Anglia Ruskin University

Inheritance

- Principle technique of OO programming

  (Property of any OO language)

- Promotes software re-usability

  (Avoids having to re-write similar code)

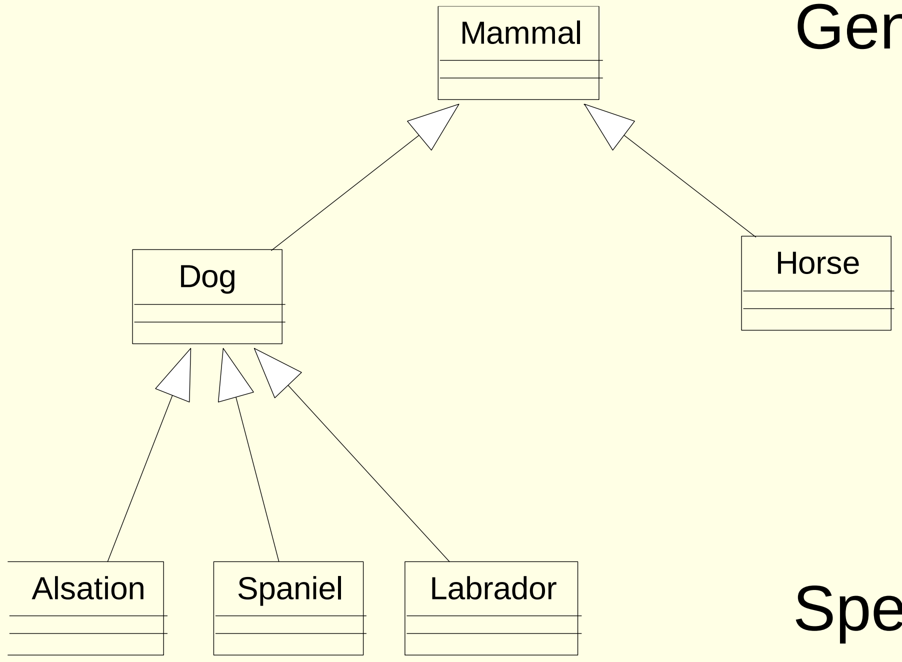
- Promotes high level of abstraction

  (Makes complicated things appear simple)

