CSC 1051 – Project 6

Project due Monday April 8, 2013

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
- www.csc.villanova.edu/~map/1051/Chap04/Account.java
- lecture slides: www.csc.villanova.edu/~map/1051/04GUIclasses.pdf

This week we will not have a proper “Lab” but will work on this project in class.
We will work with the examples from the textbook, section 4.6, to create classes for graphical objects, similar to the Circle class.

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

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 that 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.
   - In the following steps, we will be turning the circles into smiley faces, but first you should re-compile all the classes and run to make sure everything still works and that the displayed image looks exactly the same as before.
   - We will be adding more elements to the HappinessPanel, so you should increase its size (try 1000x600 or even larger to accommodate more things that will be added later).

3. 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. In the Smiley class, get rid of the instance variable

---

1 If you like the challenge of adjusting to different diameters, you may omit this step.
for diameter. Replace it with a constant named DIAMETER, set that to 50. Replace diameter with DIAMETER in the draw method. Eliminate diameter from the parameters of the constructor and eliminate the accessor and mutator methods for diameter altogether, as these no longer make sense.

In the HappinessPanel class, modify the code that creates the Smiley objects (it should no longer have a diameter parameter). 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
Modify the Smiley 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 (hint: if you miscalculated, they may have ended up outside the circle). If you have trouble finding where you went wrong, you may want to change the background color of the applet to something other than black, so you can more easily locate any missing eyes.

5. Adding more instance variables
Modify the Smiley class further, making it more like the Person class – i.e., add name, age, and happiness state attributes. You will need to also modify the way the constructor and the draw methods work:
• Add another constructor with more parameters that allows you to set the name, age, and happiness to other values.
• Modify the old 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.
• The draw method can draw the smiley differently depending on the age, happiness state, and it can also include the name (use drawstring()).
Test your code well before proceeding.

6. 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 (e.g: arms up vs. arms down).
Test your code by drawing a few Snowmen in your HappinessPanel.

7. A draw() method for the Die class
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, rather than invoking the method like this:

\[
\text{die1.draw(page)};
\]

which assumes die1 has a position, we will specify the position (e.g., horizontal position 40, vertical position 60) by invoking the method like this:

\[
\text{die1.draw(page, 40, 60)};
\]

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

\[
\text{public void draw( Graphics page, int x, int y)}
\]

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

- Test your code by drawing a few dice in your HappinessPanel.
- Test it again, inserting die1.roll() right before the draw() method is invoked in the paintComponent() method of HappinessPanel. Only do this for one of the dice and observe the dice as you resize the window. The paintComponent() method is invoked every time the window is resized. Thus, you can tell from this, that the paintComponent() is invoked not just once, but potentially many times when the program runs.

* Optional: a draw() method for Account

The same considerations apply to creating a draw() method for the Account class.

8. Demonstrate your work

At this point, you should have five classes implemented:

Happiness
HappinessPanel
Smiley
Snowman
Die

Go over the code and make sure it is indented correctly and has appropriate comments. It is recommended (though not required) that you show your work to the TA or instructor to make sure you are doing things correctly. Begin writing the report for this project: Include listings of the programs for these classes in your report for this project.

9. Design your own class

You should include the x, y position among the attributes (as in the Smiley and Snowman classes, not like the Die class). What else can you use to vary the objects you create? Be sure to use at least two additional attributes. Here are some ideas:

- number of petals/eyes etc
- size (this is the attribute we eliminated for Smiley, but it may be easier to implement for other things)
- big/small (like size, but only two options – easier to control)
- color
- a boolean attribute, like happy
- a String (such as a name or description)

You have a lot of freedom in this project, but be sure to use as much of what you learned as possible.

Testing
Incorporate your new objects in HappinessPanel, along with the other objects designed in the lab. Experiment with different arrangements and positioning. Test your program well.

UML class diagram
Include a UML class diagram that shows all your classes and their dependencies.

Style and Documentation
Be sure to review your code to ensure that all your classes have appropriate comments and that the code is well indented. In addition to working well, programs should adhere to the general style and documentation guidelines used in the textbook. It is especially important that your name appears in the comments of all your classes.

Report
What to include:

1. A detailed description of the class that you chose to design. Include:
   a. discussion of your choice of attributes
   b. implementation notes
   c. examples of how your class can be used
2. Source code for all your classes
3. Screen shots (more than one if you are using any random elements).
4. UML class diagram for your classes
5. A reflection about this project, including comments about your collaboration, if any, in addition to the usual required comments: How long did it take? Was it easier/harder than you expected? How did you approach it? What would you do differently if you were starting over or had more time? Include any other comments and observations here.