Inheritance and Polymorphism

Charlotte Pierce

SWINBURNE UNIVERSITY OF TECHNOLOGY

Object oriented programs contain objects that know and can do things

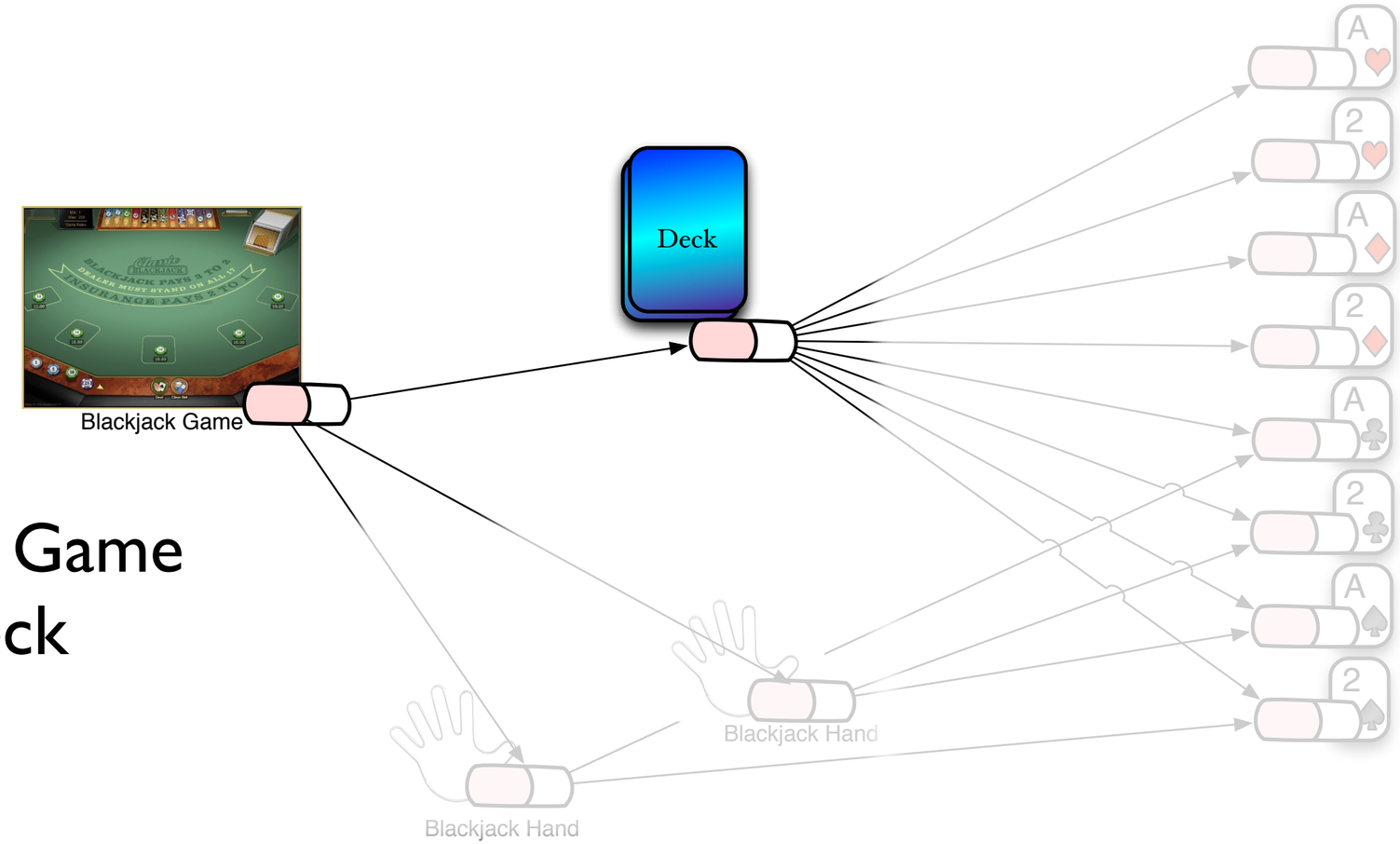
Remember there are three main kinds of relationships

