COMS W1007 Homework 3

Submission instructions

All programs must compile and run on CUNIX to receive credit. Submit your electronic files via `http://courseworks.columbia.edu`. Post your archived submission directory to the CLASS FILES section in the appropriate homework directory. We prefer electronic submission of written portions, though you will not be penalized for paper submissions. (Do **not** print out your programs.) Include a separate README text file that explains exactly what each file in your submission is. Your submission should contain your code (.java files), written problems, and your readme file. Do not include your compiled .class files or your compiled javadoc, since we will generate those from your code. Place all the files you want to submit into a submission directory with the following naming scheme.

`<your_uni>_hw<number>`

So if my UNI is `uni1234` am submitting homework 5, my directory would be `uni1234_hw5`. Archive your submission directory using zip, tar, or gzip format. In CUNIX, you can tar and gzip with the following command:

```
tar -czvf uni1234_hw5.tar.gz uni1234_hw5
```

and upload `uni1234_hw5.tar.gz` to courseworks. (You will probably need to first download the file to a local directory using an FTP program. See CUNIX tutorial for more info.)

Multiple Submissions: You can submit multiple times, but we will only consider the latest submission based on the timestamp in courseworks. Please give at least 1-2 minutes between two submissions so we can tell which is the newest submission.

Keep a pristine copy of your submission folder in case there is any submission error.

Written Problems

1. **6 points** (Horstmann 6.12) Consider the `Number` class in the standard Java library.
   (a) What are its subclasses?
   (b) Why are the methods `byteValue` and `shortValue` *not* abstract? (Note that all other methods are abstract.)

2. **6 points** (Based on Horstmann 6.22) In Chapter 6, Horstmann criticizes a design in which classes `Circle` and `Rectangle` extended a class `Point`. Describe a better design in which the `Circle` and `Rectangle` classes have a common supertype `Shape`. Should `Shape` be an interface type or an abstract class? What methods should `Shape` provide? How will `Circle` and `Rectangle` differ?

3. **6 points** (Horstmann 7.6) What Java code do you use to test

   (a) Whether `x` belongs to the `Rectangle` class?

   (b) Whether `x` belongs to a subclass of the `JPanel` class (but not the `JPanel` class itself)?

   (c) Whether the class of `x` implements the `Cloneable` interface type?

4. **6 points** (Horstmann 7.12) Write a method `dumpArray` that prints the elements of *any* array to `System.out`, using `toString` on the array elements if the array elements are objects.

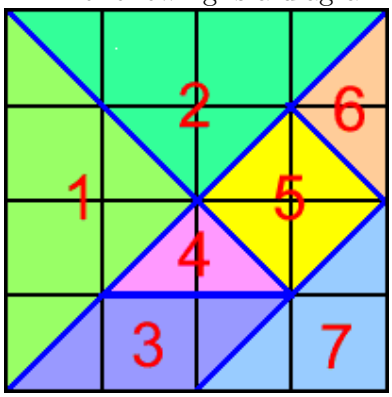
5. **6 points** (Based on Horstmann 7.20) Consider the following approach to cloning. Using serialization, save an object to a stream and read it back. You get a new object that appears to be a clone of the original, because all of its instance fields are distinct. Give two limitations of this approach.

Programming Assignment

(70 points) For this homework assignment, you will build a graphical tangram puzzle. You must build this program using your own abstract class.

Description

A tangram is a puzzle consisting of simple shapes, which are easy to draw using AWT. For this assignment, you will build a program that lets users drag and drop the tangram shapes to move them. The following is a diagram of the standard tangram solution[1].



Since dynamic rotation is tricky, for the assignment you are only required to allow the user to move the pieces up, down, left and right, and you should draw the shapes in the correct rotation. The dimensions of the shapes are proportional to the following:

1. Right triangle, hypotenuse facing left. Base length $1/\sqrt{2}$, hypotenuse length 1.
2. Right triangle, hypotenuse facing up. Base length $1/\sqrt{2}$, hypotenuse length 1.
3. Parallelogram, base length 1/2, height 1/4, offset between top edge and bottom base 1/4.
4. Right triangle, hypotenuse facing down. Base length $\sqrt{2}/4$, hypotenuse length 1/2.

