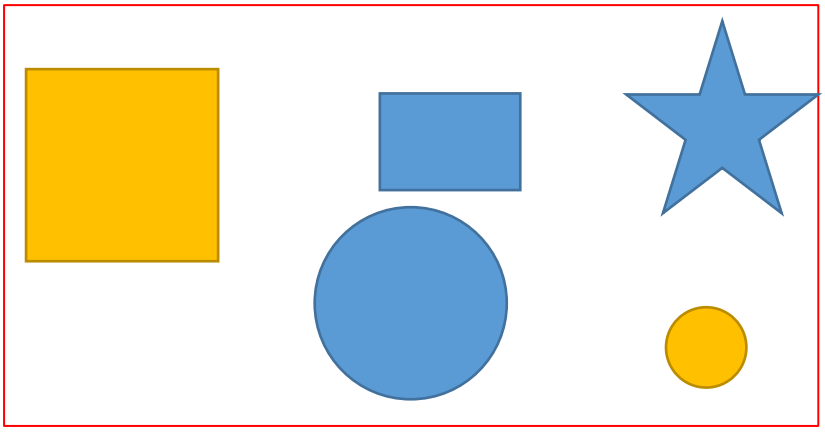
Several of the Microsoft Office products share the feature of a canvas. It allows the users to draw on it, like the following diagram (the border lines belong to the canvas):



Consider a simplified program, where we can only draw circles and rectangles on the canvas. Choose one of these languages: C++/C#/Java, and write down some pseudo-codes for the following questions.

1. Define two classes for a Circle and a Rectangle. For both classes, we would also like to know where the center of the circle/rectangle is, and get the area of the circle/rectangle. (10pts)

```
1  public class Circle
2  {
3      private double centerX, centerY;
4      private double radius;
5
6      public double getCenterX() { return centerX; }
7      public double getCenterY() { return centerY; }
8      public double getArea() { return PI * radius * radius; }
9  }
10
11 public class Rectangle
12 {
13     private double centerX, centerY;
14     private double width, height;
15
16     public double getCenterX() { return centerX; }
17     public double getCenterY() { return centerY; }
18     public double getArea() { return width * height; }
19 }
```

2. Obviously, these two classes share some common "characteristics". Define a "Shape" class that encapsulates these common characteristics. What is the name of the relationship between the Rectangle/Circle class and Shape class, and what do you need to change about your codes in Q1? Also, why do we want/need the shape class at all? (10pts)

```
21  //The Shape class in this case would encapsulate
22  //the common characteristics from Circle and Rectangle
23  public abstract class Shape
24  {
25      private double centerX, centerY;
26
27      public double getCenterX() { return centerX; }
28      public double getCenterY() { return centerY; }
29
30      public abstract double getArea();
31  }
```

```
47  //And the Circle and Rectangle classes would *inherit* from the Shape class
48  public class Circle extends Shape
49  {
50      private double radius;
51      public double getArea { return PI * radius * radius; }
52  }
53
54  public class Rectangle extends Shape
55  {
56      private double width, height;
57      public double getArea { return width * height;}
58  }
```

3. Should you be able to draw an "unspecified" Shape on the canvas? Or does it have to be a rectangular or circle? What does this mean about the Shape class? Hint: I'm looking for the word "abstract" in your answer. (5pts)

Based on this condition in the problem: "we can only draw circles and rectangles", an "unspecified" Shape is probably best not allowed, which means it would either be a rectangle, or a circle.

This means that the Shape class should be abstract. Hence the abstract keyword above. Also, the getArea() method would be implemented w.r.t each individual subclass, therefore it is an abstract method in Shape, and implemented in Circle and Rectangle.

4. Also note that, the Shapes above all have colors, fills, shades, outlines, etc. To simplify it, let's combine all of these aesthetic elements of the Shape into one thing, and call it the "Style" of the Shape. If we want to include these information, what changes would you need to make to your previous codes? (10pts)

   We can specify a Style class that encapsulates these aesthetic elements, and add a member attribute to the Shape class. Notice that, this is the only change we need – we don't have to go through every subclass (Circle, Rectangle)

```
public abstract class Shape
{
    private double centerX, centerY;

    private Style styleOfShape;

    public double getCenterX() { return centerX; }
    public double getCenterY() { return centerY; }

    public abstract double getArea();
}
```

5. Should Style be its own class? (Hint: yes). What is the relationship between Shape and Style in your code? Hint: your answer should be discussing one or more of the following: association, aggregation, composition, inheritance. (5pts)

   It should be a class, and Shape includes Style as a member attribute, which makes the relationship between them "association" – more specifically, because the Shape "has a" style, it is an "aggregation" relationship.

   Note: give full credits too if the answer is "composition", but the answer should justify it. See Q7

6. Now, the user wants to draw something on the canvas. There are many ways to do this. Consider this method: we have a Canvas class, with a "draw" function. **The user decides upon the shape first**, then clicks on the canvas, and the "draw" function is called, and the shape shows up on the canvas. Write the sketch codes for the Canvas class, and the "draw" function in this scenario. (10pts)

```
//Follow the instruction, the user would "decide" upon the shape first,
//Then click on the canvas and called the draw function.
class Canvas
{
    //because the shapes are "drawn" on to the canvas
    //the canvas should be keeping track of a list of shapes that are on the canvas
    Vector<Shape> shapes;

    //draw function
    //because the user has already decided on the shape
    //the shape should be supplied as an parameter for the draw function
    public void draw(Shape s)
    {
        //draw the shapes. Ideally, this should be in the Shape class
        shapes.add(s); //add the shape to the Vector
        s.draw(); //should be added in the Shape class itself.
    }

    //If you want to be more specific,
    //you can add an click method of the canvas
    //that actually calls the draw function
    //but that's not important for now
}
```

7. What's the relationship between Canvas and Shape: association, aggregation, composition, inheritance? Compare to the relationship between Shape and Style, what are the differences, if there are any? Explain what happen to the shapes when you delete a canvas, and what happen to the "styles" when you delete a shape. (20pts)

The relationship between Canvas and Shape is "composition".
Notice that, between Shape and Style, which is "aggregation", Style is not "dependent" on Shape. Consider the diagram at the beginning of this assignment, if you delete the yellow circle, the yellow square would still be there – i.e., the "Yellow" Style instance is still available.

However, if you delete the entire Canvas, all the shapes on the canvas will be deleted as well. That strong connection indicates the essence of the "composition" relationship.

Note: alternatively, if the answer makes a case why the Shape-Style relationship should also be "composite" (i.e. the Style instances are created specifically for each Shape thus deleting Shape instance would remove the Style instance it contains), then full credits as well.