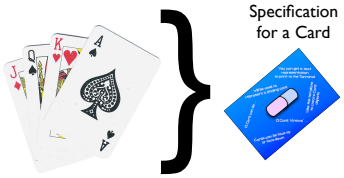
Inheritance and Polymorphism

by Andrew Cain and Willem van Straten

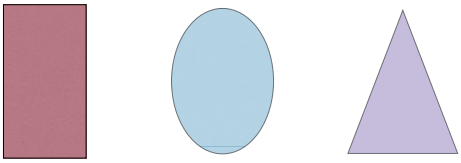
Object Oriented Programming

Object oriented programs contain objects that know and can do things

Developers use the process of abstraction to define object classes



Abstraction also includes generalisation and specialisation

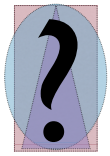


What are these?

Use generalisation and specialisation to create families of classes

What do you want to do with shape?

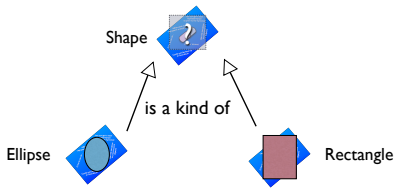
Do you care if they are ellipses, rectangles, triangles?



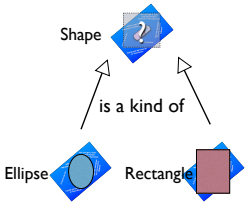
Use inheritance to model generalisation and specialisation in your OO code

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

Inheritance models **is-a** relationships



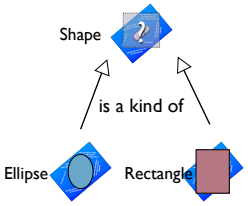
The child class **inherits** all of the features of the parent...



Has a position, size, and can be drawn.

inherits the position, size, and can be drawn.

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



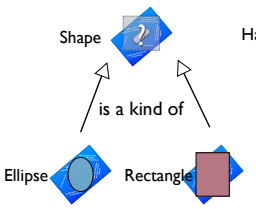
Has a position, size, and **can be drawn**.

Rectangle = Draws a Rectangle

Ellipse = Draws an Ellipse

Role-play: Shapes in action

Add additional features in the child class

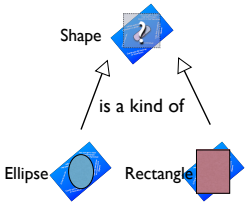


Has a position, size, and **can be drawn**.

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

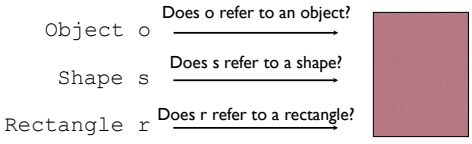
C#
class Rectangle : Shape

Java
class Rectangle extends Shape

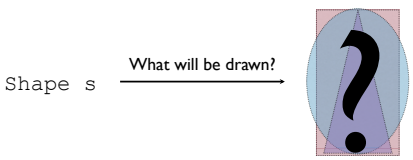
Objective-C
@interface Rectangle : Shape

Use child objects where the parent is expected

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



Watch objects behave based on their actual class!

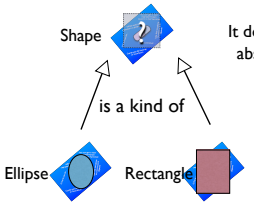


This is called **polymorphism**

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;

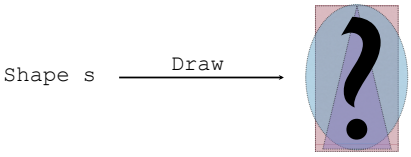
C#
public abstract void Draw();

Java
public abstract void draw();

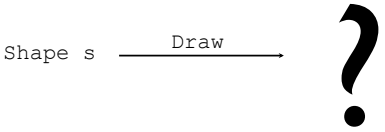
Objective-C
- (void) draw;

See how inheritance and polymorphism lead to good design

Flexible: Ask the parent to do it, don't worry about which child does it

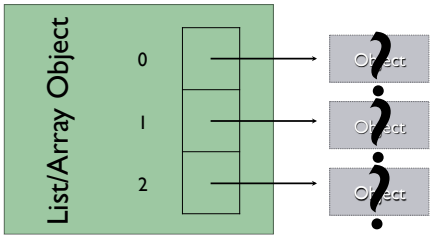


Extensible: add new children without needing to change uses

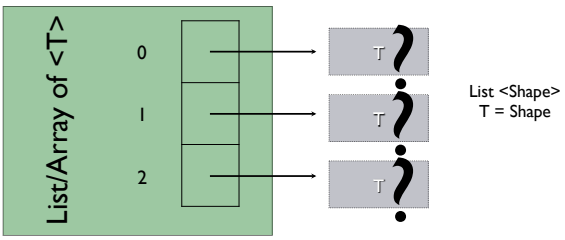


Can it only be Rectangle/Triangle/Ellipse?

Adaptable: Utilities like collection classes can work on Objects



Languages extend these capabilities with generics/templates



Three kinds of polymorphism

1. Parametric: generics & templates
2. Ad hoc: function & operator overloading
3. **Subtype: abstract class / interface**

Parametric polymorphism

```
template <class T>
T max (T a, T b)
{
  if (a > b)
    return a;
  else
    return b;
}
```

```
int a = 3;
int b = 4;
int c = max (a, b);

float f = 5.6;
float g = 7.8;
float h = max (f, g);
```

Ad hoc polymorphism

```
String a = "Hello ";
String b = "World";
String c = a + b;    // concatenates the strings

float f = 5.6;
float g = 7.8;
float h = f + g;     // computes the sum
```

Sub-type polymorphism

```
public abstract class Shape
{
  public abstract void Draw ();
}

public class Rectangle : Shape
{
  public override void Draw ()
  {
     // draw a rectangle
  }
}
```

Will inheritance and polymorphism help you create object oriented programs?

Abstraction is much more than just classification

Use inheritance to model generalisation and specialisation in your OO code

Polymorphism brings flexibility, extensibility, and adaptability to your OO programs

This Week's Tasks

Pass Task 11 - Shape Drawer
Pass Task 12 - The Spell Book