# COSC 102 Spring 2015

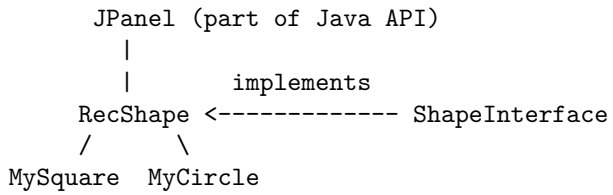
# Graphics Polymorphism & ArrayList Lab 6

In this lab you will work on the following concepts:

- inheritance and interface using Java `Graphics` classes
- polymorphism
- collection: changing an array implementation to use Java's `ArrayList` linear structure.

### Overview

The provided code creates the following class hierarchy.




`Experiment.java` is the client/driver class that builds a window with a drawing area maintained by the `DrawingCanvas` class. `DrawingCanvas` keeps the information of `RecShape` objects, which are drawn and manipulated within it.

- An `Experiment` object **has-a** `DrawingCanvas` object.
- A `DrawingCanvas` object will **have-a** *collection* of `RecShape` references (each element of which either instantiated as a `MySquare` object or a `MyCircle` object)
  - In Part I: this collection is an array of `RecShape`
  - In Part II: this collection is an `ArrayList` of `RecShape`

Most of the code you will add is in `DrawingCanvas.java`.

## I. Array Implementation

The folder `codeDemo` contains our solution to Part I with `ExperimentPartI.java` and Java executable for the other classes. Open `ExperimentPartI.java`, run it and explore the behaviors (using the mouse and when a shape is selected push the keys `=`, `-`, `t`) which you are expected to implement in the provided code stored in the folder `codePart1`.

Then, run this provided code `codePart1/Experiment.java`, most of the functionality for manipulating `RecShape` is implemented as you can see by interacting with the two hard-coded shapes.

Your first task is to complete and to correct the available manipulations.

- The circle can't be dragged for now. Read the code, identify how the rectangle works and make the appropriate change so that the circle can also be dragged.
- The circle can be resized (try the key `=` and `-`) but the rectangle can't. Fix it.

- This resizing for both shape isn't great: the scale invariant point is the left top corner, while it is more natural to be the center of the shape. Identify which of the provided class is responsible of this behavior, read the hint there, you might have to make a diagram and read a bit of the API to find what you want.

You observed in the provided demo `ExperimentPartI.java` that the key `t` on a selected shape changes the actual instantiated object: if it was a circle, it becomes a rectangle and vice-versa.

Complete the particular constructor of `MySquare` and `MyCircle` to make it possible and don't forget to write the `switchSelectedShape` method in `DrawingCanvas`.

When you have all these shape functionalities working, you are ready to use an array of `RecShape` instead of having two hard-coded lonely shapes... You should only change `DrawingCanvas.java` to implement it.

- Write the `initArray` method to instantiate and initialize an array of `numOfShapes` where half of the elements are `MyCircle` objects and the other are `MySquare` objects.
- Update the appropriate methods in `DrawingCanvas` so that the array is considered for each interaction events and drawing calls instead of the two hard-coded shape

## Transition: Readings

Next you will create a new version of the program in which the `DrawingCanvas` class uses a Java's `ArrayList` to store the shapes. Doing so will permit to add a couple of new features taking advantage of `ArrayList` automatic expansion and internal storage.

But first, it is essential you learn how `ArrayList`s work and how to use them with generics. Once you understand the `ArrayList` API coding Part II should not be long.

- Read the following tutorial until Section 15 and answer the question on each page before advancing. Try to do it fast, i.e. in 20 mins by spending around one minute by page. Ask us if you do not understand.

## II. ArrayList Implementation

Make a copy of your folder `codePart1` and rename the copy `codePart2`. Work on these files, keeping the one within `codePart1` untouched.

The folder `codeArrayListDemo` contains our solution to Part II with `ExperimentPartII.java` and Java executable for the other classes. Open `ExperimentPartII.java`, run it and explore the behaviors (as you did before and also try `a` and `d` when a shape is selected). Pay attention to the differences:

- what is happening when you are releasing a shape?
- `a` adds a new random shape
- `d` deletes a selected shape

Change `DrawingCanvas` such that an `ArrayList` is used to store the `RecShape` objects and modify the code so that it first all works as before and then that the additional behaviors highlighted above are available.

## Submit

**Show the lab instructor the work you completed before leaving the lab.**

Submit on Moodle by the deadline a `zip` file called `lab06` containing your two folders (You should delete the *.class* files before creating the `zip` so it is a smaller file).