Langara
THE COLLEGE OF HIGHER LEARNING.

Department of Computing Science & Information Systems
CPSC 1181
Lab#4
October 1, 2019

Objectives:

Inheritance
Animation

Preparation:

Study class notes and example of section Inheritance
Optional: Study chapter 9 of your text book
Review sections 2.9, 2.10, and 3.8 from your text book.

Due date:

Due Date: 11:00 PM on Monday October 7, 2019

Where to upload:

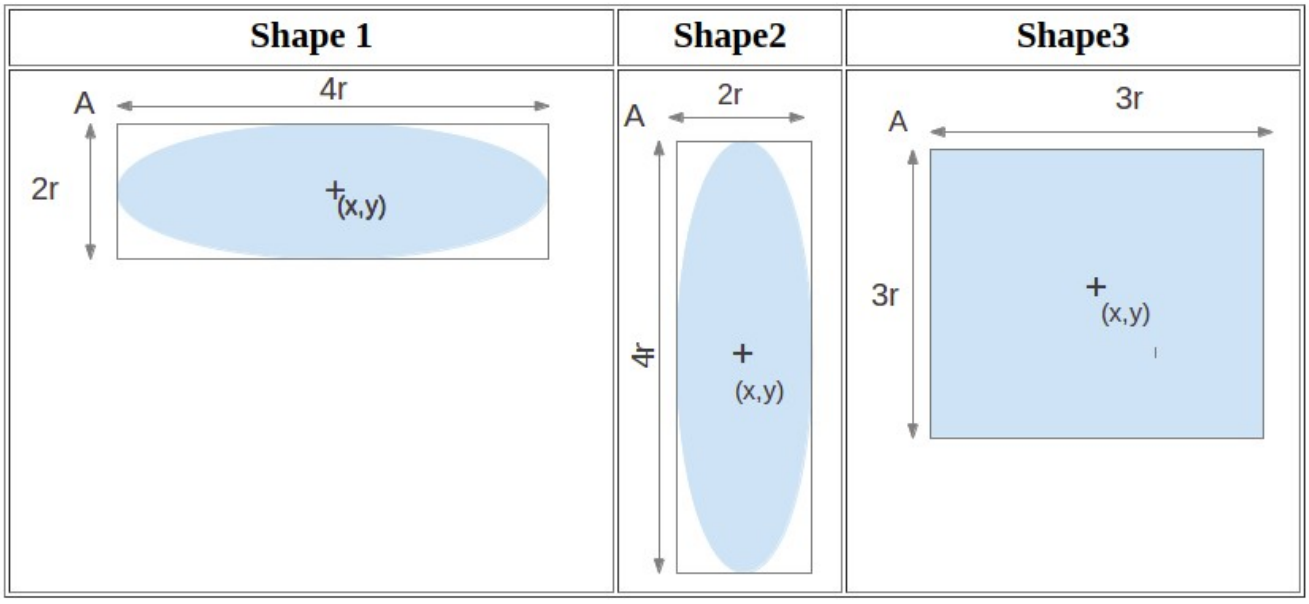
Zip your files into yourstudentID.zip where yourstudentID is your student number, and upload it to dropbox lab3 in D2L.

Part A: Tutorial

Do tutorial4
You should finish tutorial and upload your files by 17:00 today.

Part B:

This assignment is based on three basic shapes that follows:



Each shape is identified by its center coordinates (x, y) and a parameter r.

use Ellipse2D.Double and Rectangle2D.Double to create the shapes.

The top left coordinates of each shape is calculated as follows:

Shape1:
x.top : x - 2r
y.top : y - r
width : 4 r
height: 2r

Shape2:
x.top : x - r
y.top : y - 2r
width : 2r
height: 4r

Shape3:
x.top : x - 1.5r
y.top : y - 1.5r
width : 3r
height: 3r

Phase 1: [35 marks]
B1)
Download the lab4.java and modify it to finish your lab assignment.
Develop class Shape1:

```
public class Shape1 {
    private double x, y, r;
    private Color col;

    // add constructor, and required methods.


    public void draw(Graphics2D g2){
      // create a horizontal ellipse and then draw it

    }


}
```

- You should declare all instance variables as private, unless they are final.
- Create appropriate constructors, and methods.
- Object of class shape1 should assign a random Color (col) to itself during construction.
- By calling draw method of the object, it should create and draw a filled horizontal ellipse centered at (x,y), horizontal diameter equal to 4r, and vertical diameter equal to 2r.

