GRASPing Object-Oriented Programming

Charlotte Pierce

SWINBURNE UNIVERSITY OF TECHNOLOGY

People have been using OOP for a while now…

Turns out they’ve learned some stuff along the way

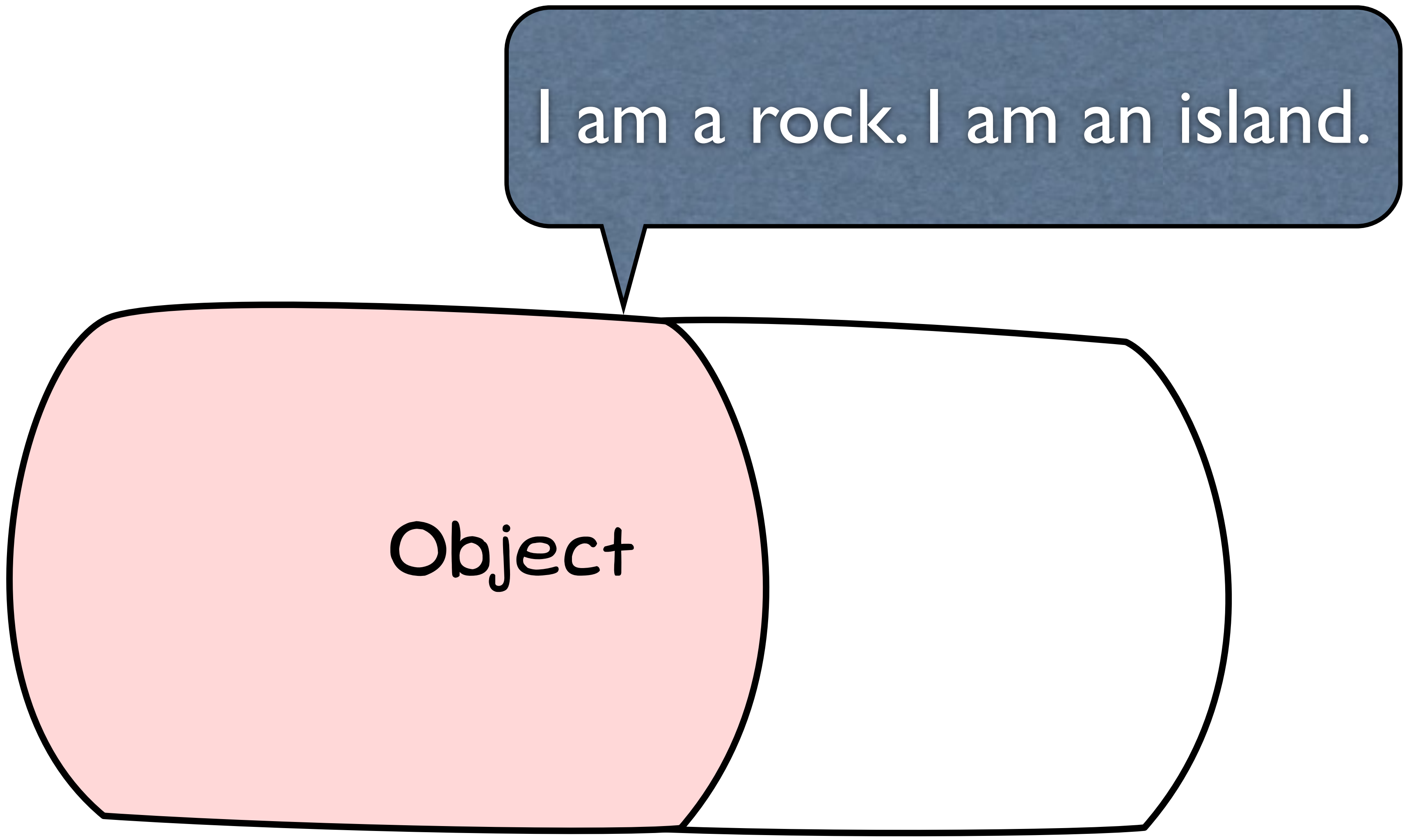
GRASP: General Responsibility Assignment Software Patterns

(a.k.a., how to make good design choices)

