Scalable Fish ASCII Art Project

**Part 1**: Create a program that prints various size ASCII art fish by calling the drawFish method specifying the fish size as an integer parameter. Your main method should call the drawFish method, passing the fish size parameter. The fish size parameter is the only value that should change in order to print different size fish! The drawFish method must be able to handle illegal sizes (e.g. 0, negative sizes) in which case it should do nothing.

### Size 1 fish

vvv

v

vov

vooov

vooooov

vooooooov

voo oov

v v

### Size 2 fish

vvvvv

vvv

v

vov

vooov

vooooov

vooooooov

vooooooooov

vooooooooooov

voooo oooov

voo oov

v v

## Size 3 fish

vvvvvvv

vvvvv

vvv

v

vov

vooov

vooooov

vooooooov

vooooooooov

vooooooooooov

vooooooooooooov

vooooooooooooooov

voooooo oooooov

voooo oooov

voo oov

v v

## Part 1 Grading Scheme/Rubric

|  |  |
| --- | --- |
| ***Functional Correctness*** | |
| Correctly define a fish size parameter | 1 point |
| Correctly define the main method. | 1 point |
| Correctly call the drawFish method in main. | 1 point |
| Correctly define the drawFish method with correct parameter. | 1 point |
| Proper use of procedural decomposition.   * All work done in drawFish (0 point) * Partial procedural decomposition (1 point) * Full procedure decomposition (2 points) | 2 points |
| Correctness of output.   * Correct output for illegal sizes (+1 point) * Correct output for size 1 (+1 point) * Correct output for odd sizes (+1 point) * Correct output for even sizes (+1 points) | 4 points |
| **Total** | **10 points** |
| ***Coding Style*** | |
| Readability: Indentation.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 points |
| Readability: Meaningful method and variable names.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 points |
| Readability: Comments/Documentation.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 points |
| Efficiency: Code duplication.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 point |
| Efficiency: Smart algorithm.  Good: 2 point, Fair: 1 point, Poor: 0 point | 2 point |
| **Total** | **10 points** |

**Part 2**: Modify your main method to call the foodChain method. The foodChain method takes three integer parameters: minSize, maxSize and sizeStep that are defined as *class constants*. It will call the drawFish method to print several fishes from maxSize to minSize decreasing the fish size by sizeStep on each fish so that the resulting output should look like the big fish is chasing and eating the smaller fish. The following example demonstrates a call to foodChain(1, 5, 2) with 1 as minSize, 5 as maxSize and 2 as sizeStep. This will print the fishes sized 5, 3, and 1. The foodChain method may assume the parameters are valid. In other words, it doesn’t need to check for parameter error. It is important to note that all the fishes printed must be center justified. In order to do this, the drawFish method must accept a second integer parameter leftPadding that specifies the number of spaces inserted to the left of the fish so that it will be aligned with bigger fishes at the center column.

## Example output of foodChain(1, 5, 2)

vvvvvvvvvvv

vvvvvvvvv

vvvvvvv

vvvvv

vvv

v

vov

vooov

vooooov

vooooooov

vooooooooov

vooooooooooov

vooooooooooooov

vooooooooooooooov

vooooooooooooooooov

vooooooooooooooooooov

vooooooooooooooooooooov

vooooooooooooooooooooooov

voooooooooo oooooooooov

voooooooo oooooooov

voooooo oooooov

voooo oooov

voo oov

v v

vvvvvvv

vvvvv

vvv

v

vov

vooov

vooooov

vooooooov

vooooooooov

vooooooooooov

vooooooooooooov

vooooooooooooooov

voooooo oooooov

voooo oooov

voo oov

v v

vvv

v

vov

vooov

vooooov

vooooooov

voo oov

v v

# Grading Scheme/Rubric

|  |  |
| --- | --- |
| ***Functional Correctness*** | |
| Correctly define the minSize, maxSize, sizeStep class constants.   * Fully correct (2 points) * Partially correct (1 point) * Missing or incorrect (0 point) | 2 points |
| Correctly call the foodChain method in main. | 1 point |
| Correctly define the foodChain method with correct parameters. | 1 point |
| Correct implementation of the for-loop in the foodChain method. | 1 point |
| Proper use of procedural decomposition.   * All work done in foodChain (0 point) * Partial procedural decomposition (1 point) * Full procedure decomposition (2 points) | 2 points |
| Correctness of output.   * Correct alignment for odd size, odd center column (+1 point) * Correct alignment for odd size, even center column (+1 point) * Correct alignment for even size, odd center column (+1 point) * Correct alignment for even size, even center column (+1 point) * foodChain(1, 3, 1) (+1 point) * foodChain(1, 9, 2) (+1 point) * foodChain(2, 10, 2) (+1 point) * minSize 1, random maxSize up to 20, random step between 1 and 4 (+1 point) | 8 points |
| **Total** | **15 points** |
| ***Coding Style*** | |
| Readability: Indentation.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 points |
| Readability: Meaningful method and variable names.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 points |
| Readability: Comments/Documentation.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 points |
| Efficiency: Code duplication.  Good: 2 points, Fair: 1 point, Poor: 0 point | 2 point |
| Efficiency: Smart algorithm.  Good: 2 point, Fair: 1 point, Poor: 0 point | 2 point |
| **Total** | **10 points** |