CS 145 Half-Homework 2

Routine Ops

1 Overview

Your objective in this homework is to make code self-contained and repeatable using methods. You will do this in the context of solving several disconnected problems that have no overarching story. Sorry.

Breaking code up into methods has several nice benefits: it enables large problems to be decomposed into smaller, more mind-sized bytes; small methods with a distinct purpose are easier to test than gangly spans of code; and because of the scope boundaries of methods, data becomes more insulated and less likely to get accidentally overwritten or inappropriately referenced.

Additionally, methods have an aesthetic quality that resonates with the human spirit: we want to make things that have lasting value. Methods can be easily shared with others and reused in other contexts. To connect to these other contexts, methods accept parameters and produce return values.

2 Requirements

Complete the four classes described below. Place all classes in package `hw2`. Make all methods `static`.

2.1 Main

Write a class `Main` with a `main` method, which you are encouraged to use to test your code. Nothing in particular is required of it, but it must exist.

2.2 ShapeUtilities

1. Write a method `getCircleArea` that accepts a `double` radius parameter. It returns the area of a circle with the given radius. Use `Math.PI`.
2. Write a method `getSquareWaste` that accepts a `double` length parameter. It returns the difference between the area of a square with the given side length and the area of the circle whose diameter is the given length. If you are manufacturing circles by cutting them out of squares, this difference is wasted material.
3. Write a method `getAnnulusArea` that accepts in the following order two `double` parameters: an outer radius and an inner radius. An annulus is a 2-D ring, a larger circle with a smaller concentric circle cut out. Don't repeat yourself. You may assume the outer radius is greater than the inner radius.

2.3 StringUtilities

1. Write a method `extractBracketed` that accepts a `String` parameter containing some text surrounded by square brackets. It returns the bracketed text, sans brackets. For exam-

ple, `extractBracketed("Respect [covers] the empty hole where love should be.")` → "covers". Assume the `String` contains exactly one pair of brackets.

2. Write a method `getThird` that accepts a `String` parameter of a comma-separated list. It returns the third item in the list. For example, `getThird("Alice,Bob,Carol,Dwight,Eve")` → "Carol". There are several ways to solve this problem. My personal favorite is to use a `Scanner`. Instead of constructing the `Scanner` to read from `System.in`, pass the list in instead. Then change the delimiter, which breaks words up on whitespace by default, using the `useDelimiter` method. Assume the list contains at least three items.

2.4 MathUtilities

1. Write a method `getHypotenuseDifference` that accepts as `double`s the two side lengths of a right triangle. Return the difference between the combined lengths of these sides and the hypotenuse that shortcuts them. For example, `getHypotenuseDifference(3, 4)` → 2.

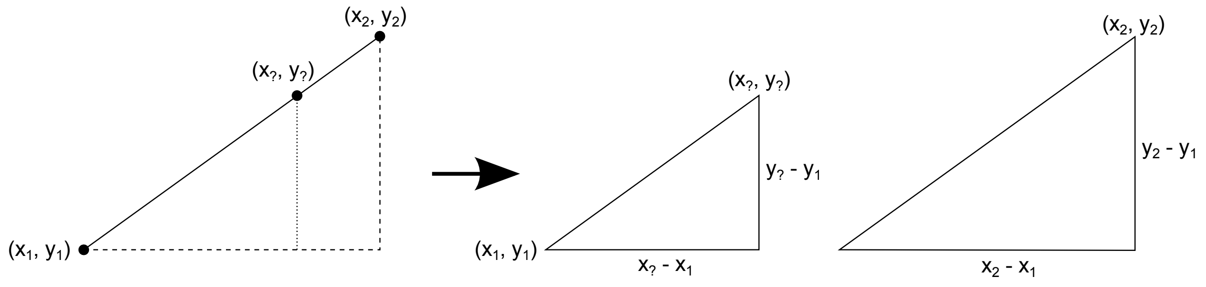
2. Write a method `lerp` that accepts five `double` parameters in the following order:

   (a) The x-coordinate of the first endpoint of a line segment.
   (b) The y-coordinate of the first endpoint of a line segment.
   (c) The x-coordinate of the second endpoint of a line segment.
   (d) The y-coordinate of the second endpoint of a line segment.
   (e) An x-coordinate whose corresponding y-coordinate is to be determined.

   This method considers a line segment spanning between the two known endpoints: (x_1, y_1) and (x_2, y_2). The third point lies somewhere on this segment. We know the point's x-coordinate, but it's up to us to calculate the y-coordinate.

   Using a straight line to place missing information like this is called linear interpolation, or lerp for short. Lerp routines are used heavily in computer graphics and scientific computing. When you scale up a photo, you might be using linear interpolation to fill in colors between the original pixels.

   One can use similar triangles to determine the missing y-coordinate, as shown in the figure below. The hypotenuse of the large triangle spans between the two endpoints, and the hypotenuse of the smaller spans from the first endpoint to the interpolated point. The lengths of the sides can be determined as the differences along the x- and y-axes:



Because the two triangles have the same proportions, we can craft an expression relating the sides:

$$\frac{x_? - x_1}{y_? - y_1} = \frac{x_2 - x_1}{y_2 - y_1}$$

All but one of these values is known. We solve for and return $y_?$.

3 Submission

To submit your work for grading:

1. Put the SpecChecker for this homework in your Build Path. Run the SpecChecker as a Java Application and fix problems until all tests pass.
2. Commit and push your work to your repository. Verify that your solution is on Bitbucket.

A passing SpecChecker does not guarantee you credit. Your grade is conditioned on a few things:

- You must meet the requirements described above. The SpecChecker checks some of them, but not all.
- You must not plagiarize. Write your own code. Talk about code with your classmates. Ask questions of your instructor or TA. Do not look at others' code. Do not ask questions specific to your homework anywhere online but Piazza. Your instructor employs a vast repertoire of tools to sniff out academic dishonesty, including: drones, CS 145 moles, and a piece of software called MOSS that rigorously compares your code to every other submission. You don't want to live in a world serviced by those who squeaked by through questionable means. For your future self, career, and family, do your own work.
- Your code must be submitted correctly and on time. Most excuses devolve into, "I started too late." The fix for this problem is not an extension.