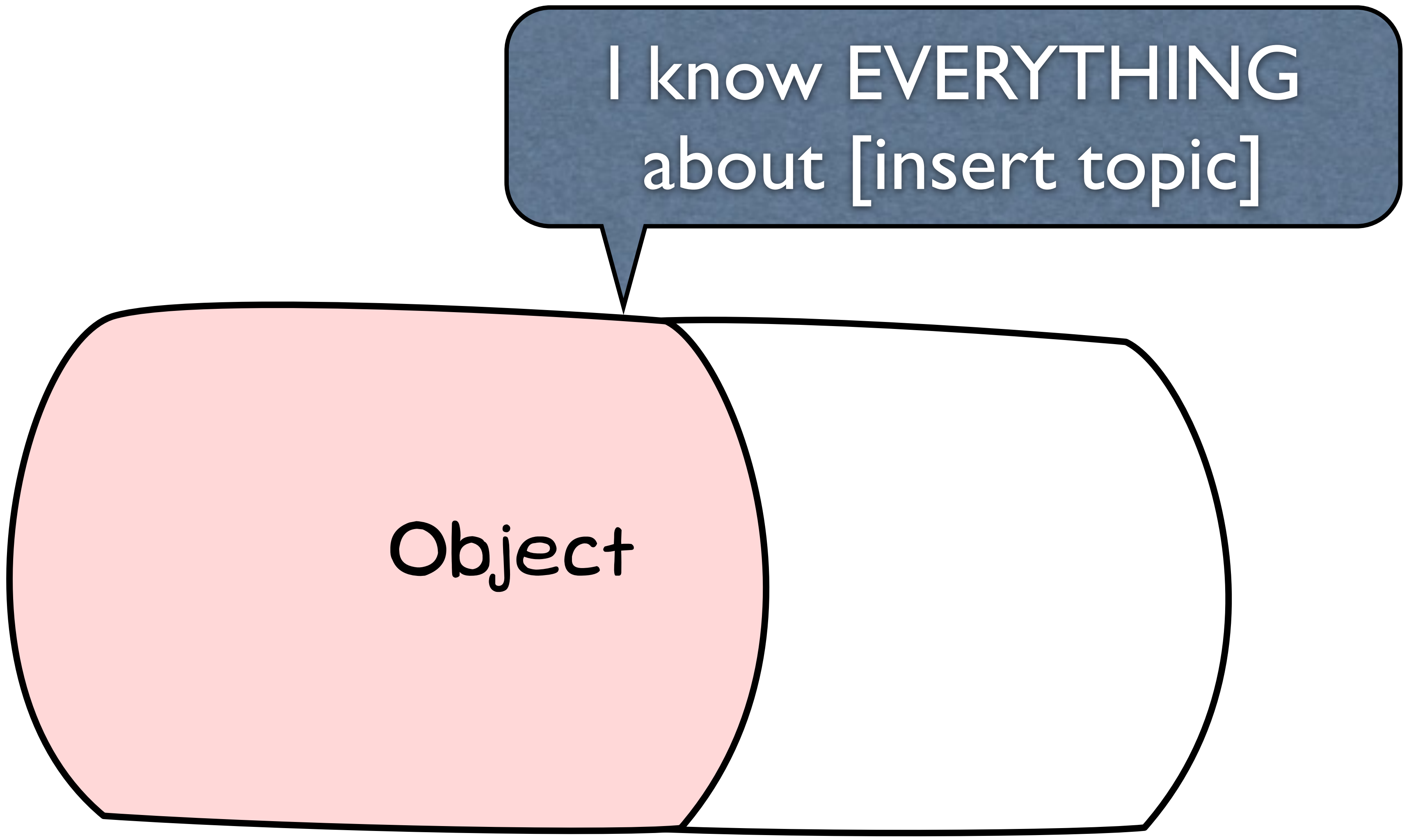
Software patterns provide optimised, reusable templates to solve problems

