Lab 9: Advanced GUI: Drawing Graphics

UNK CSIT 150 Object Oriented Programming

Objectives:

- Familiarize yourself with basic software design principles
- Familiarize yourself with the MouseListener Interface
- Introduce programming of interactive graphics.
- Practice using the Model-View-Control software development architecture to keep the data, logic and the view separate.

1. Exercise 1

- Create a new project
- Create a new package "exercise"
- We will be creating three classes that will be given rather generic names as to their function:
    1. ExerciseWindow
    2. ExerciseDisplay
    3. ExerciseGame

  Create these three classes making ExerciseWindow your executable by adding a main method to this class only.

```
public static void main(String[] args)
{
    ExerciseWindow ew = new ExerciseWindow();
}
```

  Remember, the only code in your main method should be a call to the constructor of the class.

ExerciseWindow:

- In this class we will be working with classes from both the javax.swing and java.awt libraries, so it is recommended that these both we imported using the wild card method. So, type or cut and paste the following code immediately following your package statement.

  import javax.swing.*;
  import java.awt.*;

- We will start working on the Exercise window by adding the line to allow our class to inherit the subclass **JFrame.** So add the phrase **extends JFrame**, to the class header of the ExerciseWindow class header

Next we will create the constructor. Remember, the constructor is public, with no return value, and named exactly after the class. This constructor will not have any arguments. Now, call the four methods that should be included in each window program. Remember, the program will compile and run without these calls but good programming style is the difference between a running program and a great running program!

- For basics method calls, type or cut and paste the following code into the no-argument constructor of the **ExerciseWindow**:

  this.setTitle("Exercise Window");
  this.setSize(win_width,win_height);
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  this.setVisible(true);

- In the **setSize** method, you can see that we set our height and width with instance variables. These should be declared as instance variables and can be initialized either through the

constructor or by having default values set. We will use the default value method. So, type or cut and paste the following code immediately after the class header.

```
private int win_width = 400;
private int win_height = 300;
```

This takes care of the "administrivia" of the class. Now let us add the instance variable **panel**, which will be of the type ExerciseDisplay. This should be declared in the same area as the width and height variables.

- So, type or cut and paste the following code immediately after the width and height variables.

```
private ExerciseDisplay panel;
```

Now we will instantiate this variable. We will do so in a method call initiateWindow(), which will be a void no argument helper method.

So, type or cut and paste the following code following the constructor:

```
public void inititalizeWindow()
{
        panel = new ExerciseDisplay();
        this.add(panel, BorderLayout.CENTER);
}
```

You will notice that the compiler is complaining about this. This is because we need to add a bit of code to the ExerciseDisplay class. Ignore it for the time being. Don't forget to add the method call , `inititalizeWindow()`, to the constructor**, before** the setVisible method call. Your code in the constructor, should look like:

```
this.setTitle("Exercise Window");
this.setSize(win_width,win_height);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
inititalizeWindow();
this.setVisible(true);
```

Now this class is complete for the time being. We now will work on the ExerciseDisplay class.

ExerciseDisplay class:

- In the **ExerciseDisplay** class we will be working with classes from both the javax.swing, java.awt, and the java.awt.event libraries, so it is recommended that these both we imported using the wild card method. So, type or cut and paste the following code immediately following your package statement.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

- We will add the line to allow our class to inherit the subclass **JPanel.** So add the phrase **extends JPanel,** to the class header of the ExercisePanel class header

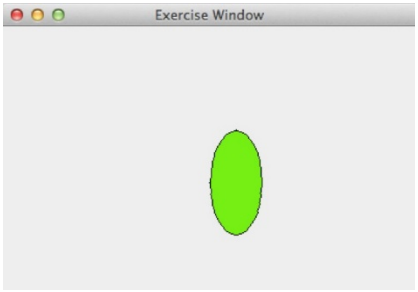
- Now we will add the method that draws the graphics we want to appear in the exercise panel and initioze the instance variables we use to avoid numbers in the code. So type or cut and paste the following code within the class braces.

```
private int circleX = 200;
private int circleY = 100;
```

```
private int circleWidth = 50; private int
circleHeight= 100;


public void paintComponent(Graphics  g)
{
    g.setColor(Color.GREEN);
    g.fillOval(circleX,  circleY, circleWidth, circleHeight);
    g.setColor(Color.BLACK);
    g.drawOval(circleX,  circleY, circleWidth,  circleHeight);
}
```

- Now run the program. You should see a window as shown below.



You should notice that click as you may on the graphic, nothing happens!! Next we will add the code to make something happen.

2. Exercise 2: Interactivity

- Type or cut and paste the following code, to create the default constructor with an anonymous MouseListener.

```
public ExerciseDisplay() {
   this.addMouseListener(new MouseListener() {
      public void mouseClicked(MouseEvent me) {
         circleX = me.getX() - circleWidth / 2;
         circleY = me.getY() - circleHeight / 2;
         System.err.print("\n mouse clicked");
         repaint();
      }
      public void mouseEntered(MouseEvent e) { }
      public void mouseExited(MouseEvent e) {}
      public void mousePressed(MouseEvent e) {}
      public void mouseReleased(MouseEvent e) {}
   });
}
```

Why do you think the term `-circleWidth/2` is add to the `me.getX()` when assigning the value to `circleX`.

