Programming Assignment 4 (100 Points)

Due: 11:59pm Thursday, October 30

START EARLY!!

README (10 points)

You are required to provide a text file named **README,** NOT Readme.txt, README.pdf, or README.doc, with your assignment in your pa4 directory. There should be no file extension after the file name "**README**". Your README should include the following sections:

Program Description (3 points) : Provide a high level description of what your program does and how you can interact with it. Make this explanation such that your grandmother or uncle or someone you know who has **no programming experience** can understand what this program does and how to use it.

Write your READMEs as if it was intended for a 5 year old. Do not assume your reader is a computer science major. **The more detailed the explanation, the more points you will receive.**

Short Response (7 points) : Answer the following questions:

Unix Questions

1. From your current directory how do you copy over a java file named fubar from a folder three directories above? Write the full command required to perform this action.
2. What does the command "man cat" do (with no quotes)?
3. From your current directory, how do you remove all html files in your home directory?

Vim Questions

4. What is the difference between :q and :q! when using the commands to close a vim file?
5. What does the command "G" do (with no quotes)?

Java Questions

6. Name the 2 requirements for a constructor signature.
7. What is overloading?

STYLE (20 points)

Please see previous programming assignments for full details.

You will be specifically graded on commenting, file headers, class and method headers, meaningful variable names, sufficient use of blank lines, not using more than 80 characters on a line, perfect indentation, no magic numbers/hard-coded numbers other than zero, and use of accessor/mutator methods to access any private data fields (getters and setters) ONLY when specified in the write-up.

CORRECTNESS (70 points)

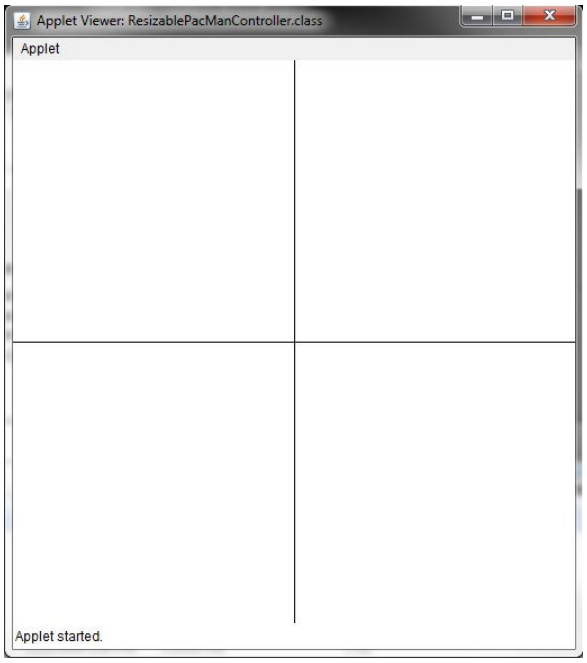
Setting up your pa4 directory:

Please see previous programming assignments.

You will need the **objectdraw** library for this assignment.

Start-up code containing constants to use for defining your ResizablePacMan class:

```
import java.awt.Color;
import objectdraw.*;

public class ResizablePacMan extends ActiveObject
{
  private static final double PAUSE_TIME = 33;
  private static final double GROWTH = 2;
  private static final double ARC_ANGLE = 300;
  private static final double MAX_DEGREES = 360;

  private final int CHANGE_ANGLE = 7;
  private int newStartAngle = 0;

  private FilledArc pacMan;
  private double startSize;
  private Location center;

  private double oldVLineX;
  private double oldHLineY;

  private Line vLine;
  private Line hLine;
    …

}
```

