# Lots and Lots of Shapes
## CSC211 Lab 09

## Lab 09 Set-Up

This lab builds on the work you did in **Lab 07**. You and your partner can discuss which of your two **Lab 07** solutions to use as a starting point for this lab. (You can also use your **Lab 06** code for this project.)

The purpose of this lab is to give you additional experience with polymorphism and inheritance and to introduce you to the use of arrays. You'll create and animate an array comprised of random `Ball` and `Block` objects.

This lab is comprised of the following classes:

- `CSC211Lab09` - the `main()` class
- `Lab09Panel` - the panel class where most of the work is done
- `DrawableInterface` - this class should not need to be modified
- `Ball` - the default constructor for this class will be modified
- `Block` - the default constructor for this class will be modified

## `CSC211Lab09.java` Requirements (5 pts)

This class is the `main()` class for the lab. It should do the following:

- Use a `JOptionPane` to ask the user for the number of shapes to animate
- Create a GUI window (`Frame`)
- Instantiate a `Lab08Panel` object and send the constructor the number of objects to create for the panel
- Add the panel to the frame
- Call the panel's `animateShapes` method to animate the shapes

## `Ball.java` and `Block.java` Requirements (10 pts each)

Change the default constructor for each of these two classes to meet the following requirements:

- Starting position is randomized between `(0,0)` and `(100, 100)`
- Starting `xVelocity` and `yVelocity` are randomized and range from `1` to `10`
- The ball's `radius` or the block's `side` are randomized with values between `1` and `15`
- The `color` is randomized. You can use a random integer from `0` to `Integer.MAX_VALUE` as an argument to the `Color` constructor to generate a random color. (*Why does this work?*)

## `Lab09Panel.java` Requirements (45 pts Total)

This class is where most of the work for this lab occurs. Here are the class's requirements:

- Declare a data field, `shapes`, that is an array of `DrawableInterface.` This array will hold all of the shapes we will draw. **(3 pts)**
- Declare an integer constant `DEFAULT_SHAPES` that represents the default size (number of elements) of the `shapes` array. You can set this to any number, but it is recommended that it be less than 100 for testing purposes. **(2 pts)**
- Create two constructors for the `Lab09Panel` class. A default constructor **(10 pts)** and a parameterized constructor **(10 pts)** that takes a single integer as a parameter.
- In the default constructor, set `shapes` to be an array of `DEFAULT_SHAPES` elements.
- In the parameterized constructor, if the parameter is greater than `0`, set `shapes` to be an array of that many elements. Otherwise, set `shapes` to have `DEFAULT_SHAPES` elements.
- In each constructor, create a shape for each element in the array `shapes` by generating a random number. If the random number is `0`, the shape should be a `Ball`. If the random number is a `1`, the shape should be a `Block`.

### `paintComponent` METHOD (5 pts)

- This method should call the parent class `paintComponent` method .
- Then use a `for`-each loop to iterate through the array `shapes` and draw each.

### `animateShapes` METHOD (10 pts)

- This method should have a `while`-loop that loops for 20 seconds.
- The body of the `while`-loop should use a `for`-each loop to iterate through the array `shapes`. The body of the `for`-each loop should :
    - Move each shape
    - Check the shape's position to ensure it is still in the window.
- The body of the `while`-loop should repaint the panel and sleep to adjust the timing of the animation.

### `checkPosition` METHOD (5 pts)

This method ensures that the shapes stay within the window frame, *even when the frame is moved or resized*. The method should:

- Get the right and bottom coordinates of the panel
- Check that the shape is inside the panel by comparing the shape's x-coordinate to the left and right edge coordinates of the panel and the shape's y-coordinate to the top and bottom edge coordinates.

- If the shape is outside the panel,
    - move the shape back inside the panel
    - change the velocity for the direction that the shape had moved outside the panel