Association

Aggregation

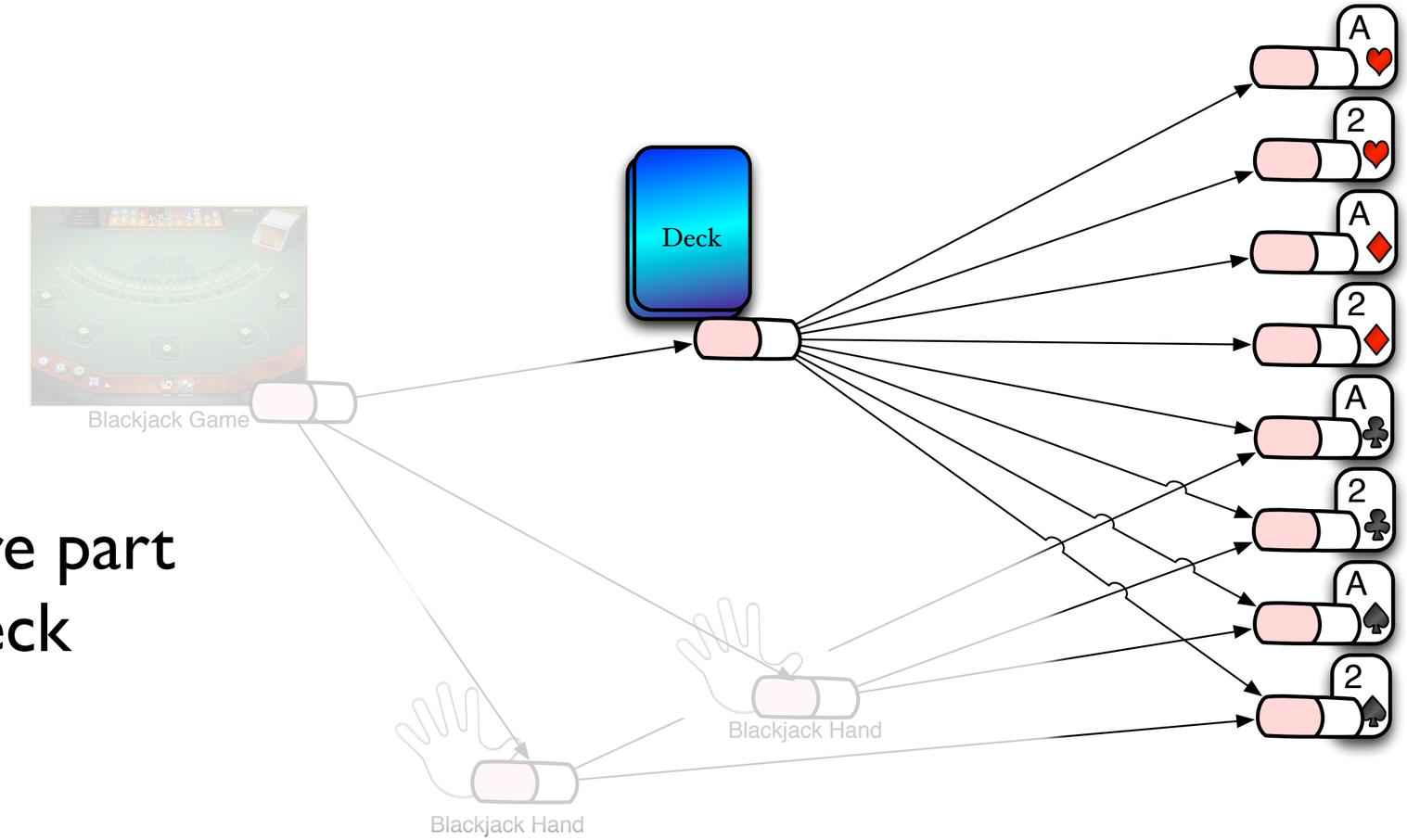
Dependence

Use association for "has-a" kind relationship



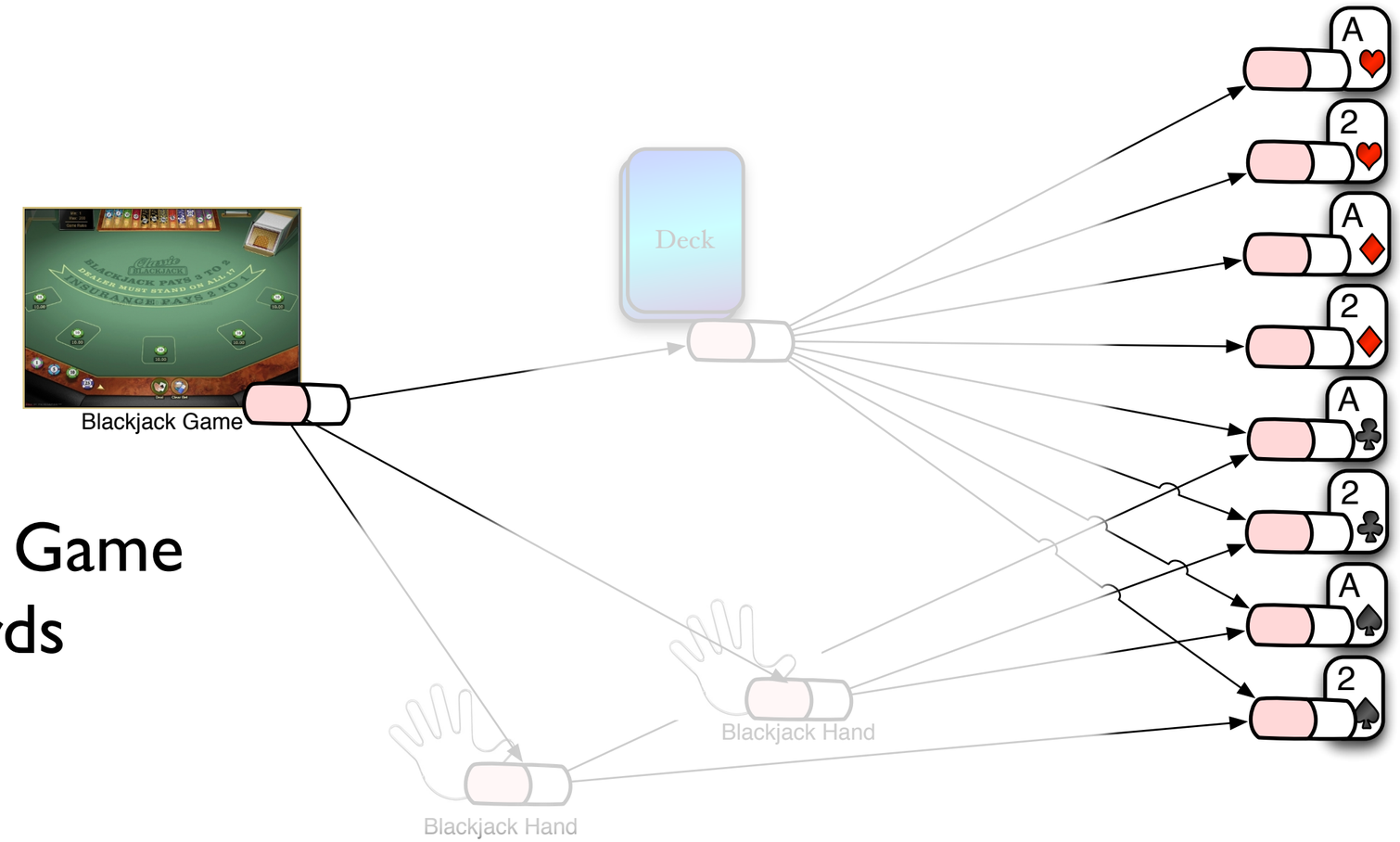
The Blackjack Game
has a Deck

Use "aggregation" for whole-part or container relationships



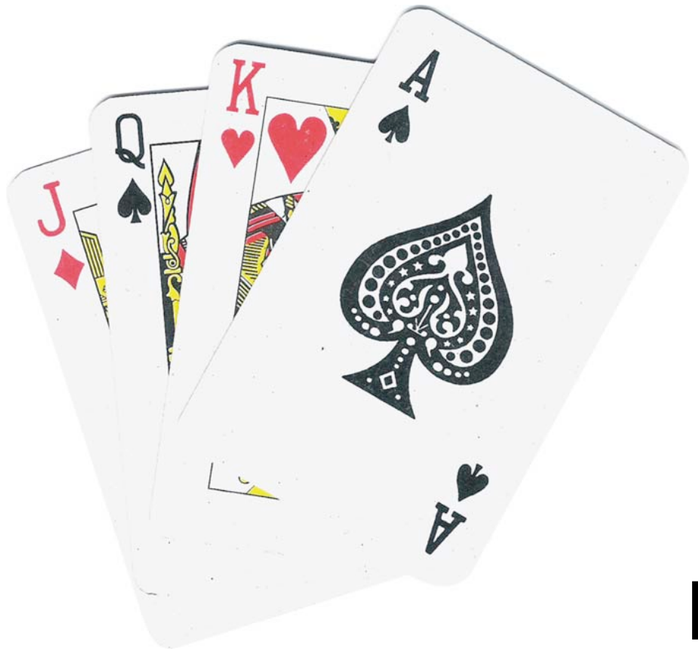
The Cards are part of the Deck

Use dependency for temporary "uses" style relationships



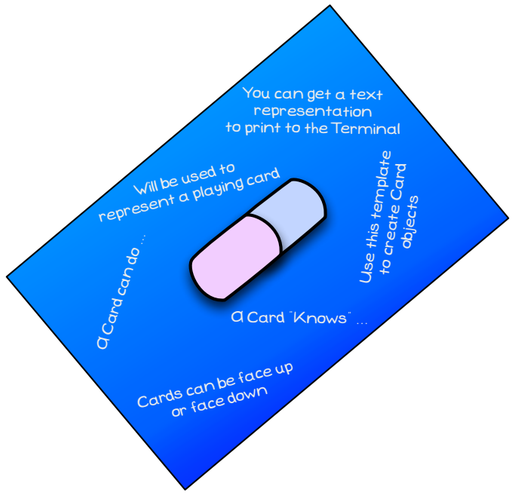
The Blackjack Game **uses** Cards

