Chapter 10 – Object-Oriented Programming: Polymorphism

10.1 Introduction
10.2 Derived-Class-Object to Base-Class-Object Conversion
10.3 Type Fields and `switch` Statements
10.4 Polymorphism Examples
10.5 Abstract Classes and Methods
10.6 Case Study: Inheriting Interface and Implementation
10.7 `sealed` Classes and Methods
10.8 Case Study: Payroll System Using Polymorphism
10.9 Case Study: Creating and Using Interfaces
10.10 Delegates
10.11 Operator Overloading

◀ ▶

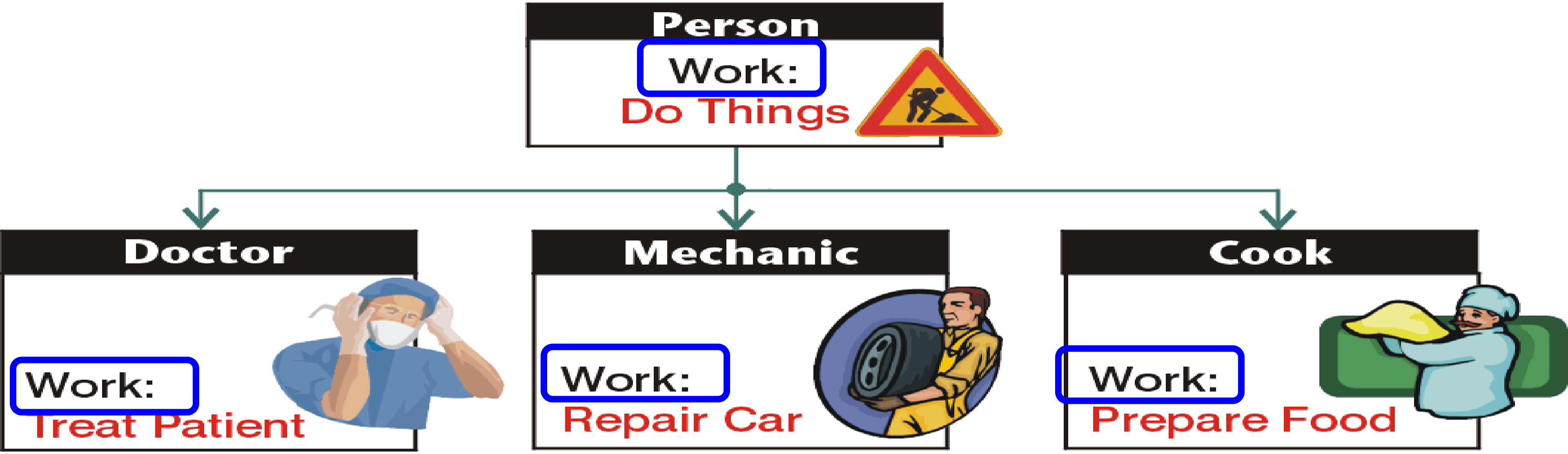
Introduction to Polymorphism

- Polymorphism
  - "Program in the general"
  - Treat objects in a same class hierarchy as an object of their superclass
  - Makes programs extensible if not know what the subclass will be
    - New subclasses added easily, no need to modify the old program
  - General type (範型) is a outstanding usage of polymorphism (to be discussed in data structure course)

Polymorphism

Polymorphism

Same Operation, Performed Differently



Person

Work:

Do Things

Doctor

Work:

Treat Patient

Mechanic

Work:

Repair Car

Cook

Work:

Prepare Food

- Polymorphism is the ability of objects belonging to different classes to perform the operation with the same name but different contents.
- Programs that handle a wide variety of related classes in a generic manner

Dynamic Method Binding

- Dynamic method binding
  - Since base-class reference variable can refer to derived-class object
    - When a method of a base-class reference variable is called,
      - the method can be dynamically bound to the method of the suitable derived class
  - That is, program itself can choose "correct" method in derived class