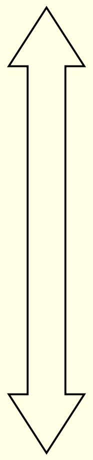
e.g. Mammal classification scheme

Base class or
Superclass

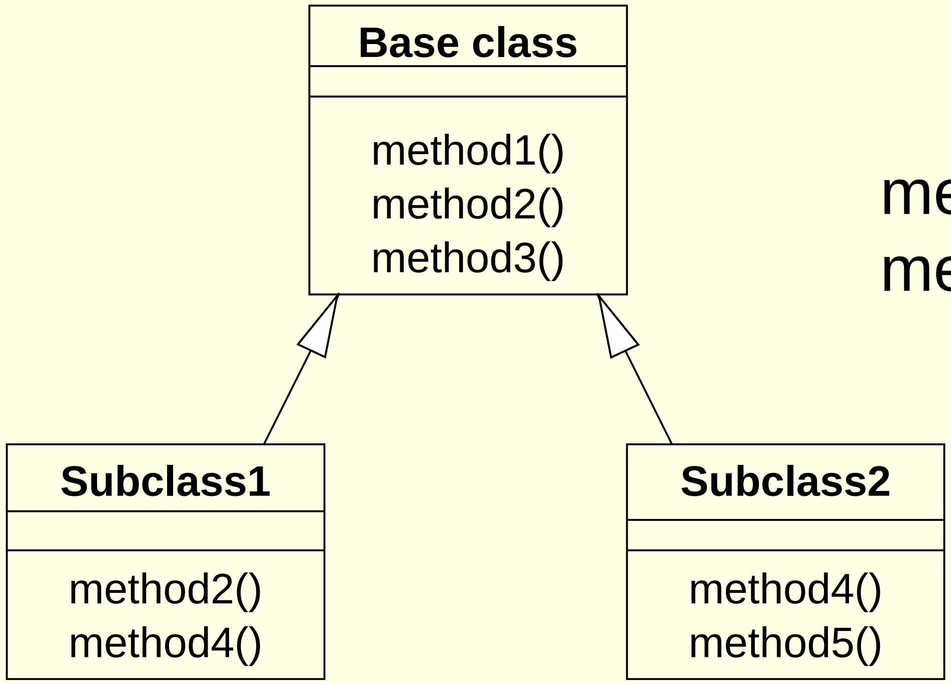
Subclass or
Derived class



Generalization



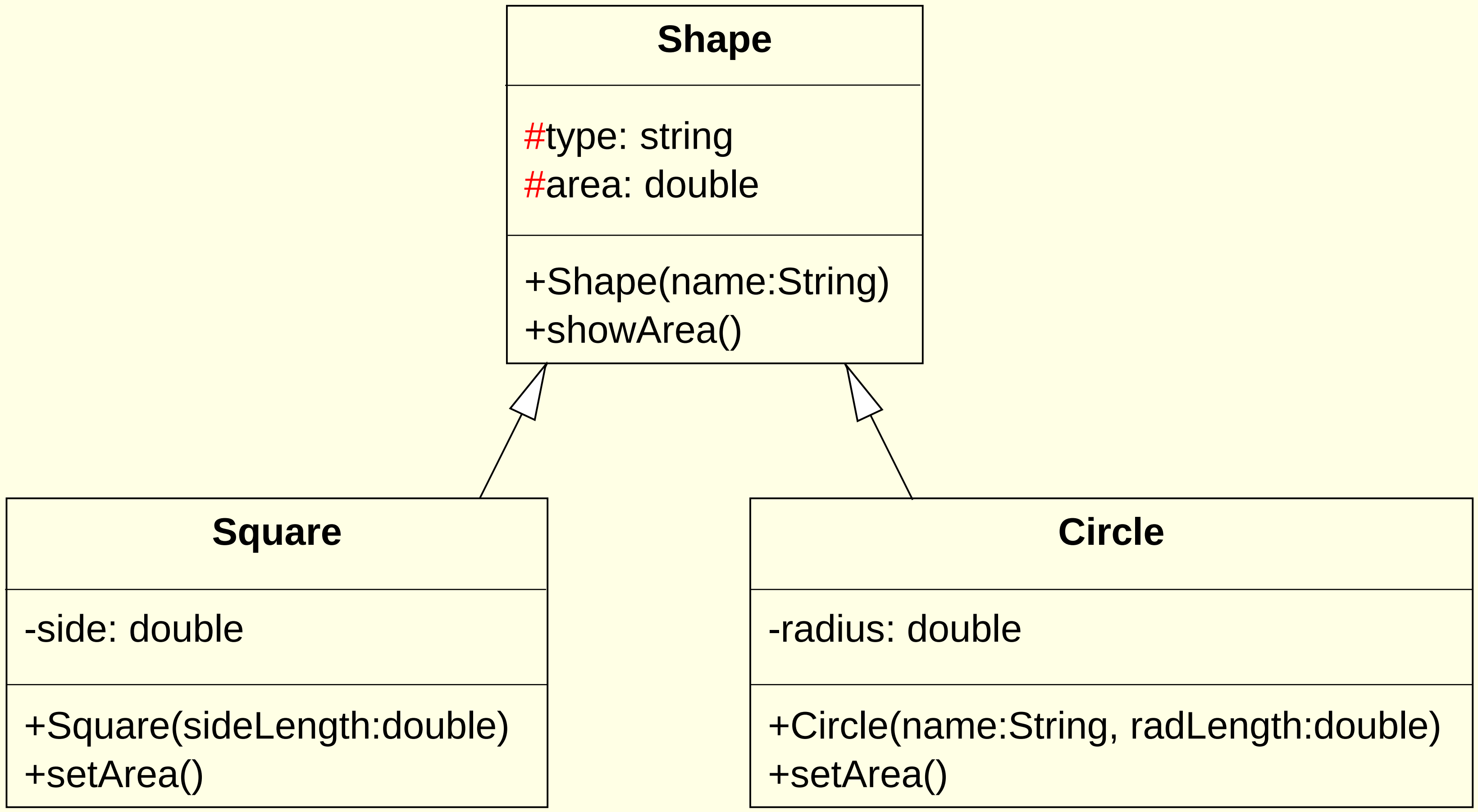
Specialization



methods 2 = over-riding
methods 4 = polymorphism

- **Base class can access methods 1, local 2, and 3**
- **Subclass1 can access methods 1, local 2, 3, and local 4**
- **Subclass2 can access methods 1, inherited 2, 3, local 4, and 5**

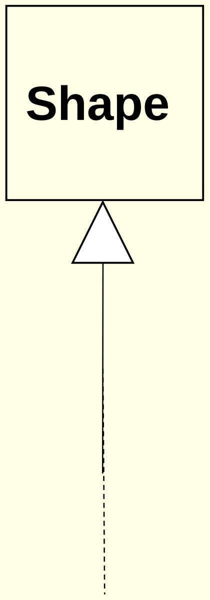
EG class **Square** and class **Circle** inherit from class **Shape**