You should notice that we are including code in the mouseClicked method only. This is the only one of the five mouseListener methods that we are using. However, all five **must** be present.

- Comment one of these 5 methods out. What happens. You should see the red underline under the add mouse Listener method call. This is because the Mouse listener you are trying to add is not complete until all 5 methods are present.

- Now your code should be ready to run. Run it! Click on the panel and observe what happens. Your circle should be moved to any spot you have clicked.

- If you add the following line to the start of the paint component method, it will remove the other circles as you click to create a new one:

```
super.paintComponent(g);
```

3. ExerciseGame:

Next we will work on the **ExerciseGame** class. This is where your data structures and your program logic should be.

- So let us start by moving the 4 instance variables from the ExerciseDisplay class to this class. The compiler will compile vehemently but ignore it for the time being.
- Add the getters and setters to **the ExerciseGame** class:

```
public int getCircleX() {return circleX;}
public int getCircleY() {return circleY;}
public int getCircleWidth() {return circleWidth;}
public int getCircleHeight() {return circleHeight;}

public void setCircleX(int circleX) {
    this.circleX = circleX;
}

public void setCircleY(int circleY) {
    this.circleY = circleY;
}

public void setCircleWidth(int circleWidth) {
    this.circleWidth = circleWidth;
}

public void setCircleHeight(int circleHeight) {
    this.circleHeight = circleHeight;
}
```

- Now we will add a constructor that sets the initial value for these two variables.

```
public ExerciseGame(int x, int y, int wid, int hei)
{
       setCircleX(x);
       setCircleY(y);
       setCircleWidth(wid);
       setCircleHeight(hei);
 }
```

Now we will add the method: `processMove`. This methods communicates between the "Game" and the display of the game.

```
public void processMove( int x, int y)
{
 setCircleX(x);
 setCircleY(y);
}
```

Back in the **ExerciseDisplay** Class and the complaining compiler. First we must create an instance variable of the ExerciseGame class. So add the following instance variable to the ExerciseDisplay class.

```
private ExerciseGame game;
private final int START_X = 200;
private final int START_Y = 100;
private final int START_Width = 50;
private final int START_Height= 100;
```

In the constructor, the game must be instantiated in the constructor of the ExerciseDisplay.

```
game = new ExerciseGame(START_X,START_Y  ,START_Width, START_Height);
```

In the mouseClicked method, the following two lines of code

```
circleX = me.getX()-circleWidth/2;
circleY = me.getY()-circleHeight/2;
```

must be replaced by

```
game.processMove(me.getX()-50,me.getY()-50);
```

And in the paintComponent method, replace

```
g.fillOval(circleX, circleY, circleWidth, circleHeight);
```

with

```
g.fillOval(game.getCircleX(), game.getCircleY(), game.getCircleWidth(), game.getCircleHeight());
```

and replace

```
g.drawOval(circleX, circleY, circleWidth, circleHeight);
```

with

```
g.drawOval(game.getCircleX(), game.getCircleY(), game.getCircleWidth(), game.getCircleHeight());
```

You should now be able to run your code completely separating you data and logic from your display.

4. On your own Exercise:

Add the FormCircleArray class (from graphics.zip) to this package. In FormCircleArray, move the width, height, startX, and startY variables in the generateCircArray method to the top of the class (making them class variables). Modify the methods appropriately to accept the starting width and height variables and the starting x and y values as input parameters. Set these variable values in the generateCircArray method. (You do not need to also add extra getters and setters for these variables.)

Now, add a FormCircleArray object to the ExerciseGame class. You will no longer need the circleX or circleY variables, as the x and y values from the mouse click will be passed into the FormCircleArray object. You can also remove the related getters and setters for the circleX and circleY variables. Create the FormCircleArray object in the constructor, using a random number of circles (like the sample DrawingDisplay program does). Be sure to pass the x, y, height, and width values along to the FormCircleArray object. In the processMove method, call the genNewArray (with a random number of circles) for the FormCircleArray object, starting at the place where the mouse clicks. (Remember, these mouse values are passed to the ExerciseGame from the ExerciseDisplay class.) Change the return type of the processMove method to return the circle array from the FormCircleArray.

Add a two dimension array in the ExerciseDisplay for the circles (see the graphics.zip DrawingPanel example). Set the array value as you call the processMove method. Then draw the circles in the paintComponent method (again see the graphics.zip DrawingPanel example.)

Be sure to document each method!

For bonus, you can modify it so the FormCircleArray width and height values are random (using the ExerciseGame's height and width as maximums rather than actual values.)

Lab 9: GUI Drawing

Name(s): _____

Evaluation

Requirement	**Possible Points**	**Points Received**	**Comments**
FormCircleArray class- startX,startY, width, height set appropriate in generateCircArray; other methods have these inputs	2		
ExerciseGame – FormCircleArray object; process move; mouse click	2		
ExerciseDisplay – two dim array of circles; drawn appropriately in paintComponent	2		
JFrame set up	2		
JPanel setup	2		

Programming style	**Possible Points**	**Points deducted**	
Inconsistent indentation	**-1**		
Poor use of white space	**-1**		
Heading documentation, includes programmer names, date, algorithm, basic purpose	**-2**		
All sources not cited (Remember to cite all code used, even class demo code.)	**-1**		
JavaDocs not used on each method	**-1**		
Poor variable names	**-1**		
Poor structure/logic issues	**-2**		
Program specifications			
Bonus:			