Derived-Class-Object to Base-Class-Object Conversion

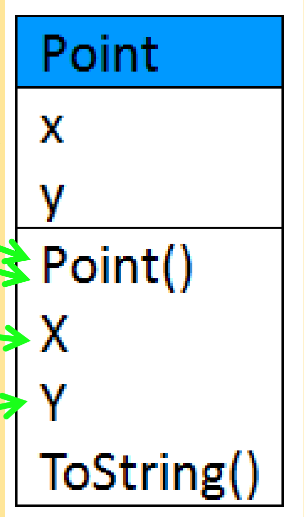
- Key concept
  - Derived-class object can be treated as base class-object (蔡五可以視為蔡家子孫)
  - But base-class object is not a derived-class object (蔡家子孫並不一定是蔡五)
- Assignment of derived and base classes
  - We can assign derived-class object to base-class reference
  - But we can not assign base-class object to derived-class reference
    - If needed, assignment operator should be cast to allow such an assignment

10.2 Derived-Class-Object to Base-Class-Object Conversion

- Downcasting a pointer
- 降級指標
  - Use an explicit cast to convert a base-class pointer to a derived-class pointer
  - 必須明確指出使用降級指標 將指到原始門派的指標改成指到衍生門派的指標
  - Format:

```
derivedPtr = (DerivedClass) basePtr;
```

```
// Fig. 9.4: Point.cs
    using System;
    public class Point   {
        private int x, y;
        public Point()     {
        }
        public Point( int xValue, int yValue )  {
            X = xValue;
            Y = yValue;
        }
        public int X    {
            get   {
                return x; }
            set {
                x = value; }
         }
        public int Y {
            get {
                return y; }
            set {
                y = value; }
        }
```



Point

x

y

Point()

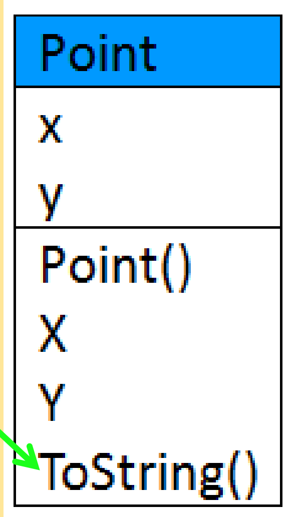
X

Y

ToString()

```
    public override string ToString() {
        return "[" + X + ", " + Y + "]";
    }
}
```

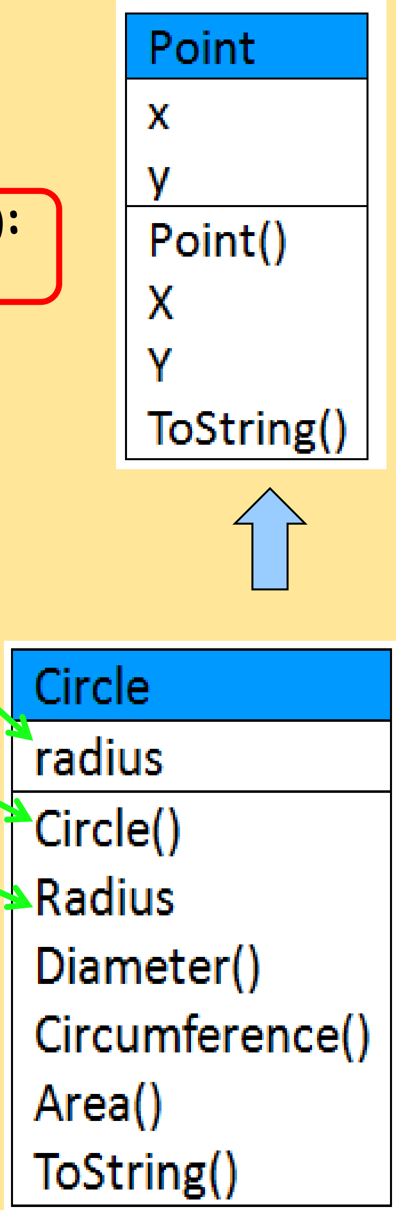
Override its base method (here the method did not call its base method)



```
// Fig. 9.5: PointTest.cs
using System;
public class Circle : Point {
   private double radius;
   public Circle() {
      // implicit call to Point constructor occurs here
   }
   public Circle(int xValue, int yValue, double radiusValue ):
                base( xValue, yValue) {
      Radius = radiusValue;
   }
   public double Radius  {
      get {
         return radius; }
      set {
         if ( value >= 0 )
            radius = value; }
   }
```