Developers use the process of abstraction to define object classes

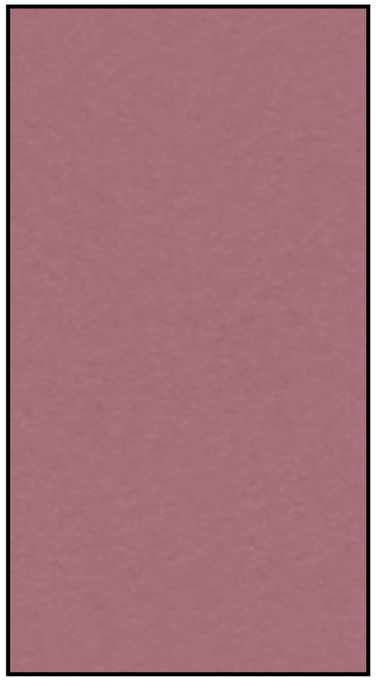


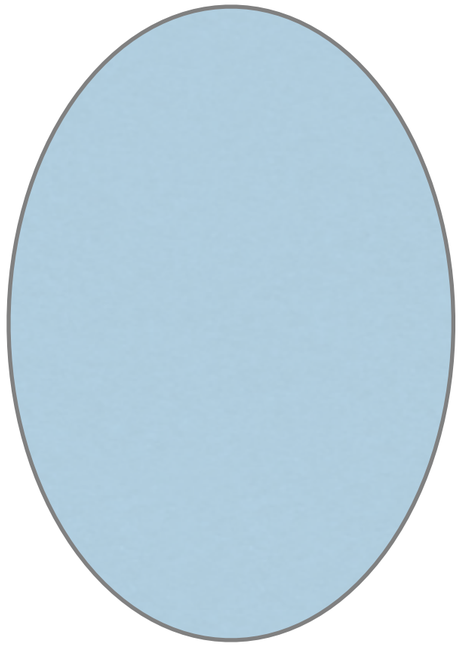
Classification

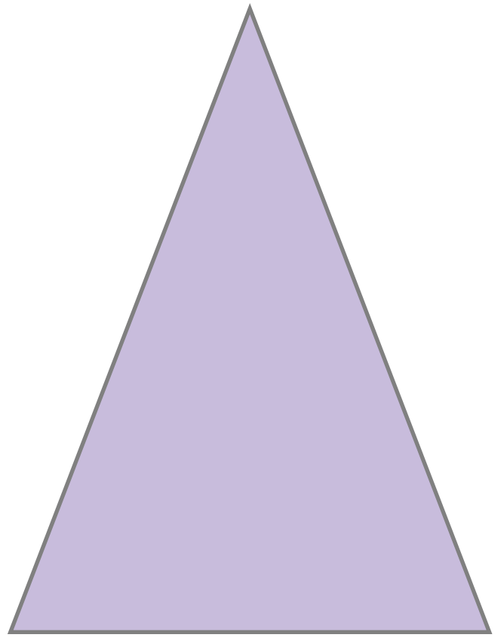
Specification for a Card



Abstraction also includes generalisation and specialisation





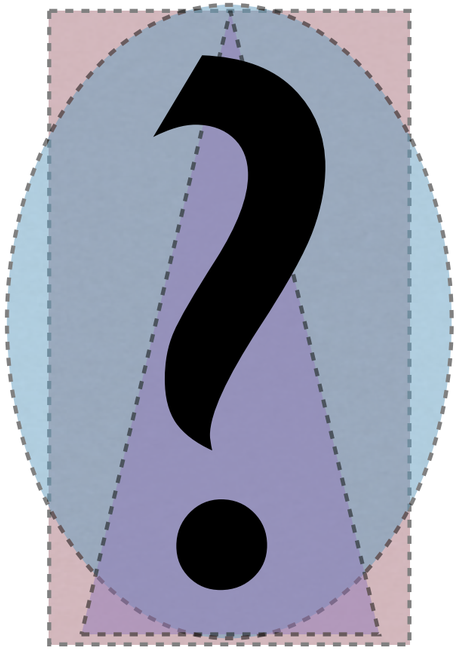


What are these?

Use generalisation and specialisation to create families of classes

What do you want to do with shape?

