Lab 9 Name:_____ Checked:_____

Objectives:

More practice designing classes and methods.
Practice creating graphical objects.

Useful Links:

- www.csc.villanova.edu/~map/1051/Chap04/Splat.java
- www.csc.villanova.edu/~map/1051/Chap04/SplatPanel.java
- www.csc.villanova.edu/~map/1051/Chap04/Circle.java
- www.csc.villanova.edu/~map/1051/Chap02/Snowman.java
- www.csc.villanova.edu/~map/1051/Chap04/Die.java
- ***lecture slides:*** www.csc.villanova.edu/~map/2014/f14/04GUIclasses.pdf

1. Getting Started

- Download and test the **`Splat.java`**(driver), **`SplatPanel.java`**, and **`Circle.java`** classes.
- Review the code to understand it.
- Incorporate two more circles in the SplatPanel.
- Change the background color to gray.
- Increase the size of SplatPanel to 1200x800 or even larger to accommodate more objects that will be added later.

In steps 2-5, we will change this to a program that displays smiley faces instead of circles.

2. Renaming Classes

We will be turning the circles into smileys. We begin by renaming all the classes as follows:

`Splat → Happiness`
`SplatPanel → HappinessPanel`
`Circle → Smiley`

Note: It is best to create a separate subfolder for the files in this exercise (and nothing else, not even Splat, SplatPanel, and Circle) to avoid problems going forward.

You will need to rename the constructors in **`SplatPanel`** and **`Circle`**. Go through the code in all three classes to make any other changes necessitated by the change of class names (many changes are needed – eliminate all occurences of SplatPanel and Circle throughout!). Re-compile all the classes and run to make sure everything still works and that the displayed image looks exactly the same as before.

In the following steps, we will be turning the circles into smiley faces, but so far they should still look just like circles.

3. Simplify the `Smiley` class by eliminating an instance variable

Determining the position of the eyes and smile will depend on the size of the circle that represents the face, so we will simplify our work by making all the circles have a fixed diameter, say 50 pixels. (If you like the challenge of adjusting to different diameters, you may omit this step.)

- In the **`Smiley`** class, replace **`diameter`** with a constant named **`DIAMETER`** (i.e., capitalize its name and add the modifier **`final`** in the declaration), and set it to **`50`**. Replace **`diameter`** with **`DIAMETER`** in the **`draw()`** method. Eliminate **`diameter`** from the parameters of the constructor eliminate the accessor and mutator methods for **`diameter`** altogether, as these no longer make sense.
- Simplify also the **`HappinessPanel`** class – the code should no longer give values for the **`diameter`** parameter of the **`Smiley`** objects.

Test your program before proceeding; it should still look about the same, except the circles are now all the same size.

4. Drawing a smiley face

In the **`Smiley`** class, modify the **`draw()`** method so that it adds black ovals and an arc on the circle, so as to make a smiley face.

Recompile the **`Smiley`** class and run **`Happiness`** again to test your code. You should see smileys where you had circles. You will probably need to make some adjustments to fix lopsided faces or missing eyes (Note: if you miscalculated, they may have ended up outside the circle, so look around!).

5. Enhance the Smiley class

Modify the **`Smiley`** class further by adding some attributes:

`name` (a String), **`age`** (an int), and **`happiness`** (a boolean)

You will also need to modify the way the constructor and the **`draw()`** methods work:

- Modify the exiting constructor (the one that only has color, x, and y as parameters): it should also set the **`name`**, **`age`**, and **`happiness`** to some default values, such as "**`Smiley`**", **`0`**, and **`true`**, respectively.
- Add another constructor with more parameters that allows you to set the **`name`**, **`age`**, and **`happiness`** to other values.
- Modify the **`draw()`** method so that the smiley looks different depending on the **`age`** and **`happiness`** state and to include the **`name`** (*Hint:* Use the **`Graphics`** method **`drawString`**() to display the **`name`** below the Smiley).

Test your code well before proceeding.

6. Create a Snowman class

Create a **`Snowman`** class, similar to the **`Smiley`** class. You can use much of the code from the Snowman applet – the constants **`MID`** and **`TOP`** will now be your instance variables, corresponding to the x, y position, as in the **`Smiley`** class. (You should rename **`MID`** and **`TOP`** to x and y or to mid and top and declare them **`private`**). You don't need any other instance variables, unless you plan to have different versions of the **`Snowman`** (eg: arms up vs. arms down).
Test your code by drawing a few Snowmen in your H**`appinessPanel`**.

7. Enhance the Die class by adding a draw() method

A die can be depicted by a white square outlined in black, with the number inside (or you can take the challenge and try to make it look like a real die, but drawing the little dots corresponding to the **`faceValue`** is tricky).
Note that the **`Die`** class does not have x and y (position) attributes. Rather than adding these attributes, we will take a different approach, and incorporate x, y as parameters to the **`draw()`** method that we are writing. Thus, in the **`paintComponent`**() method of **`HappinessPanel`**, we will use:

```
die1.draw(page, 40, 60);
```

(instead of **`die1.draw(page);`** which assumes **`die1`** has a position)

Therefore, the draw() method in the Die class will need a different heading:

```
public void draw( Graphics page, int x, int y)
```

The method definition should use the position x, y to draw the die:

a. use fillRect() to create a white square (or use it twice to create a white square with a black outline)
b. use drawString() to put the String corresponding to faceValue inside the white square. Note: remember that faceValue is an int, so use Integer.toString(faceValue) to convert it to a String, as is done in the original toString() method of Die class.
c. Test your code by drawing a few dice in your HappinessPanel.
d. Test it again, inserting die1.roll() right before the draw() method is invoked in the paintComponent() method of HappinessPanel, for one of the dice. Observe the dice as you resize the window: note that the paintComponent() method is invoked every time the window is resized, so if a die is rolled in paintComponent(), you should see that die's faceValue change as you resize.