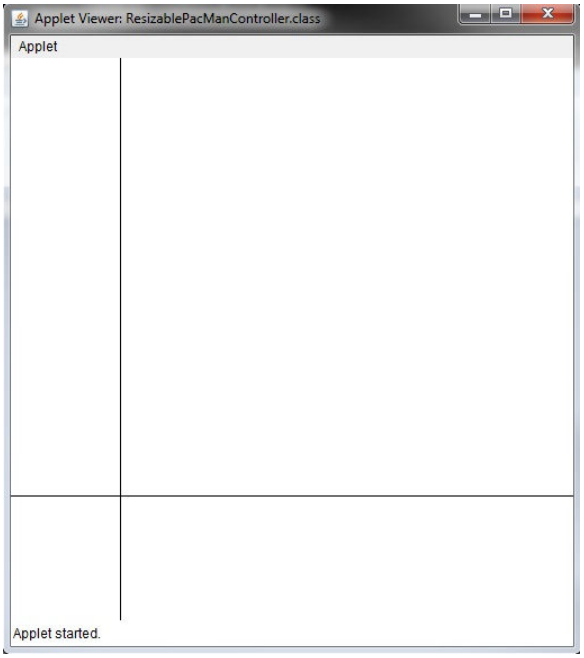


Stage 1: Creating Quadrants

Begin with the controller class **ResizablePacManController** (ResizablePacManController.java) by creating two Line objects to divide the canvas into four quadrants of equal size (a horizontal line and

a vertical line). The end points of the lines should be based on the current size (width and height) of the canvas.

Stage 2: Manipulating Quadrants



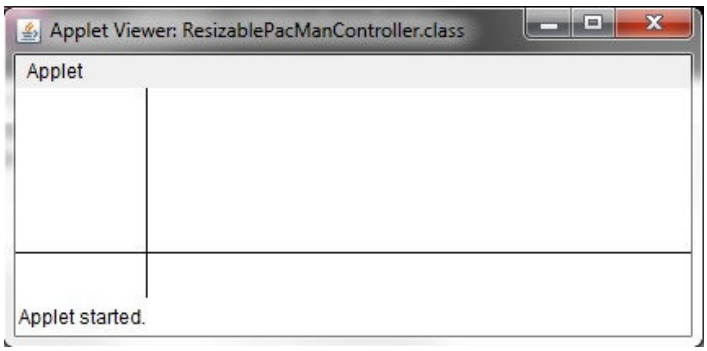
The first manipulation you should implement is dragging the Lines. Check in the onMousePress() method to determine whether one or both of the Lines have been grabbed or not. You can set boolean flags in this method to indicate which line(s) have been selected. Note that if you grab the intersection of the two Lines, you should be able to drag both lines simultaneously. Refer to the objectdraw documentation (linked from the Useful Links page) for other mouse event methods that may be useful. You will need to update the position of the Lines as they are dragged. You should include logic to make sure that the Lines are not dragged off the visible canvas/applet area. So do not let either line go beyond 5 pixels from the edge of the canvas in any direction (a 5 pixel margin).

The second manipulation you should implement is keeping the Lines proportional when resizing the window. If you resize the window, the Lines should move to preserve their current proportions on the screen. To redraw the canvas, override the paint() method:

```
public void paint( java.awt.Graphics g )
```

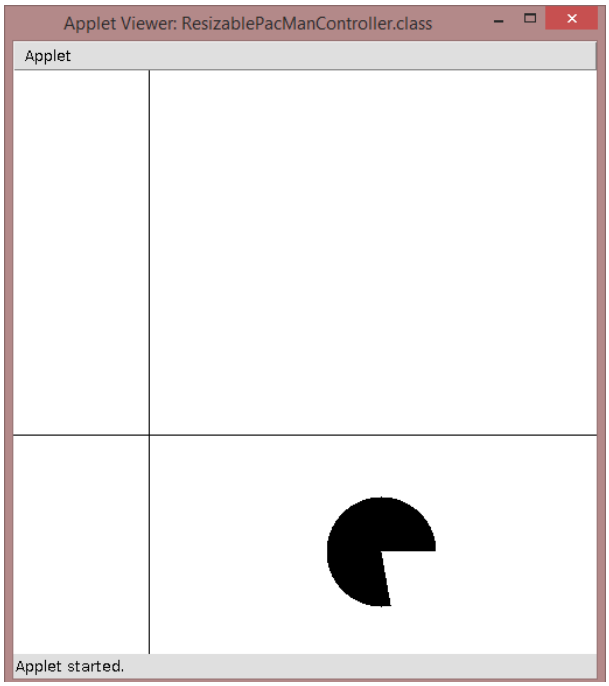
You should **first** make a call to the superclass's version of the method by adding the following as the first line of code in your paint() method:

```
super.paint( g );
```



You can then add in your code for repositioning the Lines based on the proportions of where they were previously, meaning that you need to save those proportions in two *double* variables. Make sure that you test this in appletviewer and not in a browser. When using a browser, you cannot resize the canvas itself.

