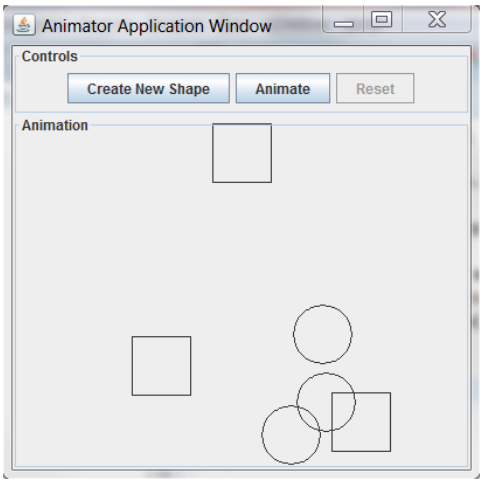
# CSSE 374: Lab 7-2

## Background

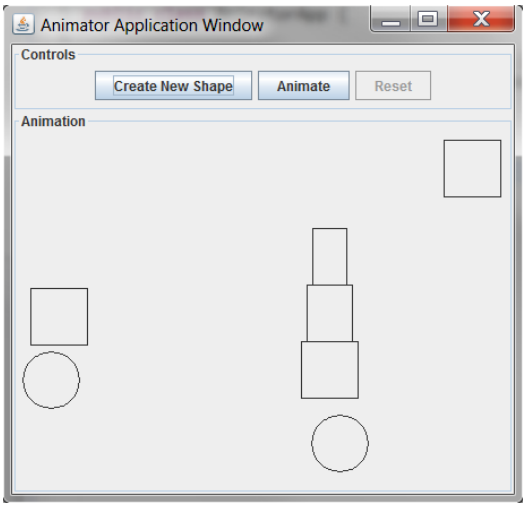
Assume you want to build the world's best arcade game in Java. You are developing a necessary infrastructure to make the game possible. Here is what you have so far:




You can create two basic shapes on the animation panel: rectangle and circle. These shapes are automatically created at random locations on the screen when user presses the **Create New Shape** button. When the user presses the **Animate** button, these shapes starts moving and bouncing off the wall. A user can press **Create New Shape** while the animation is in progress. To reset everything, the user just has to press the **Reset** button. (Please try out this pseudo-game to see how it works.)

## Feature

Since we want to build the world's best arcade game, we need to do more than these basic shapes. You came up with the following neat idea: "How about we combine these basic shapes to create more complex and meaningful shapes? The framework can be programmed to treat these basic shapes and the complex shapes uniformly allowing many new shapes in the future." (See the tower shape below for an example of complex shapes.)



As a proof of concept, you only need to **implement two different complex shapes** utilizing the rectangle and circle sprites/shapes, however you want it. You are now dying to implement this new feature in the framework, aren't you? ☺ As you have a deep understanding of design patterns, you challenged yourself to implement this new feature working around the following constraints:

1. I will not modify **problem.client** package.

2. I will not modify **problem.graphics.MainWindow** as the logic of shape creation is nicely encapsulated within the **problem.sprites** package.

3. I will have to modify the **paintComponent()** method of **problem.graphics.AnimationPanel** as it is using **ArrayList**'s iterator in the enhanced for-loop to traverse the sprites. I should be able to handle the iteration for complex sprites/shapes just by adding no more than **five** extra lines of code including brackets, but without any other updates to this class.

4. I need to modify some of the existing classes and create some more new classes to implement the new feature in the **problem.sprites** package.

5. None of the class in the **problem.sprites** package use Graphics, Graphics2D, Component or any rendering-related code. I will maintain this separation of concern in the new implementation as well.

## Design

Create **Lab7-2/docs/Answer.pdf** with answers to the following problems:

**Q1.** Create a **UML Class Diagram** to present your new design idea. Explain the design in a few lines. [**10 points**]

## Implementation

**Q2**. Implement your code satisfying all of the constraints imposed above in the appropriate subpackages of the **Lab7-2/src/problem** package. It is ok if your complex shapes deform after bouncing off the walls. [**35 points**]

## Testing

**Q3:** Add a couple of GUI snapshots to **docs/Answer.pdf** capturing the movement of the objects on the screen. [**5 points**]

## Deliverable

Bundle your project in the **zip** format [**not rar**] and turn it in on Moodle.