Good OO software classes should have **Low Coupling** and **High Cohesion**

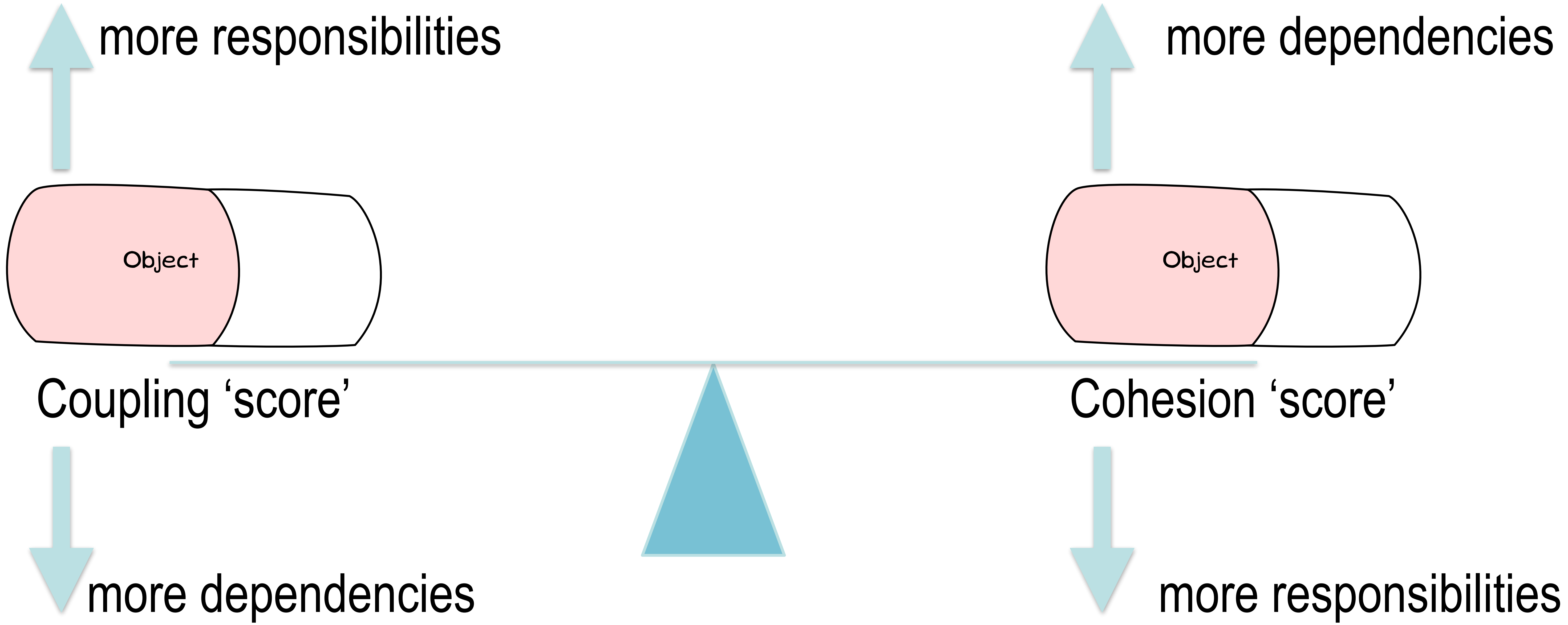
Classes with low coupling have few dependencies