Inheritance syntax

Explicitly call its base constructor with two parameter

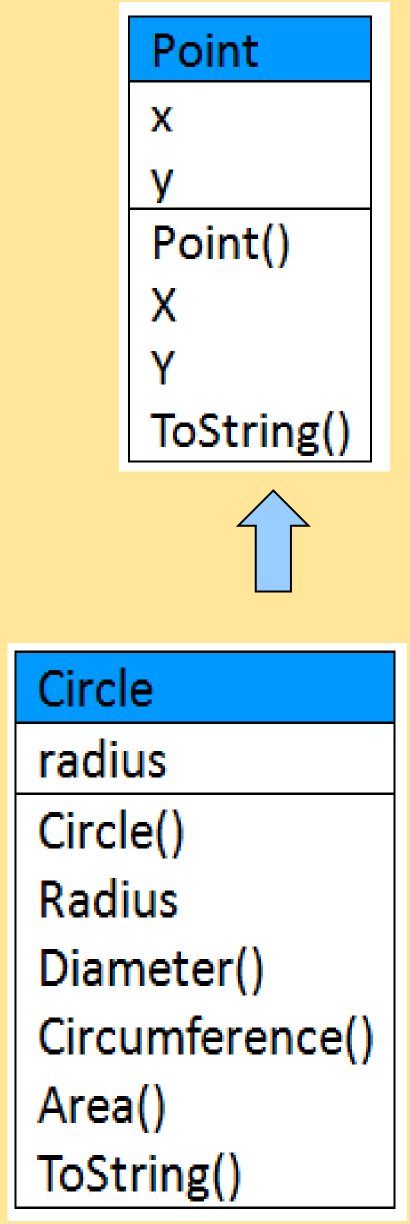


```
    public double Diameter()  {
       return Radius * 2;
    }
    public double Circumference() {
       return Math.PI * Diameter();
    }
    public virtual double Area() {
       return Math.PI * Math.Pow( Radius, 2 );
    }
```

Override its base method

```
    public override string ToString() {
       return "Center= " + base.ToString() +
              "; Radius = " + Radius;
    }
 }
```

Call its base method



Point

x

y

Point()

X

Y

ToString()

Circle

radius

Circle()

Radius

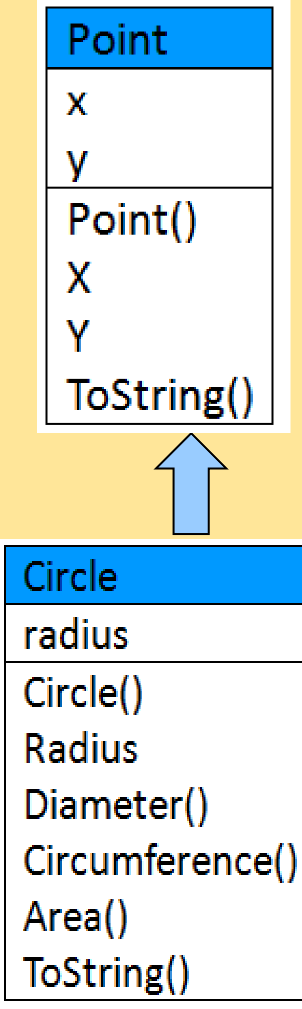
Diameter()