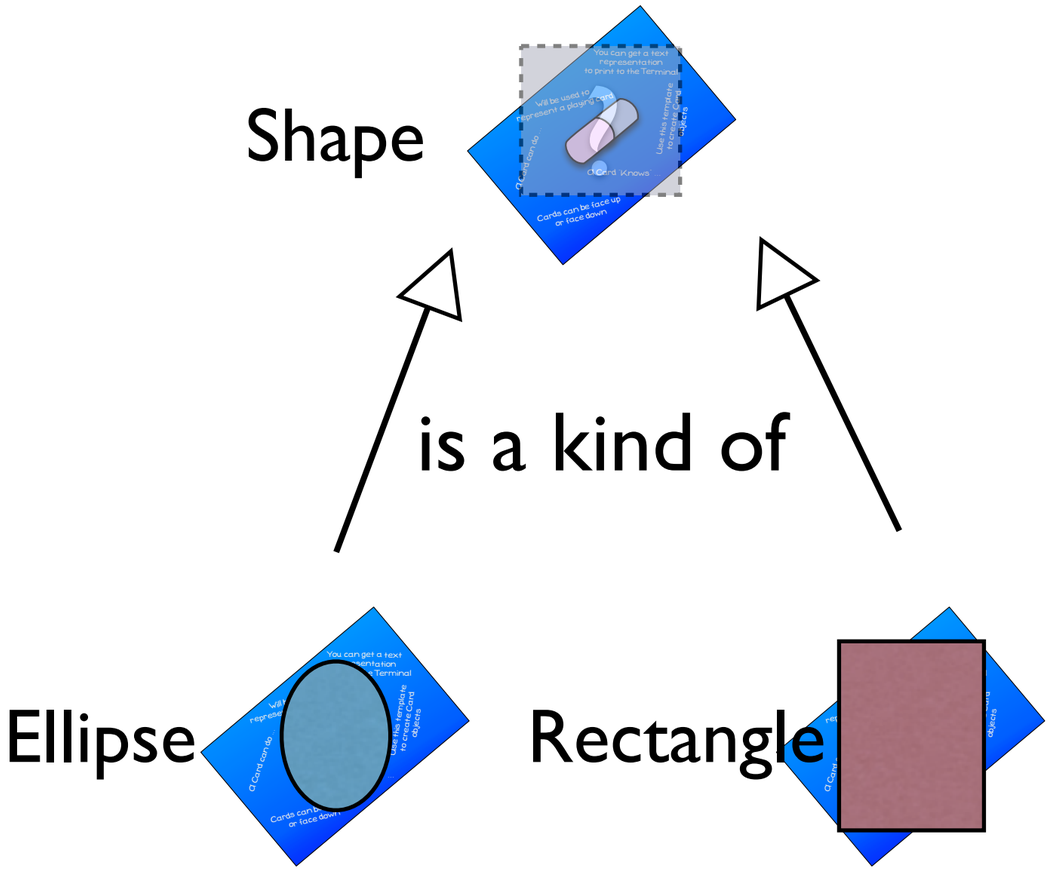
Do you care if they are ellipses, rectangles, triangles?



Use inheritance to model generalisation and specialisation in your OO code

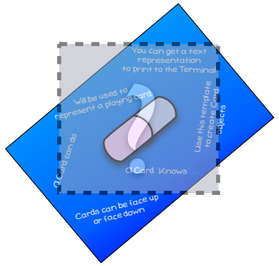
Inherit attributes and behaviour from a **parent** class

Inheritance models **is-a** kind of relationships



The child class **inherits** all of the features of the parent...

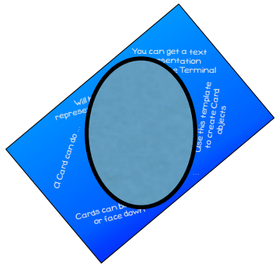
Shape



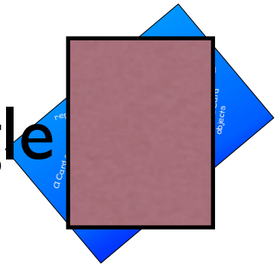
Has a position, size, and can be drawn.

