks are more creative. In general I am looking for these elements:

Validity

Student submitted a `RecursiveShapes.java` file
... which contained code that printed multiple shapes to the console
... two of which we've seen in class somewhere before
... and one of which is a new creation
... none of which used loops
... and all of which used recursive function(s) somewhere in their construction
... and which were sufficiently "tight" (meaning, at one extreme, the author did not just put the whole thing in a single println in a base case and call it a day)

Screenshot

... which was separately documented via an uploaded screenshot of the code/console

Readability

... and contained a sufficient number of comments explaining and documenting the code,
each written in professional English
... which utilized JavaDoc
... and the code as a whole used consistent indentation, naming, and formatting conventions

Fluency

... and the deliverable was executed in such a way that an experienced practitioner
would not find the deliverable "weird-in-a-bad-way" or unduly jarring
... and demonstrated modular design with methods called appropriately from `main`