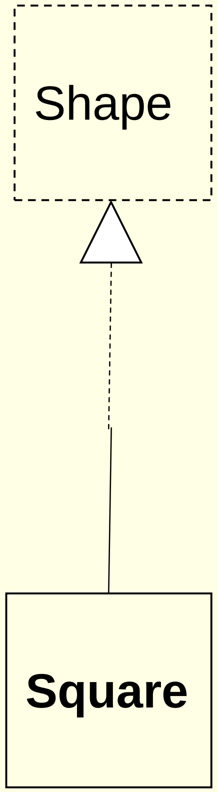
Anglia Ruskin University

The Base class

```
class Shape
{
   protected string type;     // holds description of shape
   protected double area;           // stores area of shape

   public Shape(string name)
   {
      type = name;
      area = 0;
   }

   public void showArea()
   {
      if( (int)area == 0 )
         Console.WriteLine(“Area of “ + type + “ is undefined”);
      else
         Console.WriteLine(“Area of “ + type + “ is “ + area);
   }
}
```

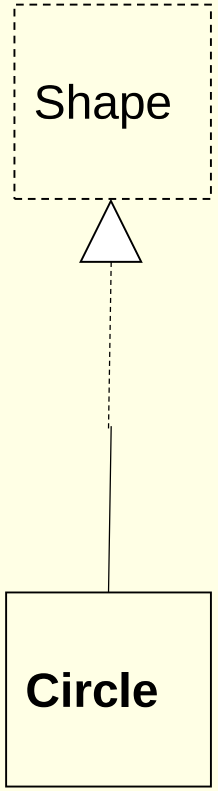


The Square class



```
class Square : Shape
{
    private double side;

    public Square(double sideLength ) : base( “Square”)
    {                                           // call to parent
                                                // class constructor
       side = sideLength;      // set local subclass attribute
    }

    public void setArea()
    {
       area = side * side;      // set inherited attribute
    }
}
```

The Circle class



```
class Circle : Shape
{
    private double radius;

    public Circle( string name, double radLength) : base(name)
    {                                                   // call to parent
                                                        // class constructor
      radius = radLength;      // set local subclass attribute
    }

    public void setArea()
    {
      area = 3.14259 * radius * radius;     // set inherited
    }                                       // attribute
}
```

Anglia Ruskin University

A driver class

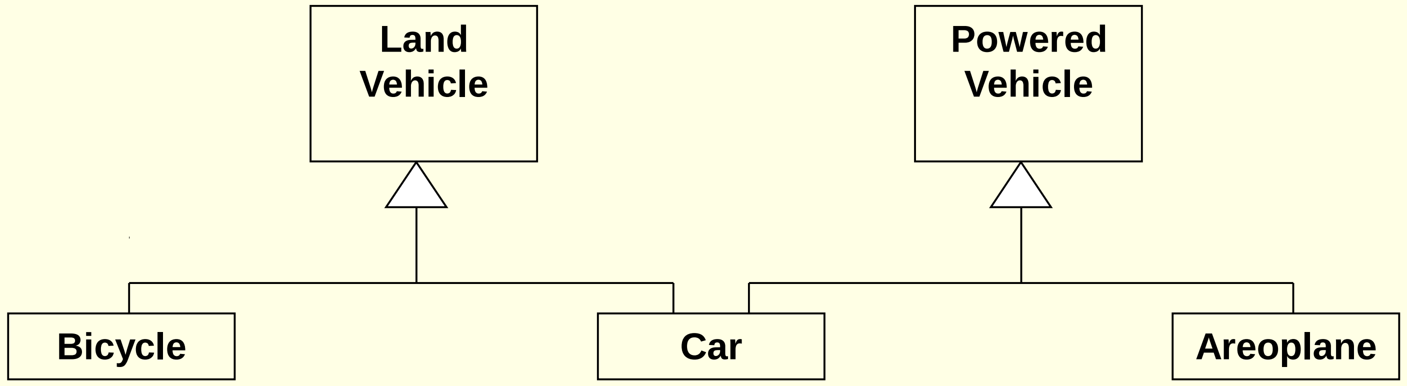
```
class TestShapes
{
  public static void Main(string[ ] args)
  {
    Shape first = new Shape( “Blob“ );
    Square second = new Square( 4.0 );
    Circle third = new Circle( “Circle”, 3.0 );

    second.setArea();
    third.setArea();

    first.showArea();

    second.showArea();
    third.showArea();
  }
}
```

$$\left.\begin{array}{l}\texttt{second.setArea();}\\ \texttt{third.setArea();}\end{array}\right\}\ \text{Polymorphism}$$

$$\left.\begin{array}{l}\texttt{second.showArea();}\\ \texttt{third.showArea();}\end{array}\right\}\ \text{Method inheritance}$$

Anglia Ruskin University

Multiple inheritance - subclasses inherit from more than one base class



A limited kind of multiple inheritance is allowed in C# using an INTERFACE class

```
interface Myclass
{
   …
}
```

```
class Aclass : Bclass, Myclass
{
   …
}
```