Stage 3: Creating the PacMen



You should create a new class **ResizablePacMan** (ResizablePacMan.java), which should be an ActiveObject (extends ActiveObject - see Ch 9), to create each of the PacMen. The controller class should not have any references to any of the PacMen created. Instead the controller should simply create a new ResizablePacMan every time the mouse is clicked in the canvas/applet. The constructor should look exactly like this:

```
public ResizablePacMan( Location center, double size, DrawingCanvas canvas,
                        Line hLine, Line vLine)
```

The first two parameters are the x and y coordinates of the center of the PacMan (where the mouse was clicked). The third parameter is the starting size (diameter) of the PacMan (40 pixels as the starting size). The last two parameters are references to the existing horizontal and vertical Lines used to divide the canvas into the four different quadrants.

The PacMan is created using a FilledArc. Here is what the FilledArc constructor looks like:

```
FilledArc(Location center, double width, double height, double startAngle,
double arcAngle, DrawingCanvas canvas)
```

For the startAngle and the arcAngle, you should define the following constants at the top of the ResizablePacMan class to use:

```
private static final double START_ANGLE = 0;
private static final double ARC_ANGLE = 300;
```

More about the FilledArc can be found in the objectdraw docs.

For now, the ResizablePacMan constructor should just create and display the PacMan.
Remember: the FilledArc object shapes use the upper-left corner of an invisible bounding box as

the x and y location to draw their shapes (just like the FilledOval for Mickey). So translate accordingly.

Since the ResizablePacMan is an ActiveObject, call the start() method as the **last** line code in this constructor.

Stage 4: Resizing the PacMen

You should add a **run()** method to your ResizablePacMan class to deal with the resizing animation of the PacMen. The diameter of the PacMan should grow by 2 pixels each step to make it a smooth transition. You should pause for 50 milliseconds in each iteration so the PacMen won't grow and shrink too quickly. If the PacMan grows to twice its starting size, it should start to shrink. Once the PacMan reaches half its starting size, it should start to grow.

Because the ResizablePacMan is an ActiveObject and therefore a Thread, it can run on its own, independent of all other objects. The run() method contains all of the code that will run when the thread is started. You start the thread by calling the **start()** method at the end of the constructor when the PacMan is created. (See Ch 9 and the lecture notes on Active Objects. We will go over all of this in class.)

Stage 5: Spinning the PacMen

Initially, in your constructor, the start angle of the pacMan should be zero (just pass in newStartAngle). To make the PacMen spin, you should change the start angle of the FilledArc. In order to do this, you should increment newStartAngle by a constant. This constant will be defined as CHANGE_ANGLE. If the angle is positive, it will spin counterclockwise, and vice-versa. Here is the logic for updating the angle:

```
newStartAngle = (newStartAngle + CHANGE_ANGLE) % MAX_DEGREES
```

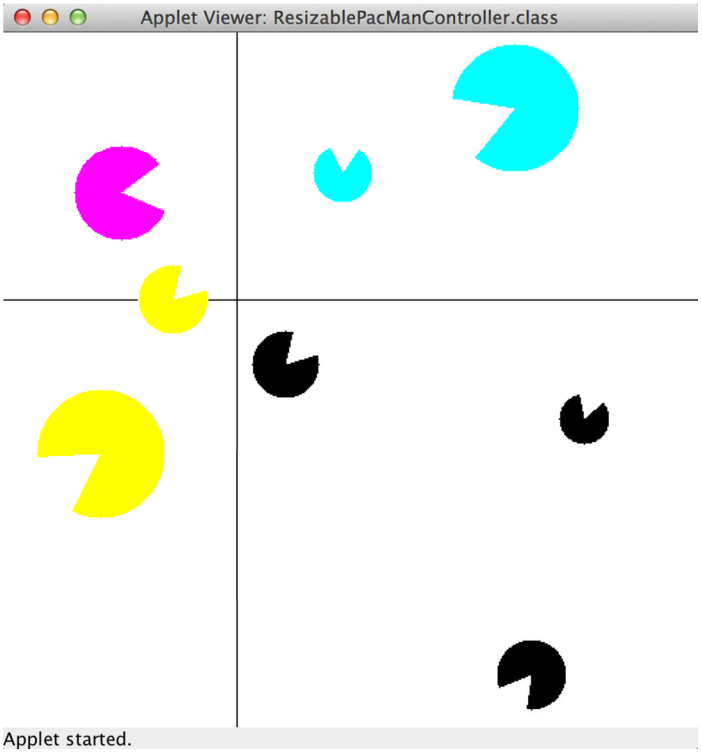
(The reason we are modding by MAX_DEGREES is to avoid overflow.)