is a kind of

Ellipse

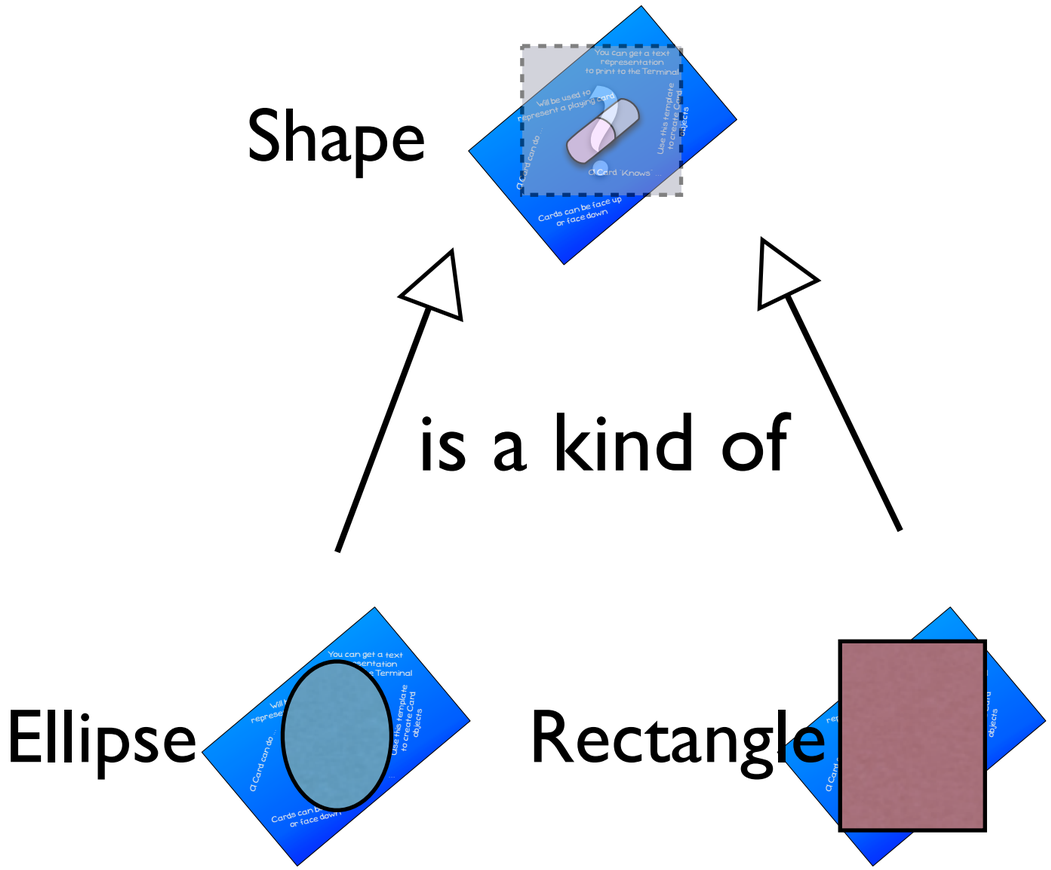


Rectangle



inherits the position, size, and can be drawn.

Change how **inherited methods** behave in the child class (overriding the parent)

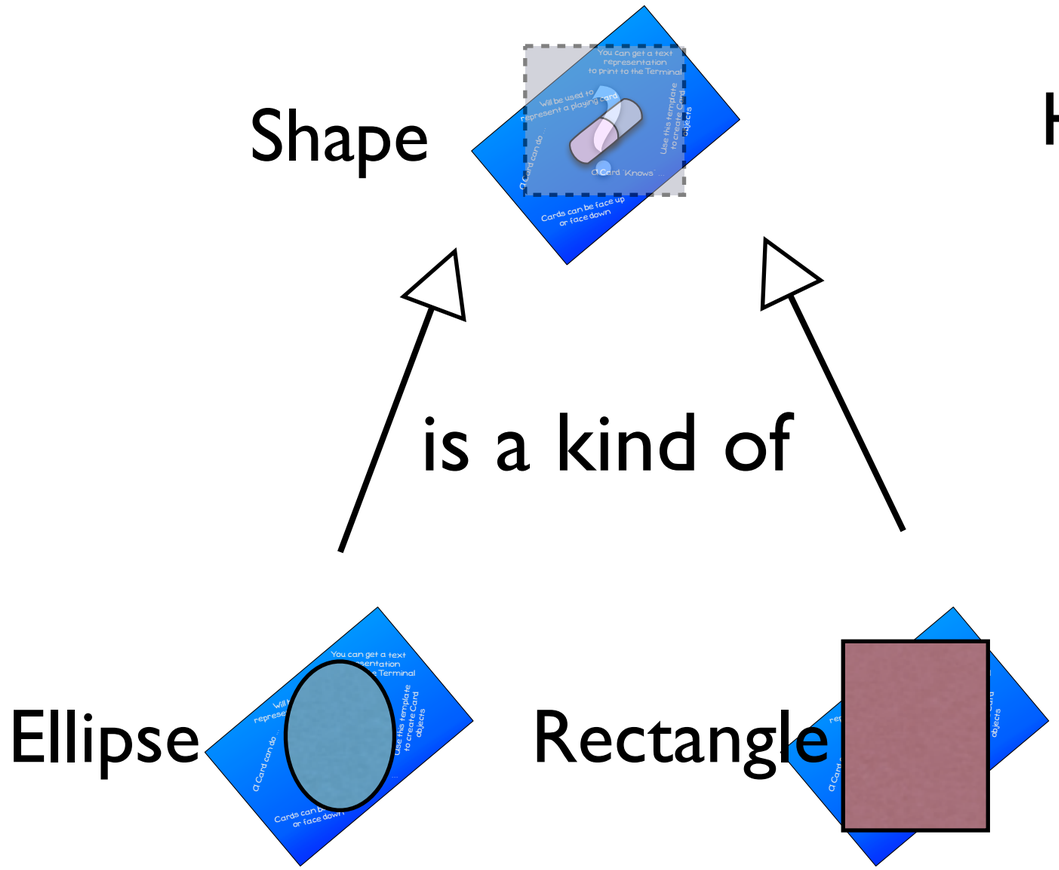


Has a position, size, and **can be drawn**.