Classes with high cohesion have strongly related responsibilities



Maintain a balance between coupling and cohesion

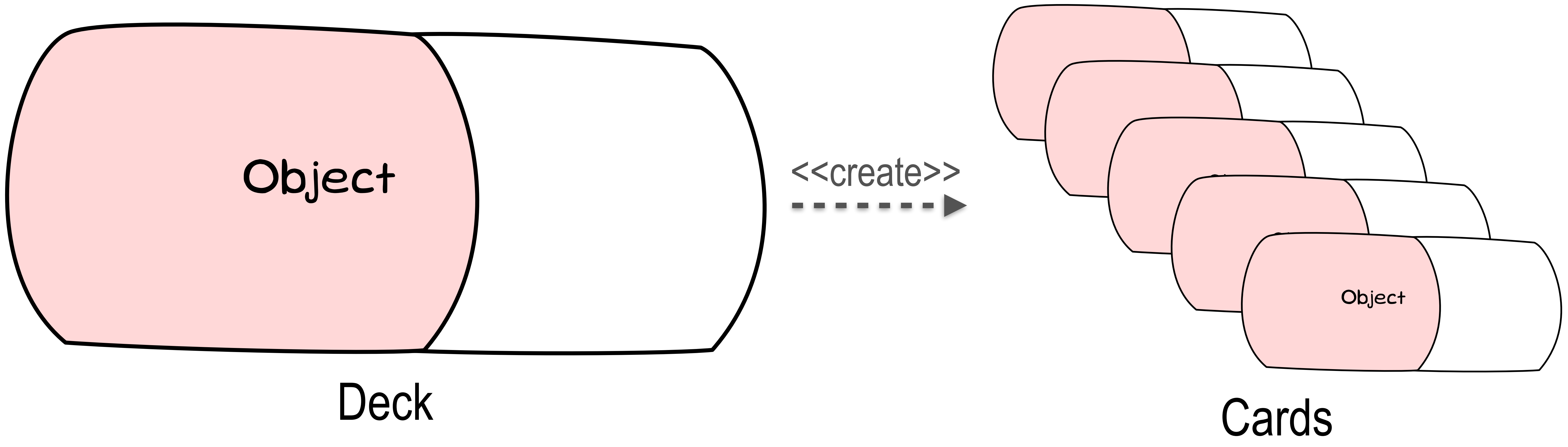


Coupling and cohesion apply at many levels

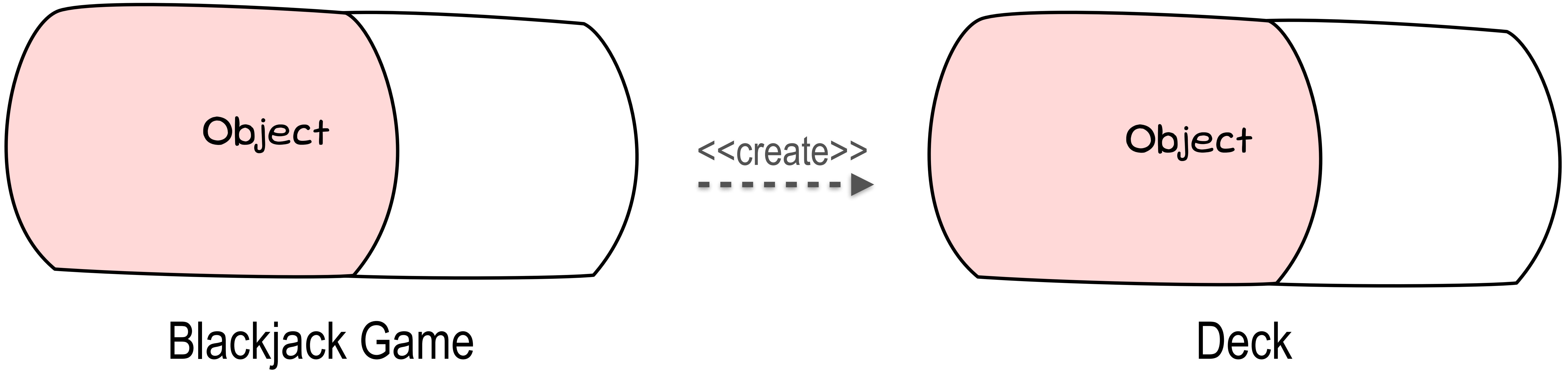
Assign responsibilities to the **Information Expert**

Use the **Creator** pattern to decide how to instantiate objects

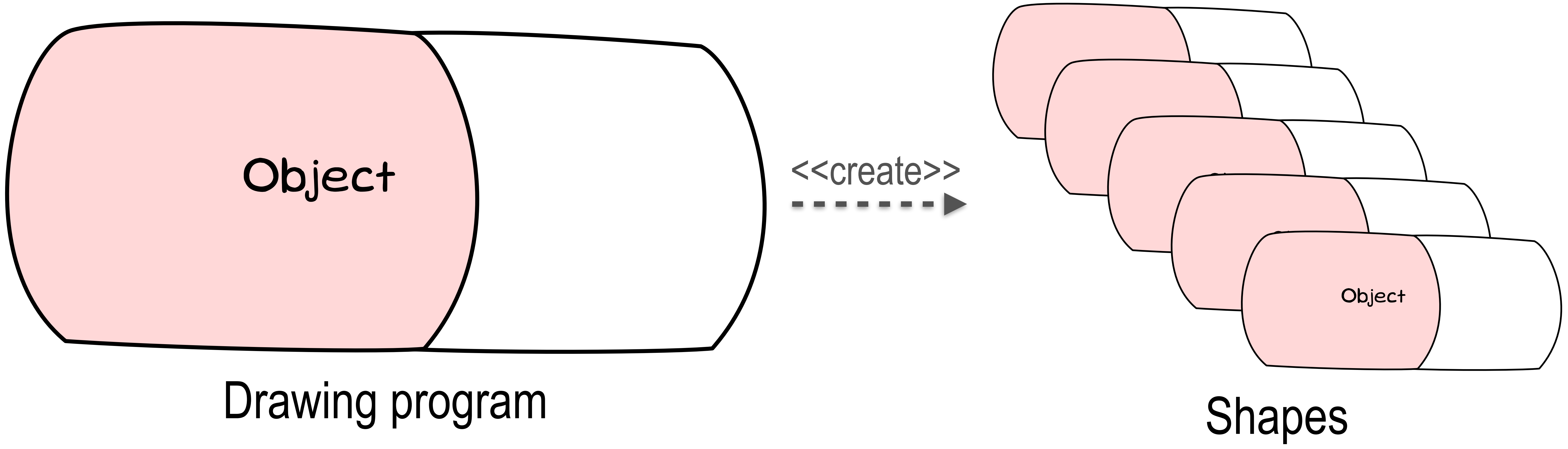
Who should create instances of class A?



