Adapter pattern

Brief description and presentation

Brinzas Vlad
Group 30432

When to use?

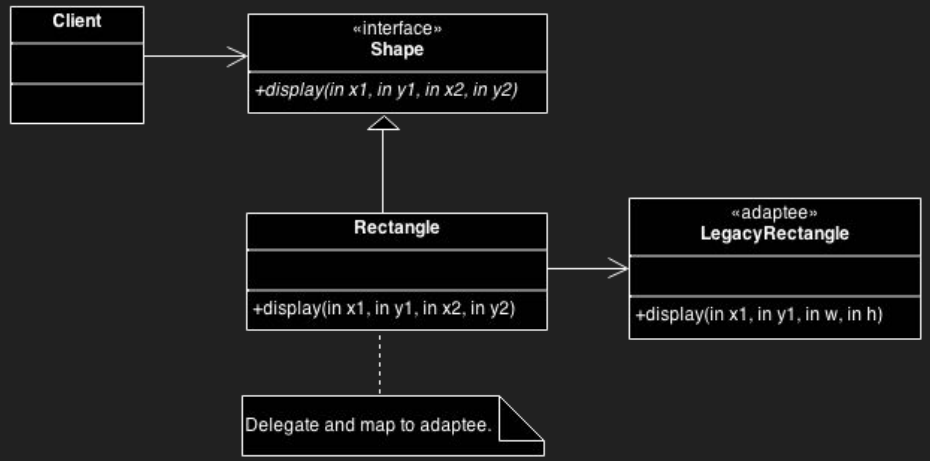
- When we already have a working sub-system...
- ...and we have another bit of code we'd like to use it with…
- ...but their interfaces are different.

...like a real adapter, basically

Examples?

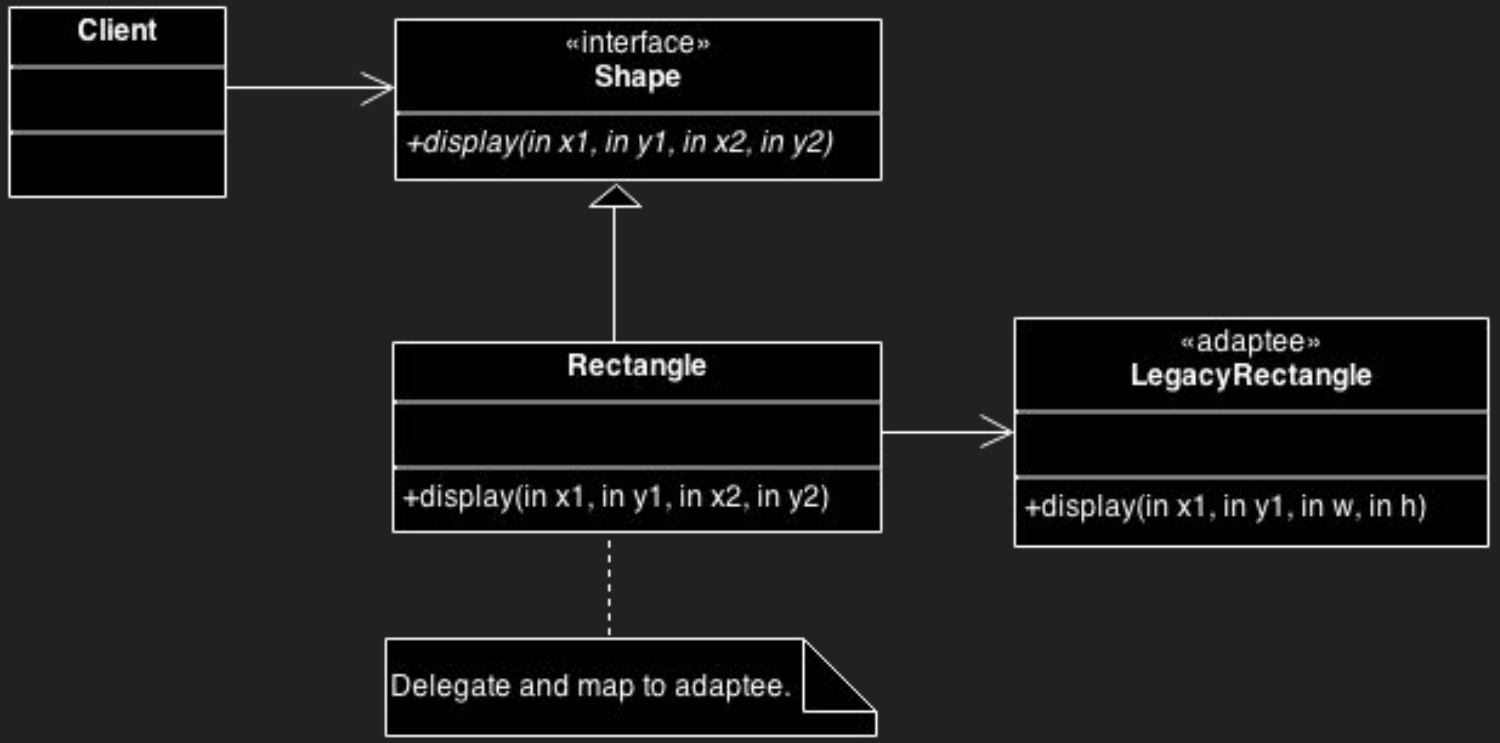
→ Suppose you have a LegacyRectangle class that you wrote long ago. It creates rectangles from a point of coordinates X and Y, and a width W and height H.

→ Now suppose you have a new piece of code that needs rectangular shapes… and creates them by giving the coordinates of two corners!



(Thank you sourcemaking.com for the image)

...just in case...


Client
«interface»
Shape
+display(in x1, in y1, in x2, in y2)
Rectangle
+display(in x1, in y1, in x2, in y2)
«adaptee»
LegacyRectangle
+display(in x1, in y1, in w, in h)
Delegate and map to adaptee.

...could we have code for that?

```
interface Shape {
    void draw(int x, int y, int z,
int j);
}
class Rectangle {
    public void draw(int x, int y,
int w, int h) {
        System.out.println("Rectangle
with left-down point (" + x + ";" + y
+ "), width: " + width + ", height: "
+ height);
    }
}
```

```
class RectangleAdapter implements Shape {
    private Rectangle adaptee;

    public RectangleAdapter(Rectangle rectangle)
{
        this.adaptee = rectangle;
    }

    @Override
    public void draw(int x1, int y1, int x2, int
y2) {
        int x = Math.min(x1, x2);
        int y = Math.min(y1, y2);
        int width = Math.abs(x2 - x1);
        int height = Math.abs(y2 - y1);
        adaptee.draw(x, y, width, height);
    }
}
```

Anything else?

- You could also call the adapter a “wrapper” around the old class.

- Don’t confuse Adapter with Proxy, Bridge, Facade etc.; recall that the adapter *adapts* an old interface to fit a new one, and nothing else.

(As for when *not* to use an Adapter, it’s pretty easy: when you don’t actually need to adapt old code)