Rectangle = Draws a Rectangle

Ellipse = Draws an Ellipse

Add additional features in the child class

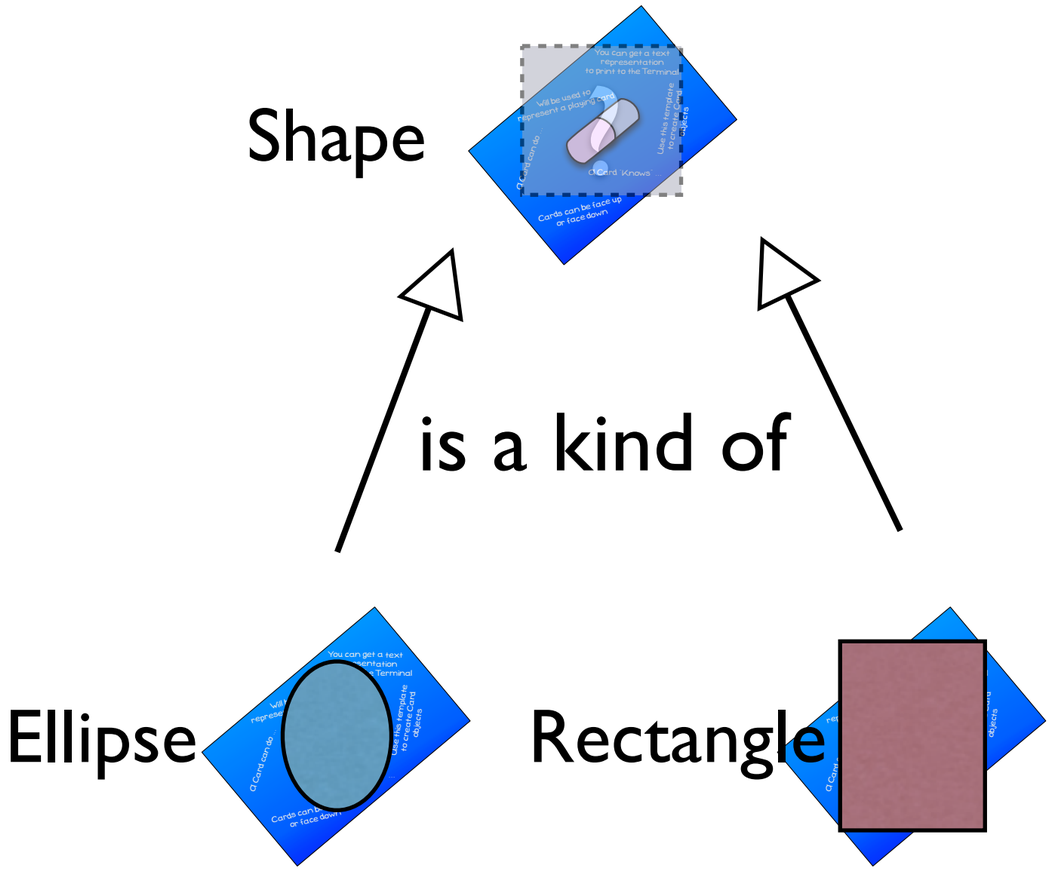


Has a position, size, and **can be drawn**.

Rectangle = Make Square

Ellipse = Make Circular

The child class can see public and protected members of the parent



Access levels

public: anyone
protected: only derived classes
private: nobody else

Inheritance declared by **derived classes**

C++

class Rectangle : public Shape

C#

class Rectangle : Shape

Java

class Rectangle extends Shape

Objective-C

@interface Rectangle : Shape

Use child objects where the parent is expected

Refer to an object using any of the classes it **is a** kind of

```
Object o
```

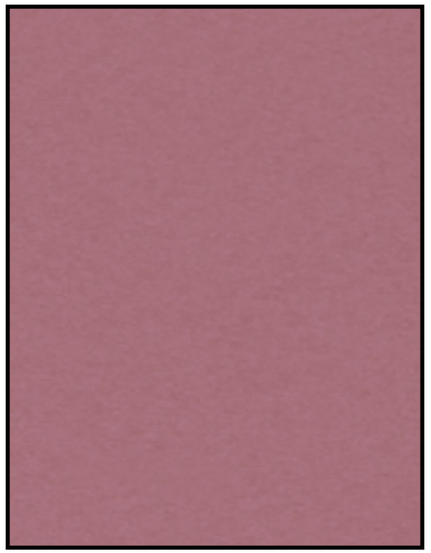
Does o refer to an object?

```
Shape s
```

Does s refer to a shape?

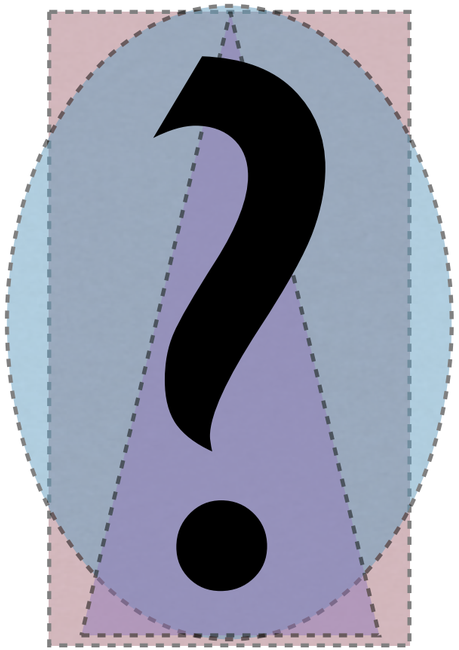
```
Rectangle r
```

Does r refer to a rectangle?



Objects behave based on their **actual** class!

`Shape s` → What will be drawn?

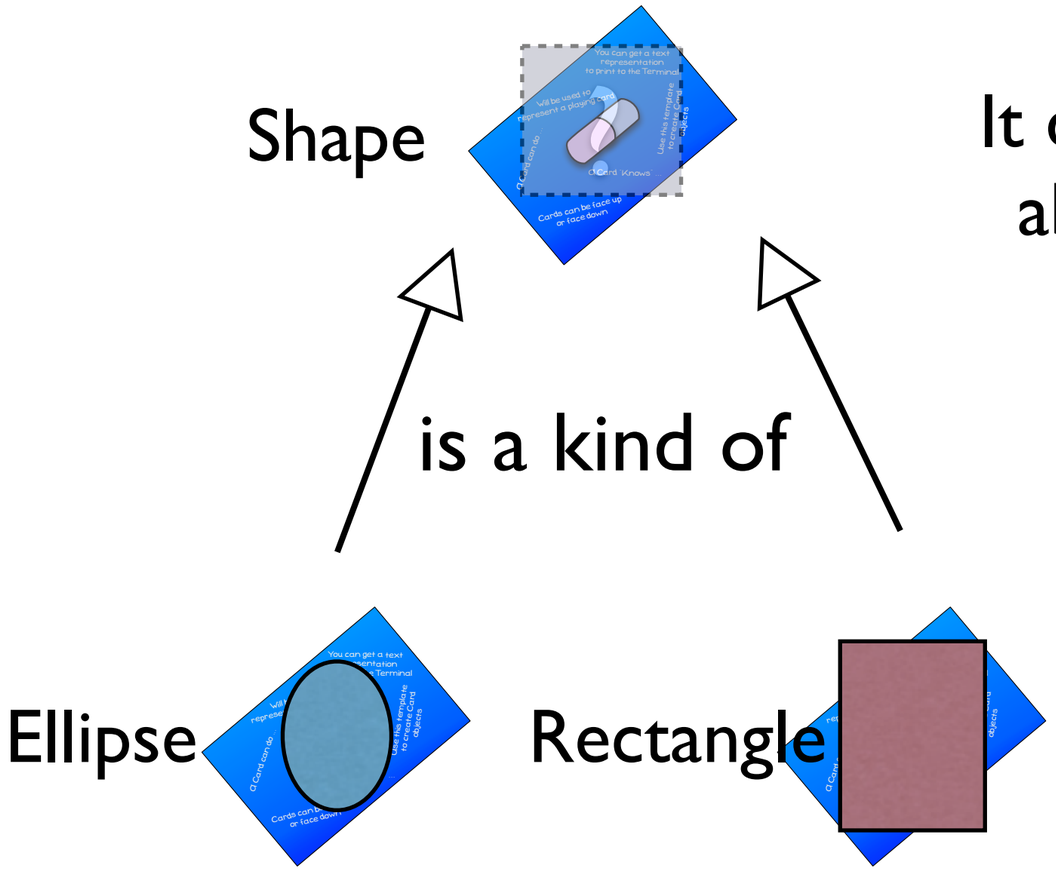


This is called **polymorphism**

Poly Morph

Many Forms

Parent classes can provide *placeholder* methods that **must** be overridden



How does Shape Draw?
It doesn't; Draw is a placeholder = **abstract**
abstract classes **cannot create objects**