Who should create instances of class A?



Who should create instances of class A?



Use **Polymorphism** to handle specialisations of a type

```
List<Shape> shapes
```

```
shapes.Add(new Rectangle(…))
shapes.Add(new Circle(…))
shapes.Add(new Line(…))
```

```
foreach shape s in
      shapes…
```

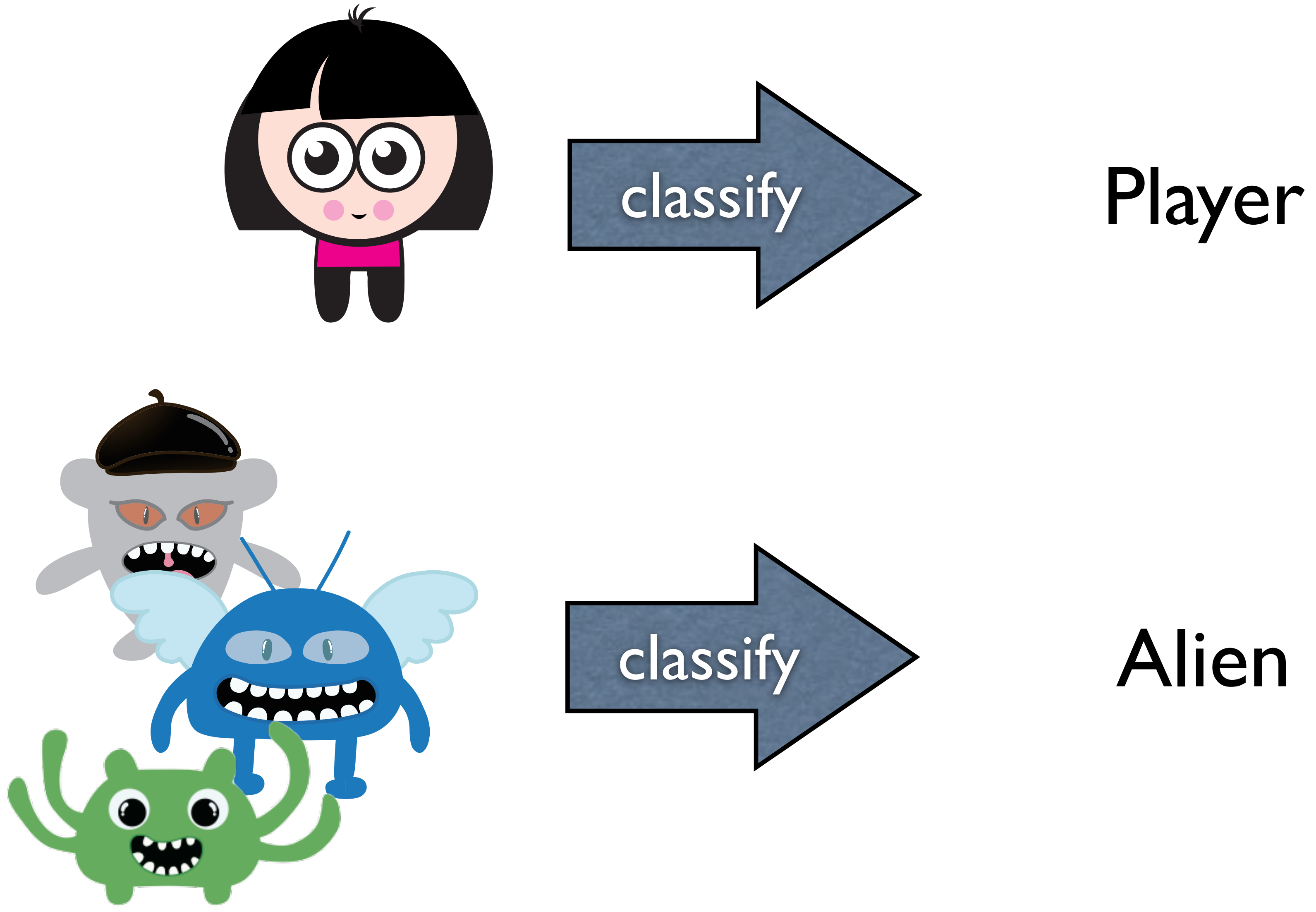
```
s.Draw()
```
$\longrightarrow$

```
s.Draw()
```
$\longrightarrow$

```
s.Draw()
```
$\longrightarrow$

