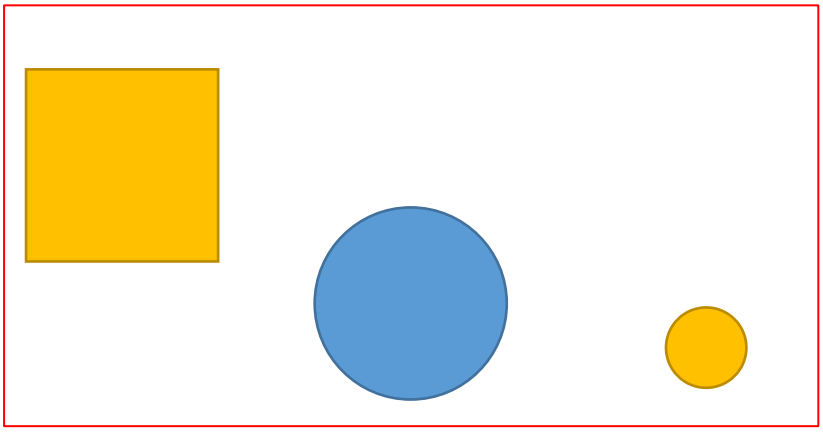
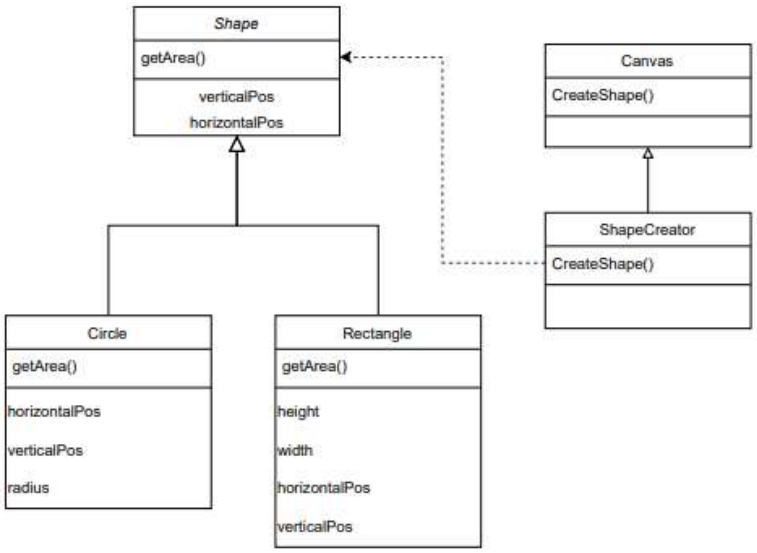
Sean Skinner
CptS 487
12/12/2022
Hw2



1. Consider the example in Assignment 1 (a canvas that can draw only circle and rectangle, disregarding Styles). Follow the Factory Method Pattern, draw a class diagram that include these classes: Shape, Circle, Rectangle, Canvas, as well as other classes necessary. Remember to include proper connections, and necessary attribute/method definitions in your class diagram. (10pts)

   Creator Class: Canvas
   Concrete Creator: Shape Creator
   ConcreteProdcut: Shape
   Product: Circle, Rectangle



2. What is the role of the Canvas class in your design: is it part of the Factory Method Pattern, or is it something external to the pattern? Justify your reasoning. (5pts).

   I decided to include it as part of the factory method pattern, as it is the class that is creating and storing shape objects as they are part of the canvas itself. It reminded me of the in class example discussing a maze game. The maze game is composed of rooms and walls the same way the canvas class is composed of shapes.

3. Write some necessary sketch code for the classes you defined in Q1, that is enough to demonstrate the process of "drawing a circle on a canvas". You may describe the details of methods/functions with comments (no need to write actual code). (10pts)

```
public abstract class Canvas{
    public void createShape(){
        return new Shape();
    }
}

Public class ShapeCreator extends Canvas{
    Public void createshape(String type){
        If(type.equals("Circle")){
            return new Circle();
        }
        else if(type.equals("Rectangle"){
            return new Rectangle();
        }
        else{
            return null;
        }
    }
}

Public abstract class Shape{
    public int getArea();
    public int horizontalPos;
    public int verticalPos;

    //Code for Rectangle Constructor
}

Public class Rectangle extends Shape{
    public int getArea();
    public int horizontalPos;
    public int verticalPos;
```

```
        public int width;
        public int height;

        //Code for Circle constructor
    }

    Public class Circle extends Shape{
        public int getArea();
        public int horizontalPos;
        public int verticalPos;
        public int radius;
    }
```

4. Describe a scenario where the Factory Method might become insufficient, and that Abstract Factory might be better, e.g. such as introducing new elements like the Styles of the Shapes. No need to write code or draw diagram. (5pts)

   A factory method might become insufficient if more features are added, with the factory method outlined above we are creating concrete objects that can not be easily changed. If we can not anticipate the class of object we are going to create because there are so many possibilities with the number of new elements being added it would be beneficial to use an abstract factory method.

5. Open question: in the course project, if you were to use either Factory or Abstract Factory, how would you apply these patterns? Note that there are other Creational patterns that we haven't covered, so this question is to encourage you to start thinking about potential design.

   Not for submission: Use your answer here for discussions within your teams and see if you can see the pros and cons of your choice. (10pts).

   I was thinking we might want to use a factory pattern for the creation of enemies in the course project, however if we want to expand upon the enemy classes and add a variety of features to them it might be smart to instead implement an abstract factory pattern so we can more easily create new objects. However, after discussing with the team this weekend our first milestone will not use a factory method as the group member working on that part of the program wanted to just create the base classes first. We could also use the factory method for the projectile class that will represent the bullets being fired to and from the player.