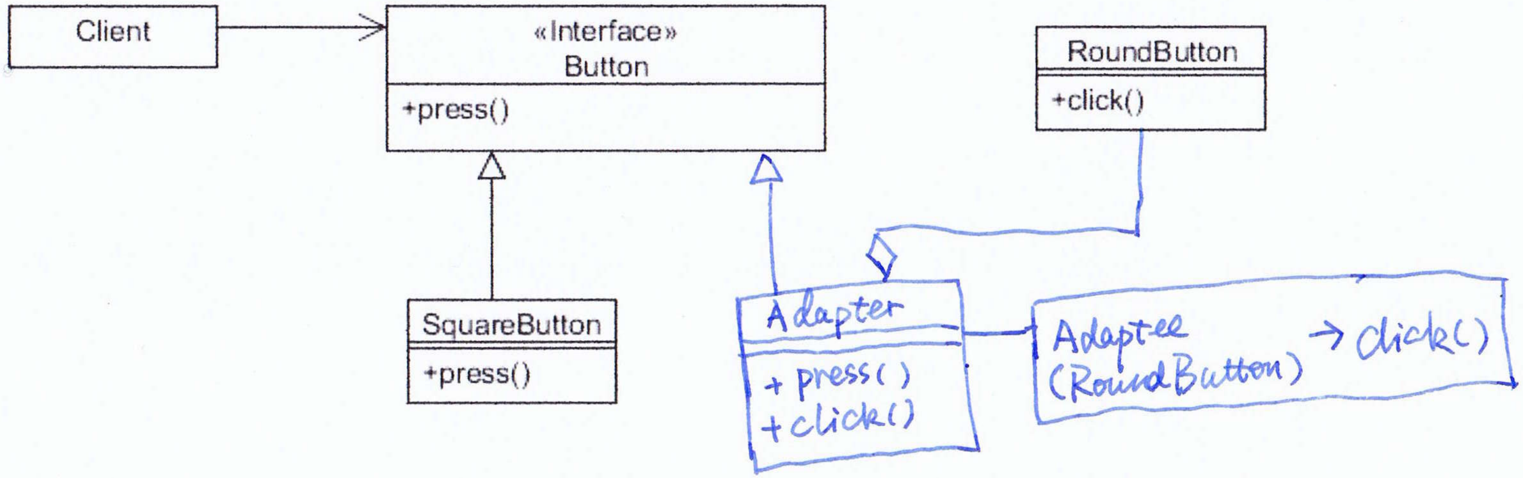
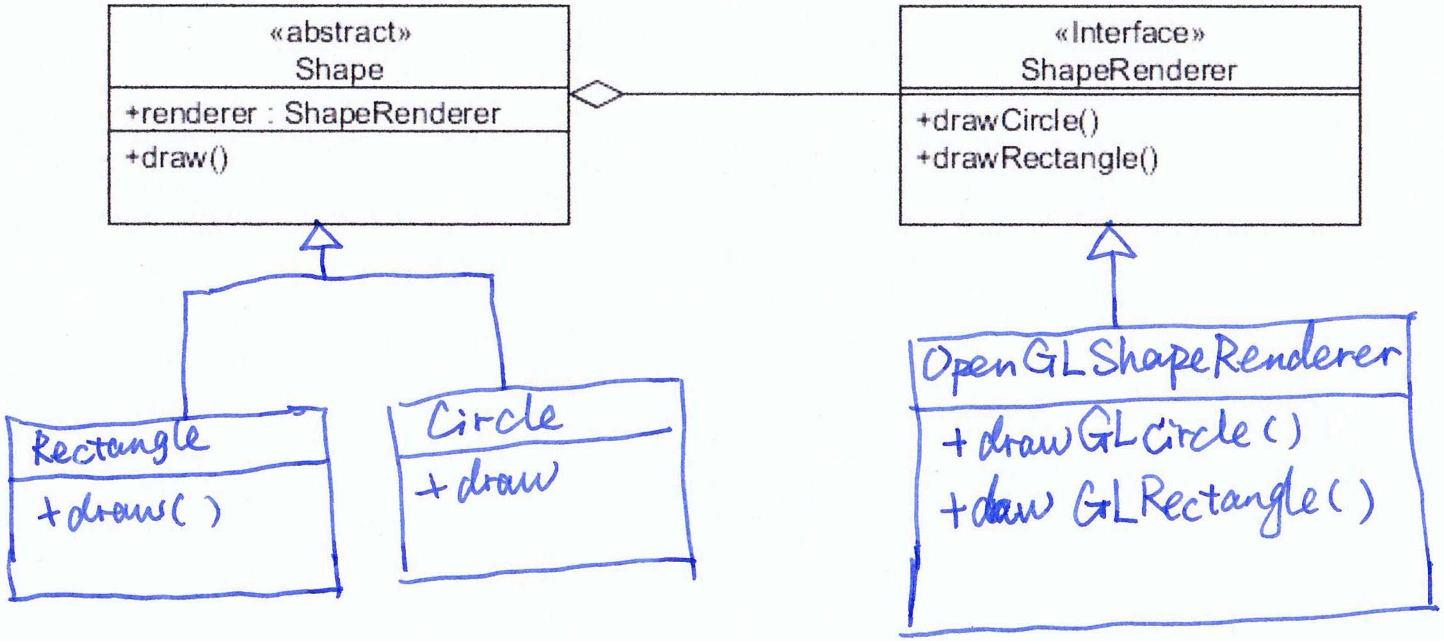
H1:

You currently have a simple GUI application which uses Buttons. You would like your client to use an **existing** class RoundButton which does not match the current button interface. Use the adapter pattern so that the client can use RoundButtons. Note: Use composition over inheritence in the adapter class.



H2:

You wish to design a **new** application which can draw two different types of shapes: rectangles and circles. It is possible that in the future the application will draw other shapes or even draw them differently. Implement the bridge pattern by adding the following classes to the diagram below: Rectangle, Circle and OpenGLShapeRenderer.



«abstract»
Shape
+renderer : ShapeRenderer
+draw()

«Interface»
ShapeRenderer
+drawCircle()
+drawRectangle()

Rectangle
+draws( )

Circle
+draw

OpenGLShapeRenderer
+drawGLcircle( )
+draw GLRectangle( )

H3:

Implement the draw methods of Rectangle and Circle such that any concrete renderer will draw them.

```
class Rectangle extends shap {
   Private double  x, y, height ,width;
 Public Rectangle(double x, y, heigh, width,
  ShapeRenderer  shaperenderer ) {
    super(shaperenderer);
    this. x = x  ...  ...     }
 Public void draw () {
    Shaperenderer. draw Rectangle ( );
  }
 }
```

```
class Circle extends shape {
 Private double  x, y, radius;
 Public Circle (double x, y, radius,
 Shape Renderer shaperenderer) {
  super (shaperenderer);
  this. x= x  ... ...       }
 Pubic Void draw ( ) {
  shaperenderer. draw Circle( );
  }
  }
```

By using this pattern we can change which renderer we use to draw shapes without modifying any code within the Shape classes.

H4:

Many design patterns such as the adapter and bridge pattern are very similar, however it is important to know the difference between them. Describe the fundamental difference between the adapter and bridge patterns.

Adapter makes things work after they are designed; Bridge makes them work before they are.

Adapter pattern is more about getting your existing code to work with a newer system or interface. Whereas bridge pattern is going to allow you to possibly have alternative implementations of an algorithm or system.