- Add more methods as required.

```
// sample use casse:
Shape1 test1 = new Shape1(100,100,30);
test1.draw(g2);
```

Sample output of this part:

Test your class before continue.

B2)

Develop class Shape2 by extending class Shape1 :

```
public class Shape2 extends Shape1{

    @Override
    public void draw(Graphics2D g2){
      // create and draw a vertical ellipse, and then invoke the superclass
      // draw(...) method to draw the horizontal ellipse.


    }
}
```

- Develop appropriate constructors, methods, and instance fields.
- Do not shadow the instance fields of the superclass.
- Override draw(...) method of the superclass. This method should only draw a vertical ellipse using the same color and coordinates set in the superclass, and then it should call the superclass draw(...) method to draw the horizontal ellipse.

What method(s) should you add to the superclass to access its private instance fields?

```
// sample use casse:
Shape2 test2 = new Shape2(100,100,30);
test2.draw(g2);
```

Sample output of this part:

Test your class before continue.

Note that in this part the draw method of Shape2 should only draw the vertical ellipse, and then ask the superclass to draw the horizontal one.

B3)

Develop class Shape3 by extending class Shape2 :

```
public class Shape3 extends Shape2{

    @Override
    public void draw(Graphics2D g2){
        // create and draw a square then invoke the super class draw(...)
        // method to draw the vertical and horizontal ellipse.


    }

}
```

- Develop appropriate constructors, methods, and instance fields.
- Do not shadow the instance fields of the superclass.
- Override draw method of the superclass. The method should only draw a square using the same color and coordinate set in the superclass, and then it should call the superclass draw(...) method to draw rest of the shape.

```
// sample use casse:
Shape3 test3 = new Shape3(100,100,30);
```

test3.draw(g2);

The sample output :

Test your class before continue.

Note that in this part the draw method of Shape3 should only draw a square, and then invoke the superclass draw(...) method to draw the rest of the shape.

Develop a JFrame, JComponent, and create multiple instance of Shape3 as your test case.

Phase 2: [15 marks]

Create class Shape3Animated by extending Shape3 :

```
public class Shape3Animated extends Shape3{

    private int direction; // a random direction.
    private int velocity;  // a random velocity in pixel
    // Add instance fields that required

    // moves the x or y coordinates of the shape based on direction
    // and velocity
    public void move(){

    }

   @Override
    public void draw(Graphics2D g2){



    }
}
```

direction is a variable assigns a direction to the object. You should decide and choose the appropriate value for directions.