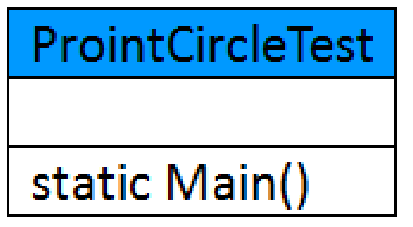
Circumference()

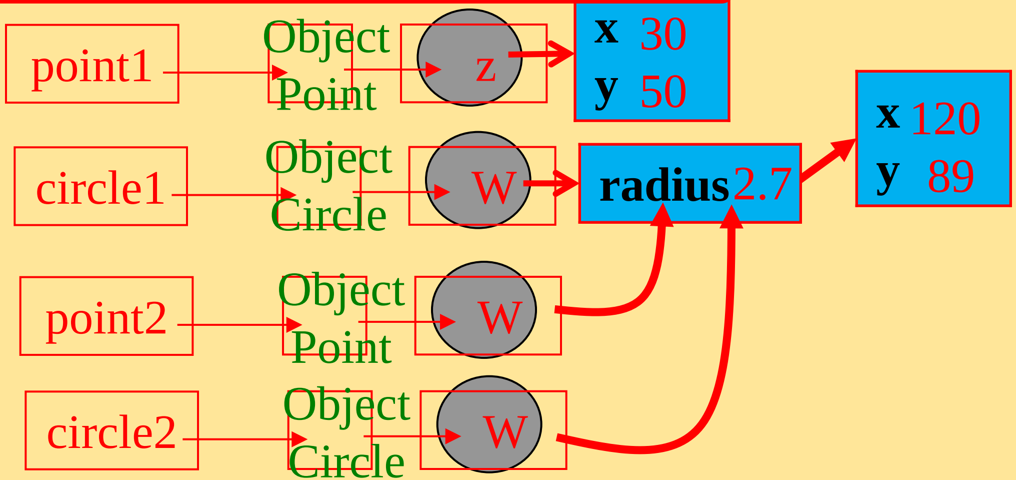
Area()

ToString()

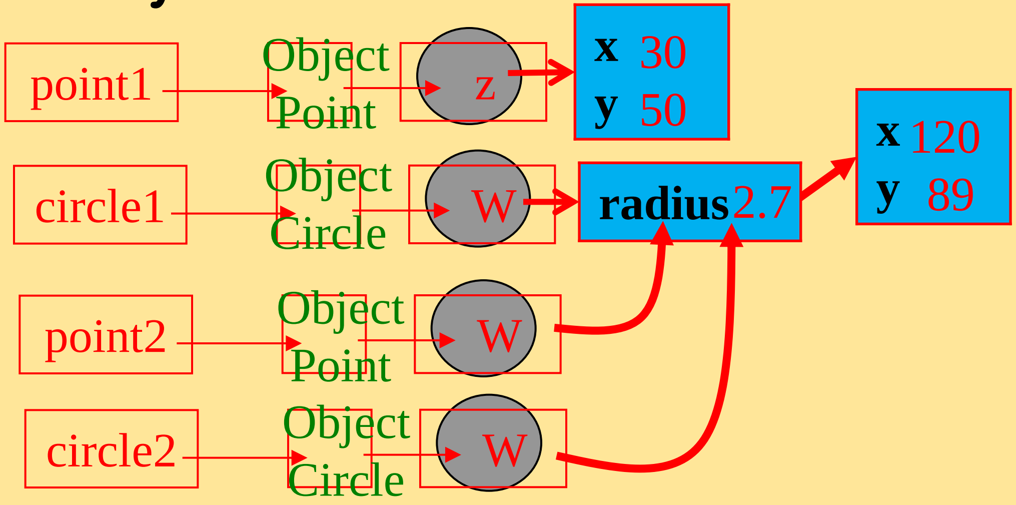
```
using System;
using System.Windows.Forms;
class PointCircleTest  {
   static void Main( string[] args )  {
      Point point1 = new Point( 30, 50 );
      Circle circle1 = new Circle( 120, 89, 2.7 );
      string output = "Point point1: " + point1.ToString() +
         "\nCircle circle1: " + circle1.ToString();
      Point point2 = circle1;
      output += "\n\nCCircle circle1 (via point2): " +
               point2.ToString();
      Circle circle2 = ( Circle ) point2;
      output += "\n\nCircle circle1 (via circle2): " +
               circle2.ToString();
      output += "\nArea of circle1 (via circle2): " +
               circle2.Area().ToString( "F" );
```

