Homework 4: Shape Array & Sort

Due date: September 21, 2015

For this homework, you will further modify the `Shape` class and subclasses from the previous homework assignment, and you will make a driver to instantiate and sort various shapes.

Specification

Shape class - You must modify the `Shape` class to be abstract and provide the following methods:

`public Shape()` - You can provide this via the default constructor, if you like.

```
public abstract double getArea()
public abstract double getPerimeter()
```

Modify the `Rectangle`, `Square`, `Circle`, `Octagon`, and `Hexagon` classes from the previous homework to provide the implementations of the abstract methods in the new `Shape` class.

When `toString()` is called on a subclass of `Shape`, it must return a `String` with the following format: `<shape> <area> <perimeter>`. Areas and perimeters must be printed with at least one digit to the left of the decimal point and exactly three digits to the right of the decimal point.

ShapeSortTest class - This class accepts from the command line a list of shapes, of any length, formatted as `<letter><size>`, where `<letter>` can be either `C` for circle, `S` for square, `H` for hexagon or `O` for octagon and `<size>` is an integer (but you could make doubles work if you like). For circle, size would be the radius, for square, octagon and hexagon, size is the length of a side. (We will not be testing `Rectangle` objects in `ShapeSortTest`.)

Here's a sample command line: `java ShapeSortTest C10 S85 H77 H19 C100 S12 O10`

The class stores references to these shapes in an array of `Shape` (similar to `StockHoldingTest2`) and sorts the objects in ascending order of area of `Shape` and prints information out in sorted order, one

shape per line using the `toString()` method. It then re-sorts the objects in descending order of perimeter of `Shape`, and prints them out again, in the same format. (You may add extra blank lines and whitespace for improved readability.)

Notes

- Be sure to use `Math.PI` in the computation of the area/perimeter, if necessary, so that your lower-order digits are correct.
- When sorting an array of objects you swap the positions of objects that are out of order. Note that you can swap the references to the objects. You don't need to swap (copy/change) the instance variables, themselves.
- Do not put your homework file(s) in a package.
- Do not include extraneous code in your Java files.

Turning-in Your Work

Place all your .java files into a ZIP file called `homework4.zip`. The following JAR command may be used to do this:

```
jar cvf homework4.zip *.java
```

Submit your `homework4.zip` file using AutoLab (http://autolab.cs.cmu.edu).

Grading

AutoLab will grade your assignment as follows:

- Java files exist and compile: 5 points
- `Shape` class is abstract and subclass updated accordingly: 10 points
- Test cases from Homework #3 still work: 10 points
- `ShapeSortTest` class tests: 50 points
- Follows coding conventions: 10 points
    - We'll deduct one point for each coding convention issue detected.
- Author tag in a JavaDoc comment at beginning of file: 5 points

AutoLab will show you the results of its grading within approximately one or two minutes of your submission. You may submit multiple times so as to correct any problems with your assignment. Autolab uses the last submission as your grade.