Program #2 - Shape factory

Let's make some shapes

Logistics:

- Due: **Mon May 5, 2014**
- Worth: 10 points

1. Description

We have two interfaces:

- `Shape` - a circle or square or whatever… with an area and center and bounding box
- `ShapeFactory` - a guy who knows how to create a given Shape

I'll place these on the k: drive.

Program #2 has two major components:

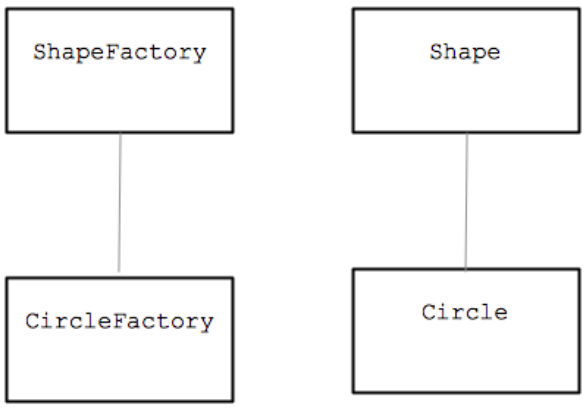
1. A viewer so that you can see and select your shapes
2. A console-based command line so that you can add shapes and other things

Class structure

When implementing an actual shape, like a circle, the following relationships exist:

- `Circle` is-a `Shape`
- `CircleFactory` is-a `ShapeFactory`

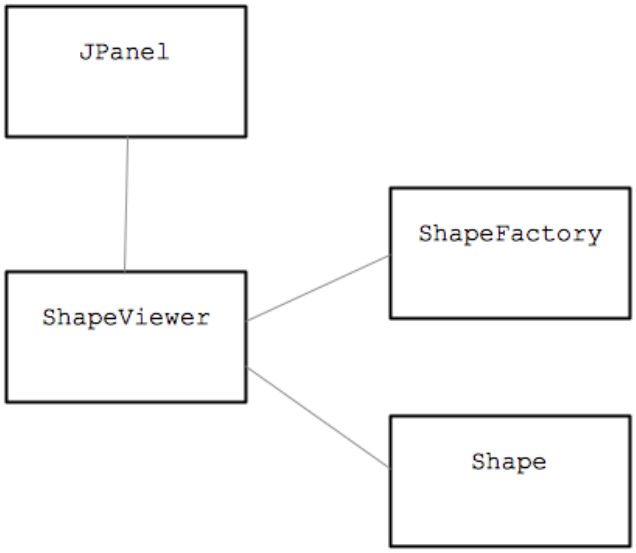
Here's the UML:



Your `ShapeViewer` can create and view shapes. The relationships are:

- `ShapeViewer` is-a `JPanel` (override the `paintComponent()` method)
- `ShapeViewer` has-a `ShapeFactory` (so that it can create new shapes)
- `ShapeViewer` has-a `Shape` (a list of shapes to draw)

The UML:



ShapeViewer

There should be no code specific to any concrete shape in your `ShapeViewer`. You'll need a list of factories for the viewer to use and a list of shapes that need to be drawn. Override the `paintComponent()` method to draw your shapes:

```
public void paintComponent( Graphics g) {
   super.paintComponent( g);   // always

   for each shape in viewer, draw it
}
```

Allow the user to select a shape. This will requires a `MouseListener` (or adapter).

Console commands

The console will be used to send commands to the viewer.

Commands include:

- add [count] [shape] [color] [name] - add shapes to the viewer
- list [shape]
- area [shape]
- clear
- exit

By default, shapes will have random size, color and location. We'll flesh out the these commands more in class.

2. Implementation

Plan. Step-by-step. Slics and dice. Go!

1. Create a `main()`, a frame and empty `ShapeViewer`
2. Create one concrete shape: `Circle` and `CircleFactory`
3. Create a circle in your code, add it to the viewer and draw it. Then two circles, etc.
4. Try another shape.
5. Create your console command loop to add shapes from your console
6. Add a mouse listener to your viewer to select a shape

There are more steps, but when you are able (and confident) to add one shape the others will be gravy.

Pseudo-code for your Program #2 main() might look something like this:

```
main() {
   Create a frame
   Create a ShapeViewer, add it to the frame
   Create shape factories, add each to the ShapeViewer
   Show the frame
   Create a console command and run it
}
```

Implement the shapes: Circle, Oval, Rectangle, Square, and Line. Of course, each shape will require a corresponding factory to create it.

Extras!

- Implement one additional shape of your own. As a final step, we'll share our individual shape and factory, using the k: drive.
- Define and add your own command to the console command loop

I will provide you with some help in generating these random things: locations, sizes, colors, and names.

3. Grading

Please place the following in your `program2/` k: drive folder:

- A `README` file describing the state of your program
- Your beautiful Java code that follows our Java Coding Guidelines
- Include any images you would like me to look at. Explain them in your `README`.