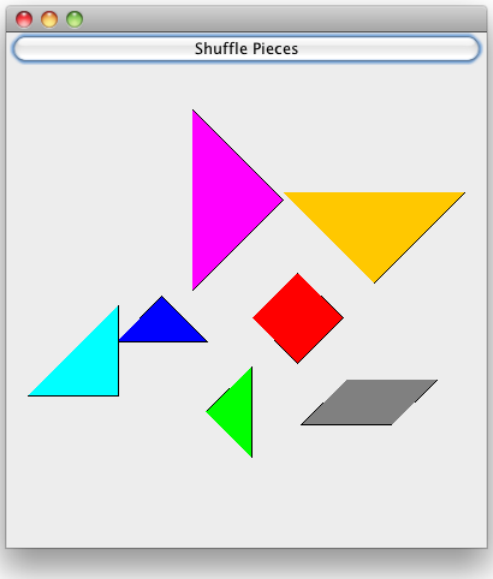
[1]Image from http://www.techitoutuk.com/projects/tangrams/dimensions.html

5. Diamond, side length $\sqrt{2}/4$. Maximum width and height $1/2$.

6. Right triangle, hypotenuse facing right. Base length $\sqrt{2}/4$, hypotenuse length $1/2$.

7. Right triangle, hypotenuse facing northwest. Base length $1/2$, hypotenuse length $1/\sqrt{2}$.

Your program should initialize with the shapes randomly scattered on a window (for simplicity, it is okay if they overlap). There should be a button on the window to shuffle the pieces. The user can then drag each piece to move it anywhere. You should **not** check for the right solution or do any collision checking (this would make the assignment much harder than I intend). The following is a screenshot of a complete assignment.



Your program design should include an model-view-controller structure. It must use an abstract class to handle the basic functionality of all shapes, and have concrete classes for each shape type. You will need to use MouseListener and MouseMotionListener objects.

Process and Deliverables

For this programming assignment, you are only required to submit code that compiles via javac and javadoc. At least all public methods must have javadoc comments for any parameters and return values. You should include your code and a "readme" file that briefly describes each included file, and can be used to indicate to us any notes that would be helpful for grading (such as any extra functionality you have added or were unable to finish in time).

Nevertheless, you are encouraged to use the design tools we practiced on previous homework, but you do not need to submit them.

As always, to get full credit, your code must not crash for any reasonable user behavior. You may exit with an informative error message, but uncaught exceptions or crashes will lose some points. (There are not many, if any, opportunities for the user to cause exceptions in this assignment, so this should be easy.)

Notes and hints

- It may help to review basic trigonometry to write a general right triangle class. You can use `Math.sin()` and `Math.cos()` to compute where the three points should be drawn given an angle of rotation. This way you do not need to write a different triangle class for each rotation.
- Look at Horstmann's Scene Editor program in Chapter 6 for an example of how to set up dragging and dropping of freely drawn graphics. While Horstmann creates a `SelectableShape` abstract class, we are less interested in selection and more in just the drag and drop behavior. You are free to build off his code, but you must cite the source. Note that his code does not follow a MVC structure.
- It is probably easiest to use the `Polygon` class for each of these shapes. This will allow you to draw the outline, fill the shape and use the convenient `contains()` method to check when mouse clicks are inside the shape.
- You may have, in your main function, hard-coded shape configurations. For example, your main method can initialize each of the seven shapes and add it to whichever object you use to store your shapes.
- As usual, resist the temptation to add extra features until you have finished the required basic features. With good design, adding extra features will be easy once you finish the basic requirements, but since you will not receive credit (other than your own enjoyment) for extra features, make sure you finish the bare minimum first.