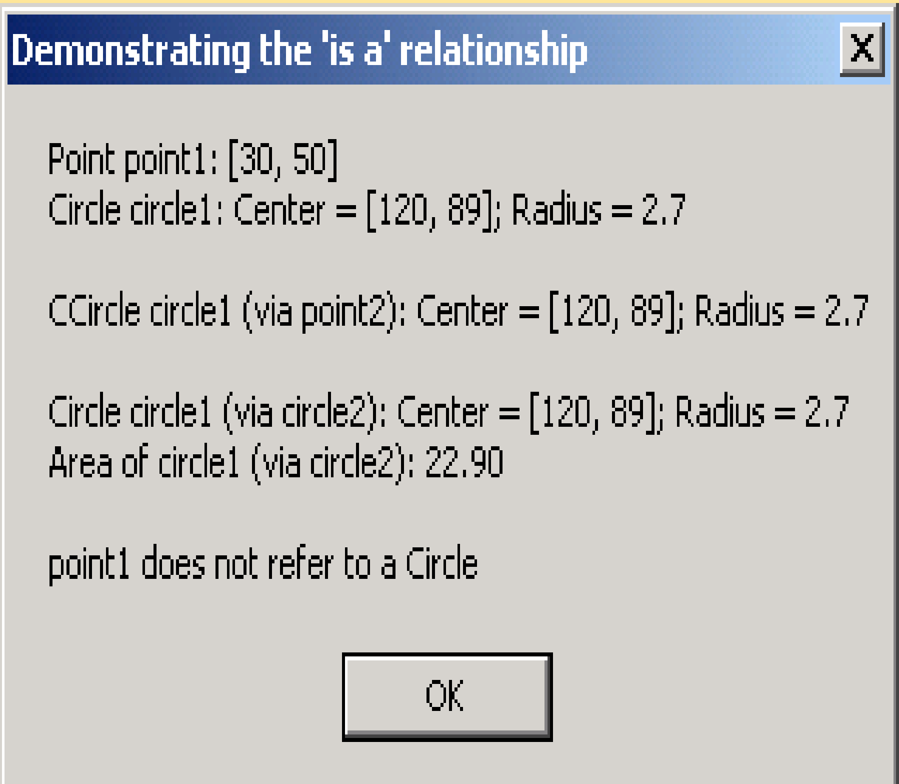






```
        if ( point1 is Circle ) {
            circle2 = ( Circle ) point1;
            output += "\n\ncast successful";
        }
        else    {
            output += "\n\npoint1 does not refer to a Circle";
        }
        MessageBox.Show( output,
            "Demonstrating the 'is a' relationship" );
    }
}
```




Demonstrating the 'is a' relationship

```
Point point1: [30, 50]
Circle circle1: Center = [120, 89]; Radius = 2.7

CCircle circle1 (via point2): Center = [120, 89]; Radius = 2.7

Circle circle1 (via circle2): Center = [120, 89]; Radius = 2.7
Area of circle1 (via circle2): 22.90

point1 does not refer to a Circle
```

Abstract Superclasses and Concrete Classes

- Abstract classes
  - *A class is too generic to define real objects*
  - Thus, its objects cannot be instantiated
  - But the class acts as superclass from which other classes can inherit
  - *abstract class used when not sure what the future for program, but need to set a canonical class beforehand*
- Concrete classes
  - A Class from which objects can be instantiated