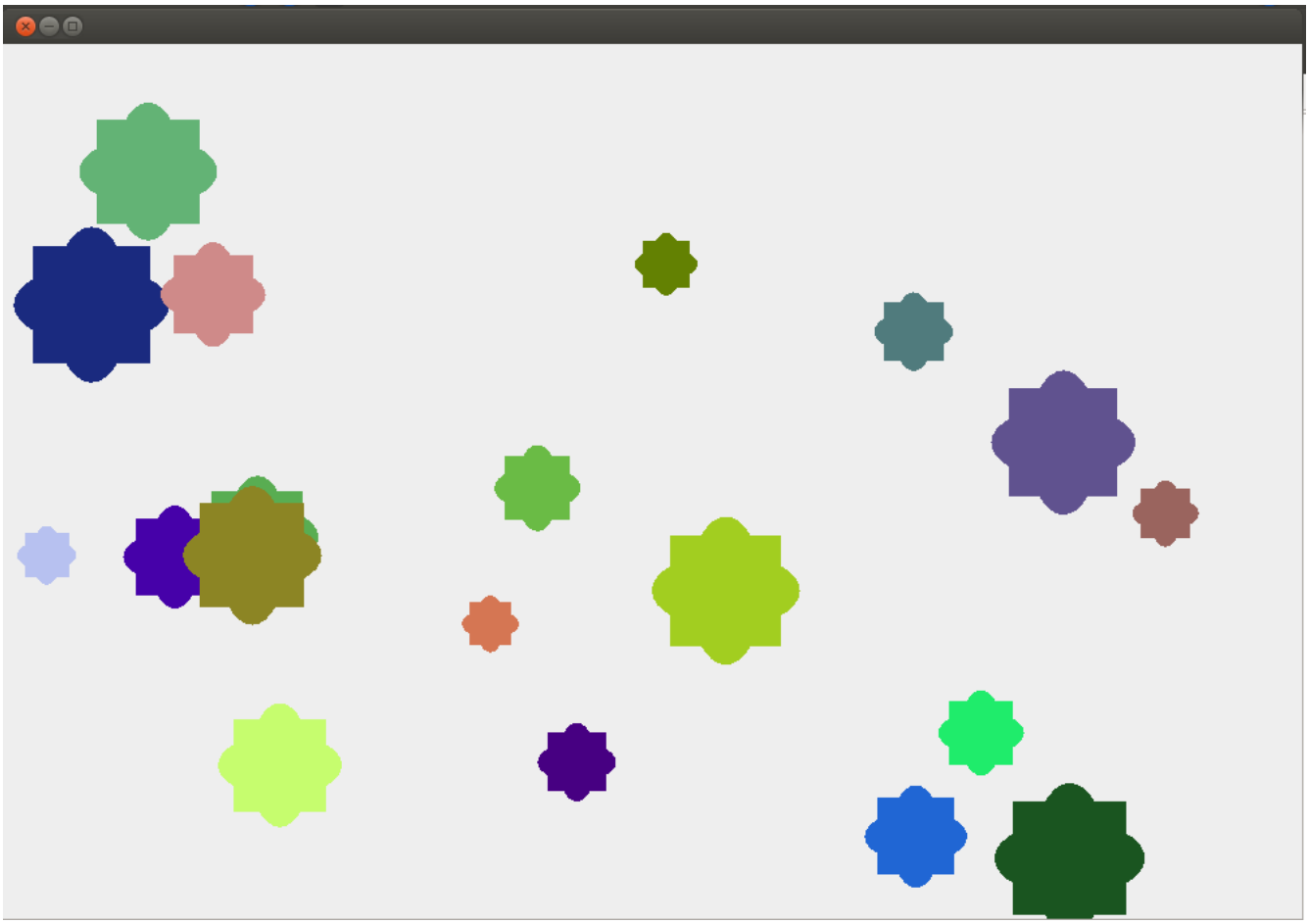
Note: You can also use enum structure for direction. Check the sample program, and check the google for more information in case you are interested.

velocity is the number of pixels object moves in each iteration.

Set your frame to 1000x 700 pixels.

Create an ArrayList<Shape3Animated> in ShapeComponent class, and create and add 10 to 20 instance of Shape3Animated with random position (x,y), random r in range (10<r<30), random velocity (1< v< 10), and random direction, and draw them on screen.

The output of your program at this point should look like something as shown below:



Test your class before continue.

Phase 3: [20 marks] Animate the Objects

Study the following sample code first myTimer.zip, and then animate your objects with following specifications:

1. Move the objects in the specified direction (top, bottom, left, right) with the specified velocity. Objects should bounce back when they hit the boundary of the frames and move in opposite direction. .

2. Each object is assigned a random velocity and a random direction when they are created.

3. Assume that objects can pass over each other. (No crash)

Notes:

1. each method of the class should do a simple job (procedural abstraction).
2. classes should be cohesive. Create auxiliary classes in case you need.

What to upload

Create one page UML diagram of your design, and save it either as PDF or as word 2003 (JPEG Inserted into word document).
Zip your source files and UML diagram into yourStudentID.zip file and then upload it.

Bonus 1 [5 Marks] Assume that we can assign any direction to the objects, and they can move in any direction, and when they hit the boundary of the frame, they bounce and move in another direction. Note that an object bounces back with the same angle that it hits an obstacle.
Bonus 2 [5 Marks] Assume that objects cannot pass over each other, and they bounce back in opposite direction when they hit each other.

TOTAL MARK: 70 +10 Bonus