Because drawing graphical images with code is fun and a good way to look at different Java classes, I created a class to handle shapes this weekend. I could have created it best as an interface, but we have not learned those yet. If not an interface, I could have created it as an abstract class, but we have not learned those either – so, I created it as a super class. When you learn more about classes, we will modify this to be an interface.

To use the Shape class, you need to extend it, using the extends keyword in the class header.

For instance:

public class Circle extends Shape {

}

Here are some questions that will help you understand that Shape class:

1. How many fields are there?
   1. There are four fields
2. What are the data types for each of the fields
   1. Int, Point, Dimension, Color
3. For each field with a non-primitive data type, look up the Java API reference for the data type and answer the following (for EACH non-primitive field in Shape):
   1. Point
      1. java.awt.point
      2. x- int and y- int
   2. Dimension
      1. java.awt.dimension
      2. height- int and width- int
   3. Color
      1. java.awt.color
      2. all of the following are of the type static Color: black, blue, cyan, darkGrey, grey, green, lightGrey, magenta, orange, pink, red, yellow, white
4. How many constructors are there in Shape?
   1. There are 11 constructors.
5. How many getters are there?
   1. There are 3 getters.
6. How many setters are there?
   1. There are 5 setters.
7. Why are there a different number of setters than there are fields in the class?
   1. There were more setters because you had top set x and y as a coordinate but also specify that that is its location as well. The same goes for height and width. You must set those but then also specify that those are the dimensions.
8. If you do not override the method getArea() in the classes that you will create to extend Shape (such as Square, Circle, and Triangle) what will a call to getArea return? What about getPerimeter?
   1. getArea will retur
9. If you do not override drawMe(), what will be drawn?
10. What information do you think your toString() should add to the toString() provided in shape?

In addition to the Shape class, you have been provided with completed ShapeGraphics and DrawStuff classes. You need to save them in the same folder as the Shape class. There are very slight modifications that need to be made in the DrawStuff class, look for the TODO comments. The DrawStuff class is the class that has the main() method that runs. There are incomplete versions of Circle, Rectangle and Square for you to fill in.

Create a Circle class that extends Shape. Like a square, a Circle needs only one dimension. You need to determine if your Circle will store the radius or diameter. A circle should have a different getArea method, a getRadius and a getDiameter, it should have a setter for both radius and diameter (even though you are only using either the radius OR the diameter), and it should also have a getCircumference method, which other shapes do not require. In order to override the drawMe method, you will need to look up how to draw a circle in Java … use the Java API. Print out a screen shot of the web-site where you found the information, include the method call and the URL in the screen shot. (fill in the getPerimeter method as well … it is a circle … but also a shape).

Create a Rectangle class … think about what the shape class offers by default – what can you not provide with the rectangle class, because it is already there by default?

Create a Square class that extends Shape (or that extends Rectangle – either will work, you decide). You need to handle the dimensions differently, as for a square, the user should only provide one dimension – you should handle setting the second one.

Consider what it would take to create a Triangle class that extends Shape (we will do this later in the year). Brainstorm with your partner how to handle the differences for this shape. What additional fields will you need? There are many different ways to store the information needed in order to draw a triangle, determine its height and width, and calculate its area. Turn in your brainstorming sheet (A series of pictures of your whiteboard brainstorming suffices). Code the class. How should you handle special triangles (Equilateral, Right, Isoceles, etc)

What about a Polygon class? How could you handle a polygon? What does Java use to draw a polygon? find it in the API Reference.

What does the Graphics2D class allow you to do (HINT: Look at the methods in the Java API)