Rectangle must override draw

Ellipse must override draw

Abstract methods of **base classes**

C++

virtual void draw () = 0;

C#

public abstract void Draw();

Java

public abstract void draw();

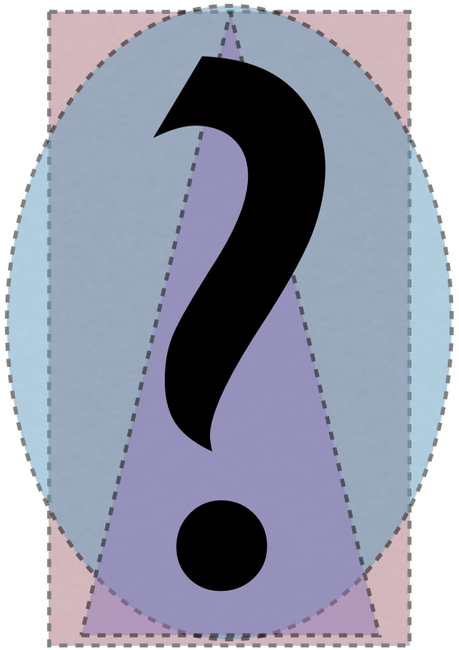
Objective-C

- (void) draw;

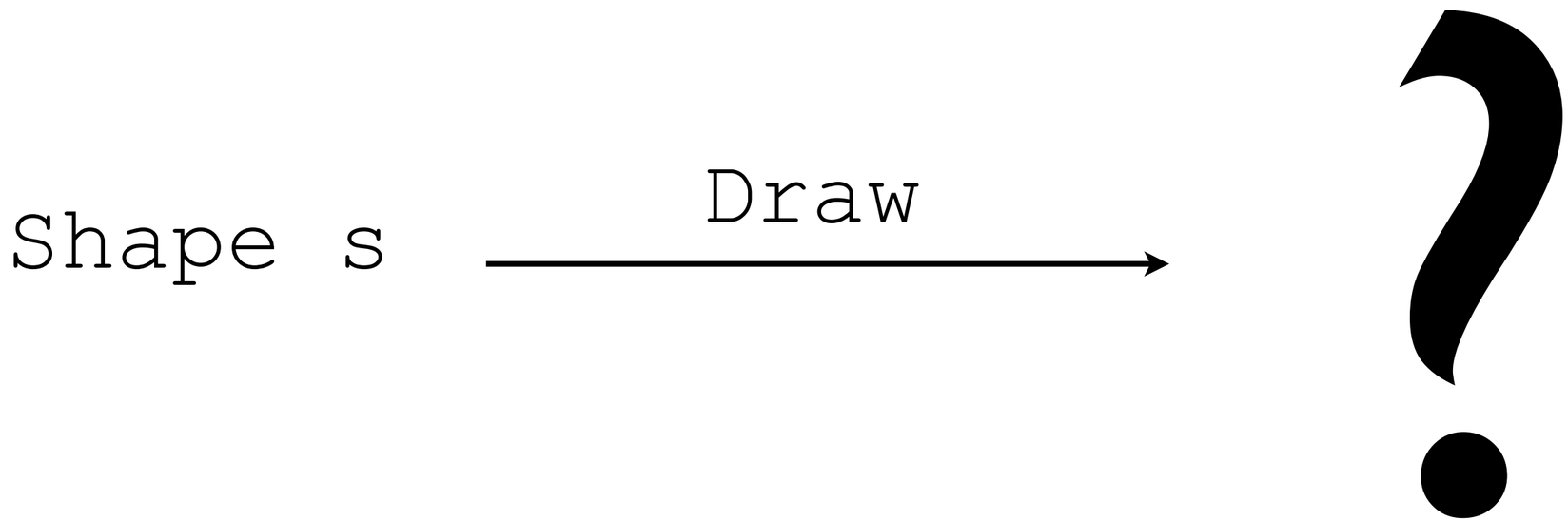
Inheritance and polymorphism help development

Flexibility: Refer to a parent class, but get child objects… they work as expected!