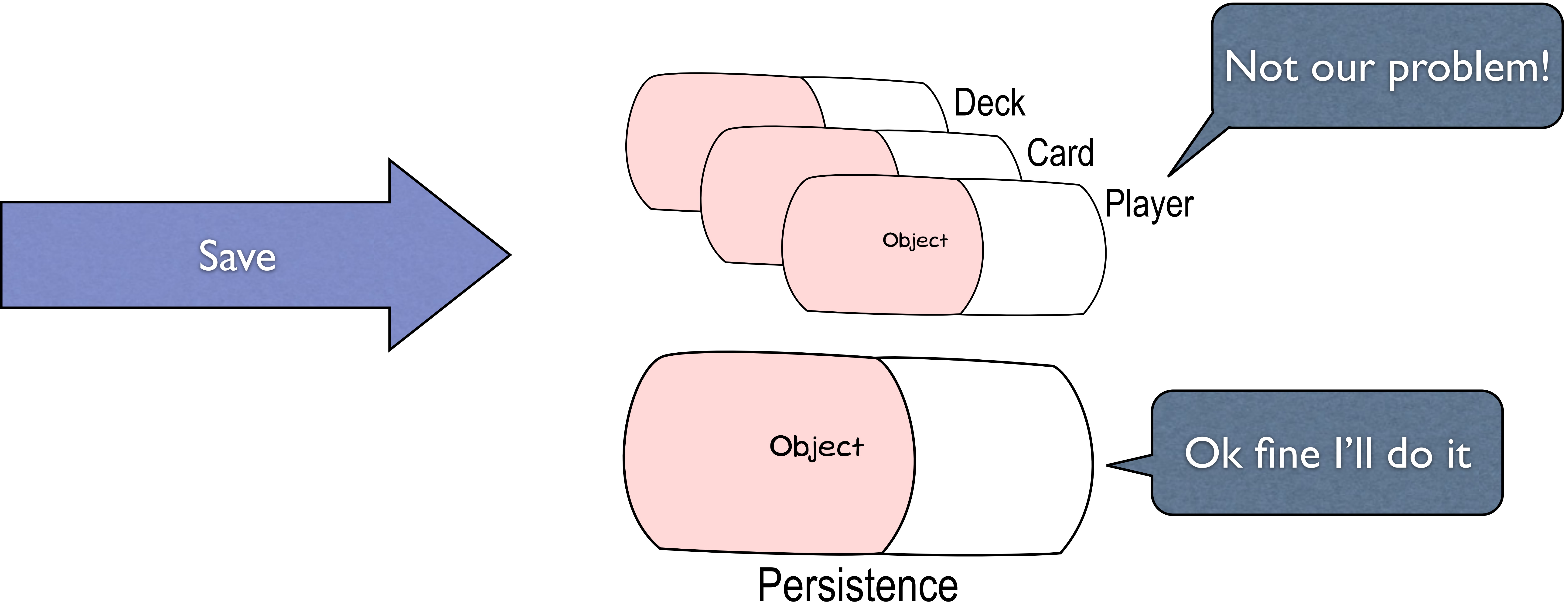
Remember this?

Use abstraction to classify the different kinds of roles objects will play in your software



Use Abstraction
(Classification)
to define object
classes

Use **Pure Fabrication** when real-world concepts aren't enough



Rules can be broken…

Use GRASP to help make good design decisions

There is more!