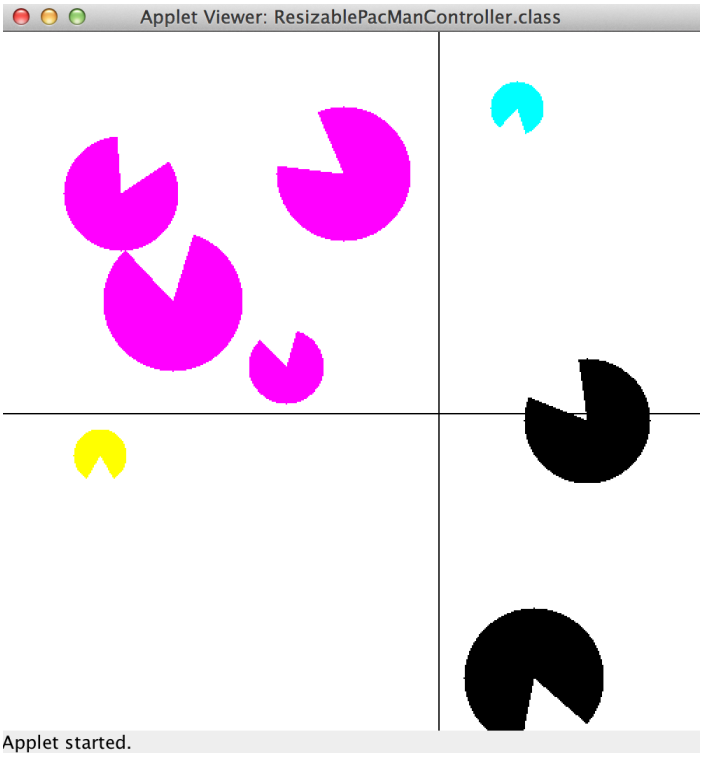
This, just like the resizing, should be done in the **run()** method. To help you achieve the spinning effect, you should look at the following method in the FilledArc class:

```
void setStartAngle(double startAngle)
```

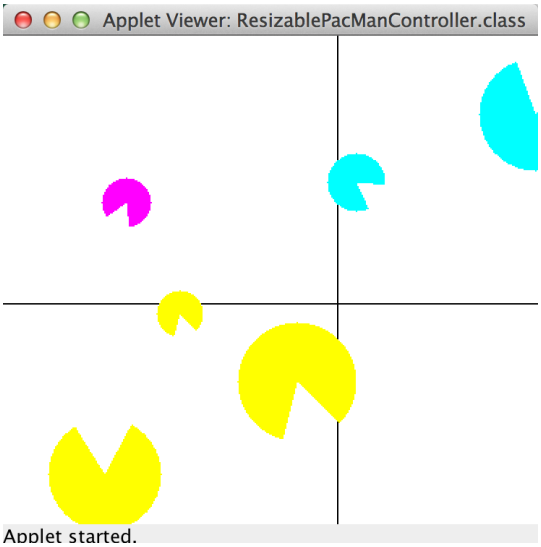
Stage 6: Adding Color

When you click in a quadrant, the PacMan that is created (centered at the point of the mouse click) will be a particular color from CMYK (Cyan/Magenta/Yellow/Black – the printer colors) based on quadrants I-IV respectively: upper-right – **Cyan**; upper-left – **Magenta**; lower-left – **Yellow**; lower-right – **Black**. You should set the color of each PacMan based on which quadrant the center of the PacMan is located in at any particular time. Use the two Lines passed in to the constructor and the PacMan's center to determine which quadrant the PacMan is in. When the lines move they redefine the quadrant areas and the PacMen need to possibly change color. The lines also move when resizing the applet, meaning that they stay proportional to how they were on the canvas before the applet resizes.