`Shape s` → `Draw`

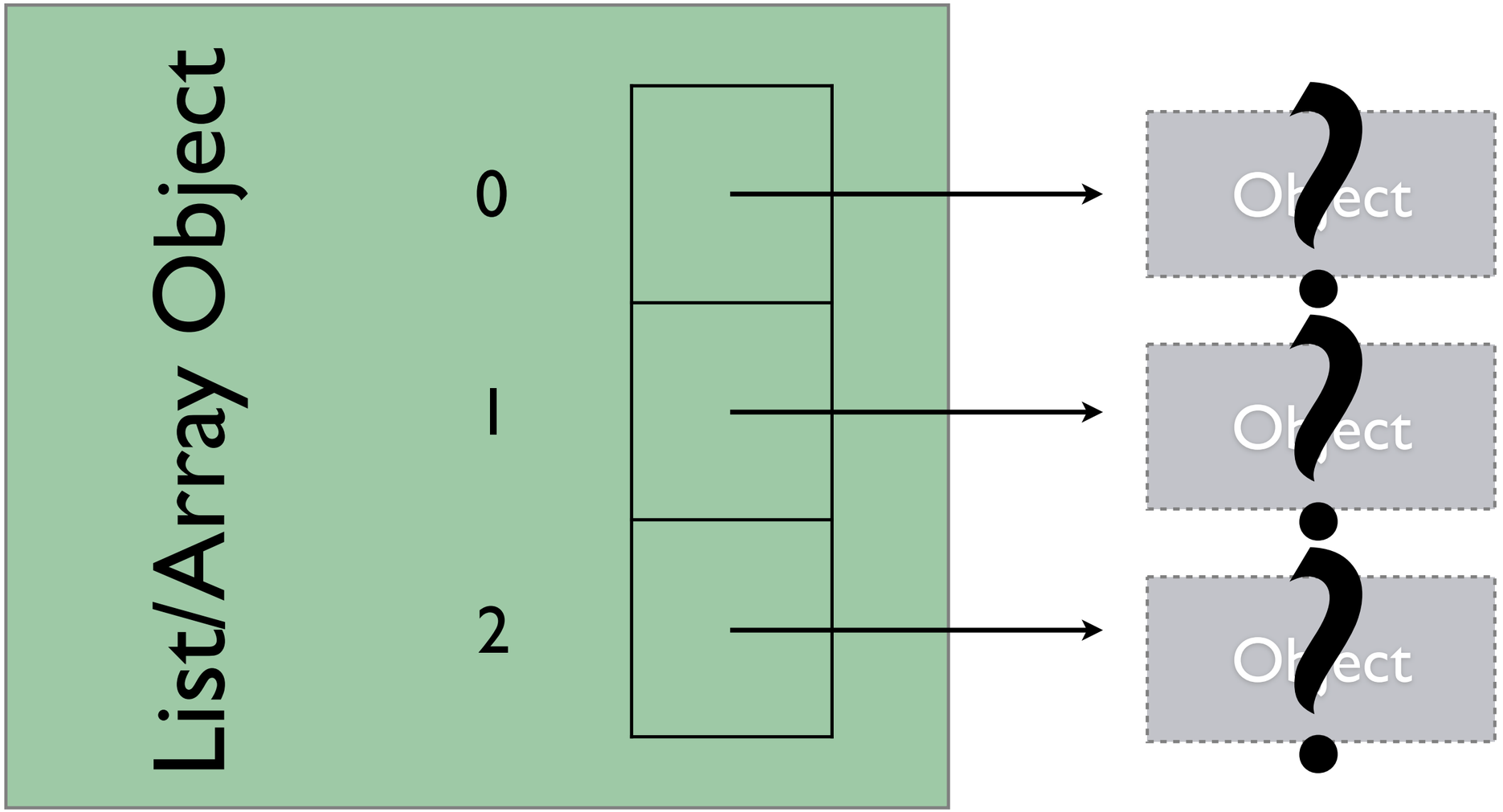


Extensible: add new children without needing to change uses

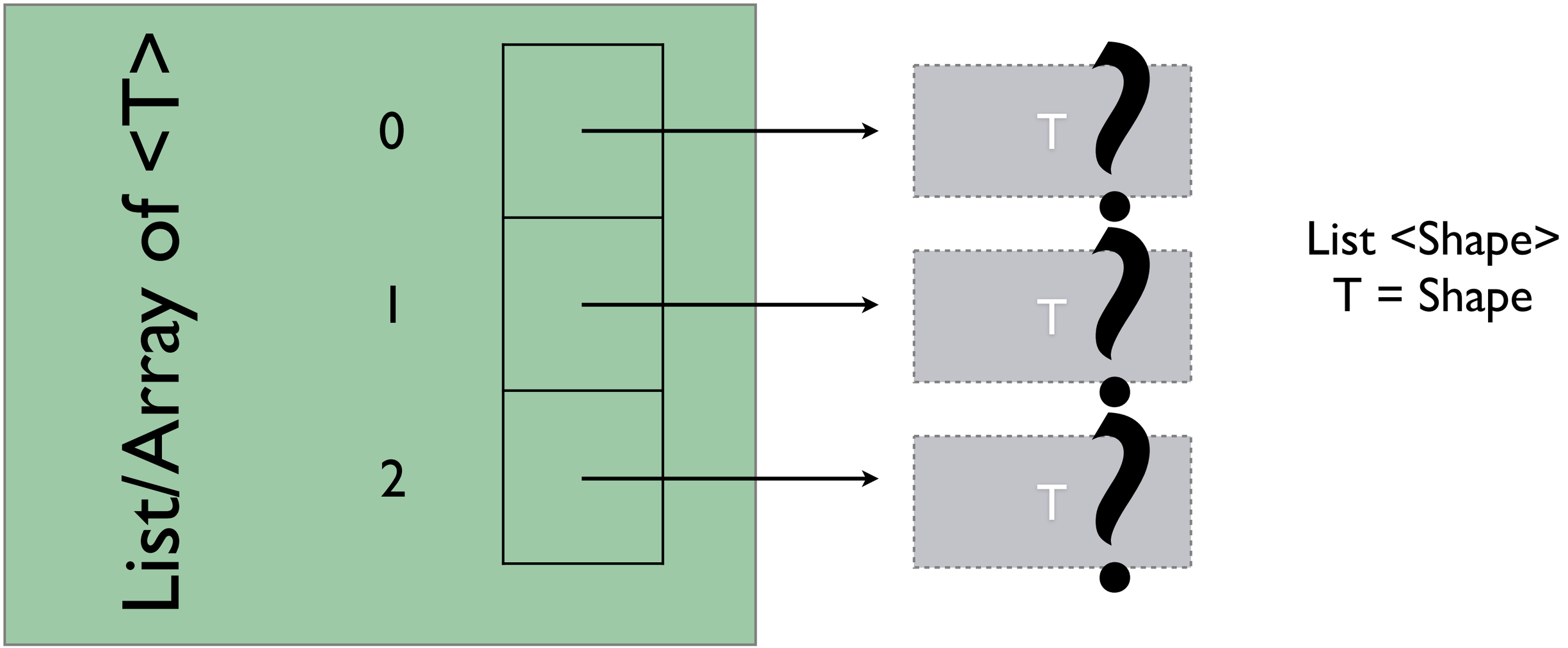


Can it only be Rectangle/Triangle/Ellipse?

Adaptable: Utilities like collection classes can work on Objects



Languages extend these capabilities with generics



Will inheritance help you create object oriented programs?

Abstraction is much more than just classification

Use inheritance to model generalisation and specialisation in your OO code

Inheritance helps bring flexibility, extensibility, and adaptability to your OO programs

Demo - Using Inheritance and Polymorphism