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, in the part1 folder, creates the following class hierarchy.

```
   JPanel (part of Java API)
   |________________________|
   |_______________________|
   RecShape <-------------- ShapeInterface
   "----------------------"  /  \
   MySquare       MyCircle
```

In addition,

- the class DrawingCanvas permits to maintain a collection of RecShape objects, which are drawn and manipulated within a graphical window and
- Experiment.java is the client/driver class that builds a window which drawing is provided by a DrawingCanvas instance.

Skim through the provided code so to reply the following questions. Please write your answers on the print-out:

- In which class and on which line is the DrawingCanvas instance created? #____
  Is it a local or instance variable? ______________________________

- The Experiment class _______-a [is/has] DrawingCanvas object.

- A DrawingCanvas object has for now two hard-coded RecShape objects. Give their names and their instantiated types; where are they declared and in which method are they instantiated

  RecShape 1: name/type ______/________ RecShape 2: name/type ______/________
  declaration/instantiation    #_____/#______    #_____/#______
  instantiation method name __________________________________________________
  Where does this method get called? ______________________________

- DrawingCanvas as you will develop it in the lab will have-a collection of RecShape references (each element of which either instantiated as a MySquare object or a MyCircle object). Read all the comments of the DrawingCanvas class to determine the data type of the collection you will implement in each part.

  - For Part I: the collection will be ___________ of RecShape
  - For Part II: the collection will be ___________ of RecShape

Most of the code you add will be in DrawingCanvas.java.
Part I. Array Implementation

The folder demoPart1 contains our running solution with ExperimentPartI.java the client accessing the .class files provided. Open ExperimentPartI.java, run it and explore the behaviors with the following interactions

- click on shape
- move the mouse
- push the keys =, -, t.

In this part you replicate this demo starting with the code provided in the part1 folder. Work through the following step-by-step.

1. Behaviour for Hard-coded Shapes

Run the starting code 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.

1. 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.
2. The circle can be resized (try the key = and -) but the rectangle can’t. Use the same strategy than above to enable this behaviour.
3. 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, make a diagram and read a bit of the Java API to find what you want.

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

Read the switchSelectedShape method in DrawingCanvas and the relevant constructors of MySquare and MyCircle. Does the running of your program make sense with regard to the code? Understand the missing parts before starting to complete this feature that switch between shapes. Ask us if you are not sure. You have to add code to switchSelectedShape and to the constructors.

2. Array

When you have all the shape functionalities working, you are ready to create a collection of RecShapes. Replace the two hard-coded lonely shapes with an array of RecShape. You should only change DrawingCanvas.java to implement it.

- Comment out all the code pieces that are dealing with the hard-coded square and circle. Think about each piece carefully as you will have to go back and add the equivalent for the array structure.
- Complete 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.
- Complete the part of paintComponent so you can see the result of your array creation.
- Update all the methods in DrawingCanvas and DrawingCanvasMouseListener so that the array is considered for each interaction event in place of the hard-coded shapes you had.
Transition: Readings

For next week 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.

First you need to learn how ArrayLists 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.

Part II. ArrayList Implementation

Make a copy of your folder of part1 and call it part2: work on this version, keeping part1 untouched.

The folder demoPart2 contains our solution with ExperimentPartII.java running 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 and write down what is happening for each interaction:

- pressing a __________ a new random shape
- pressing d __________ a selected shape
- what is happening when you are releasing a shape? where is it drawn?

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 tonight a zip file called lab06 containing at least Part 1. You should delete the .class files before creating the zip so it is a smaller file.

Part 2 is due the day of your lab next week by 11:55 p.m. If you have completed by the start of lab time next week, please demo it to the instructor. If not there may be some time to work on it at the end of the lab, once you completed lab 7 worksheets.