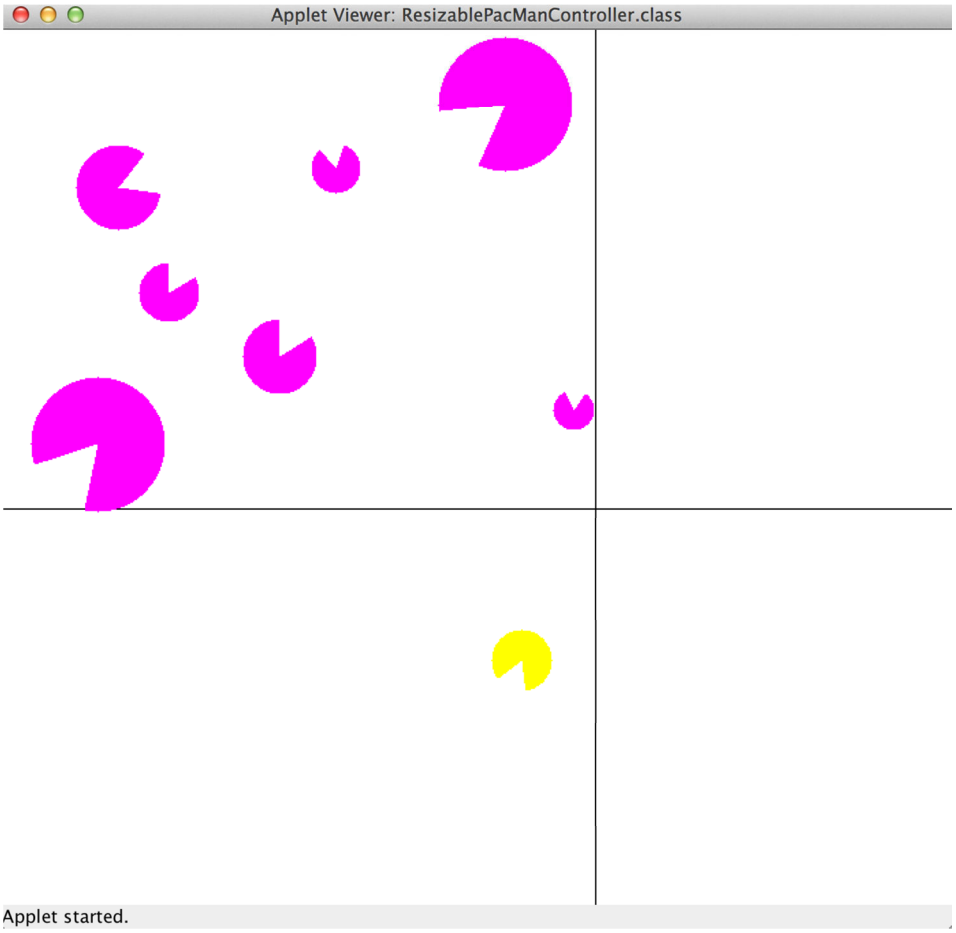




Resizing the applet to be smaller:



Resizing the applet to be larger:



Note: the PacMen do not move when the applet resizes.

Running

To run (and view) your applet, first create an html file named **ResizablePacManController.html**, which contains the following code:

```
<html>
 <body>
      <applet
           code="ResizablePacManController.class"
           archive="objectdraw.jar"
           width="400"
           height="400"
      >
      </applet>
 </body>
</html>
```

Then use the appletviewer command specifying this html file:

> **appletviewer ResizablePacManController.html**

You must have all the files in the pa4 directory and run appletviewer in the pa4 directory for it to display correctly.

Turnin

To turnin your code, navigate to your home directory and run the following command:

> **turnin pa4**

You may turn in your programming assignment as many times as you like. The last submission you turn in before the deadline is the one that we will collect.

Verify

To verify a previously turned in assignment,

> **verify pa4**

If you are unsure your program has been turned in, use the verify command. We will not take any late files you forgot to turn in. Verify will help you check which files you have successfully submitted. It is your responsibility to make sure you properly turned in your assignment.

Files to be collected:

- objectdraw.jar
- ResizablePacMan.java
- ResizablePacManController.html
- ResizablePacManController.java
- README

EXTRA CREDIT:

Extra credit will be given for turning in your assignment early. You can earn up to maximum of 3 points (3%) extra credit.

Final Turnin Date:	**Extra Credit:**
Monday, Oct. 27. 11:59pm	3pts
Tuesday, Oct. 28. 11:59pm	2pts
Wednesday, Oct. 29. 11:59pm	1pt

Note: Only your latest turnin submission will be considered for receiving extra credit. This is because each submission overrides the last one.

NO LATE ASSIGNMENTS ACCEPTED!

START EARLY!

And above all … # HAVE FUN!