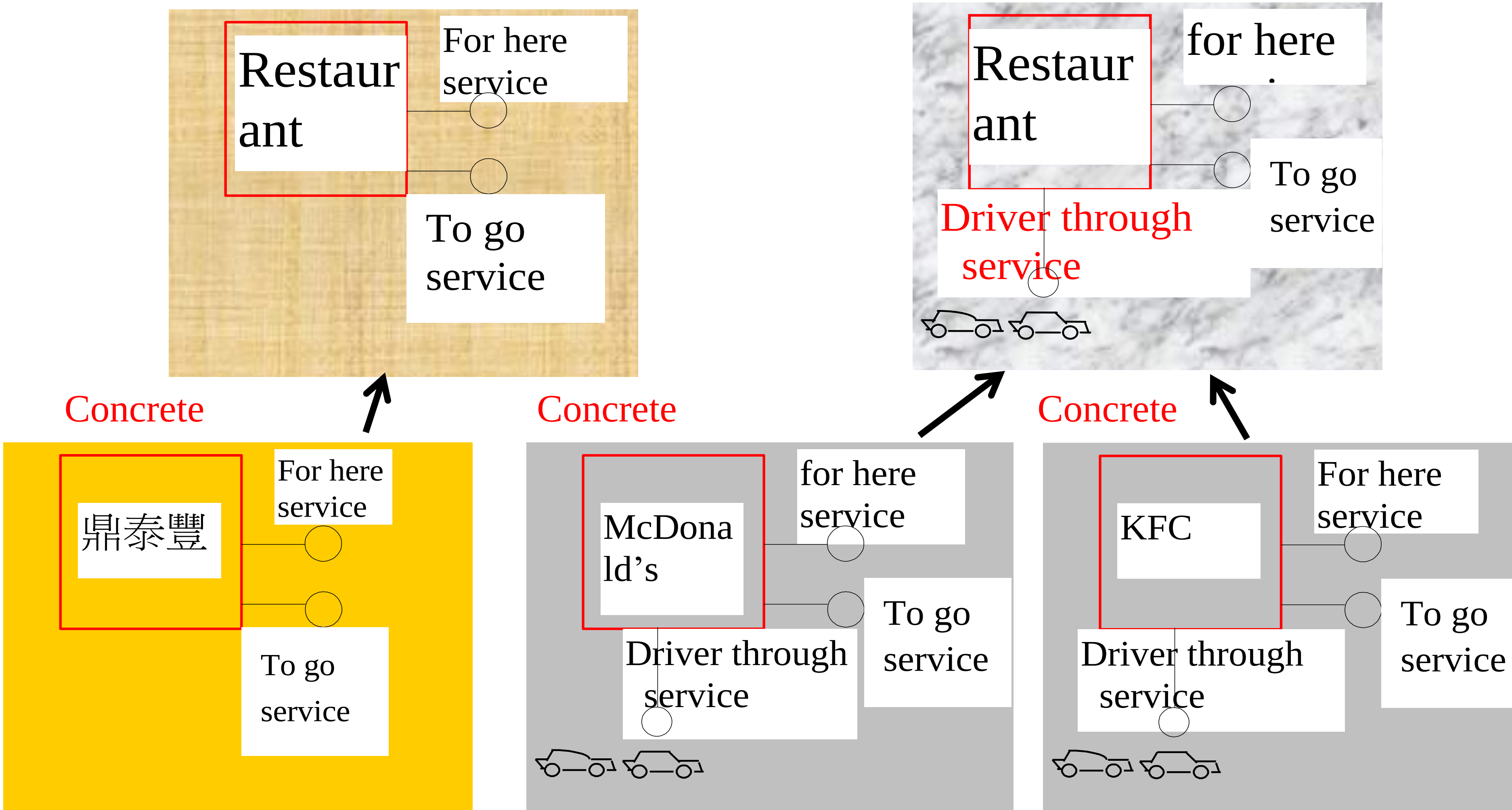
Abstract Class vs. Concrete class

Abstract

Chinese Class

Abstract

Western Class



Restaurant

For here service

To go service

Restaurant

for here

To go service

Driver through service

Concrete

鼎泰豐

For here service

To go service

Concrete

McDonald’s

for here service

To go service

Driver through service

Concrete

KFC

For here service

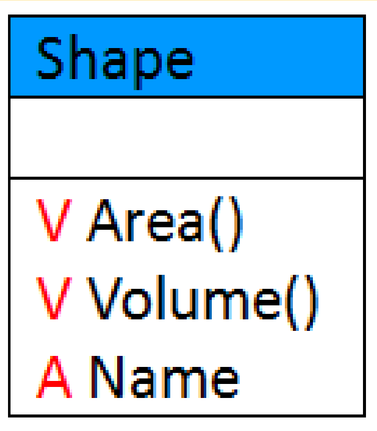
To go service

Driver through service

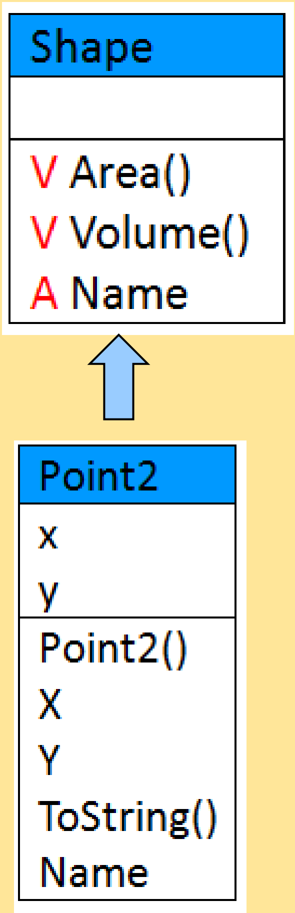
Abstract Classes and Methods

- To declare a method or property abstract, use keyword **abstract** in the declaration;
  - abstract methods and properties have no implementation
  - Derived classes must override abstract methods and properties of the base class to enable instantiation
- Concrete classes use the keyword **override**
  - to provide implementations for all the abstract methods and properties of the base-class
- Cf: virtual method:
  - A method must be declared virtual if that method that the derived class should override it
- Cf: sealed method:
  - the method cannot be override in the derived class

```
using System;
public abstract class Shape {
    public virtual double Area()  {
        return 0;
    }
    public virtual double Volume() {
        return 0;
    }
    public abstract string Name  {
        get;
    }
}
```



```
using System;
public class Point2 : Shape   {
    private int x, y;
    public Point2()   {
    }
    public Point2( int xValue, int yValue )   {
        x = xValue;
        y = yValue;
    }
    public int X   {
        get  {
            return x;
        }
        set   {
            x = value;
        }
    }
    public int Y   {
        get  {
            return y;
        }
        set  {
            y = value;
        }
    }
```



Shape

V Area()

V Volume()

A Name

Point2

x

y

Point2()

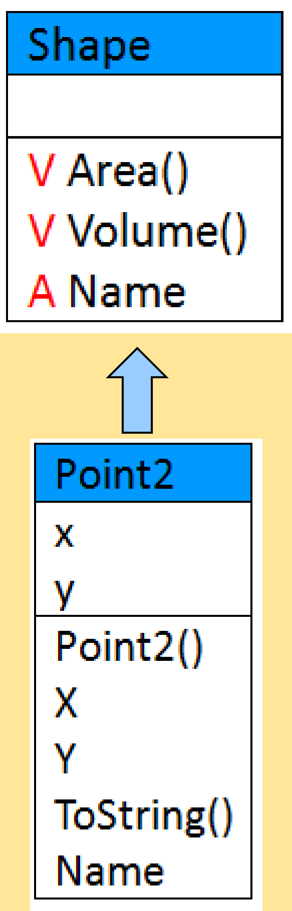
X

Y

ToString()

Name

```
    public override string ToString()   {
        return "[" + X + ", " + Y + "]";
    }
    public override string Name  {
        get     {
            return "Point2";
        }
    }
}
```



Shape

V Area()

V Volume()

A Name

Point2

x

y

Point2()

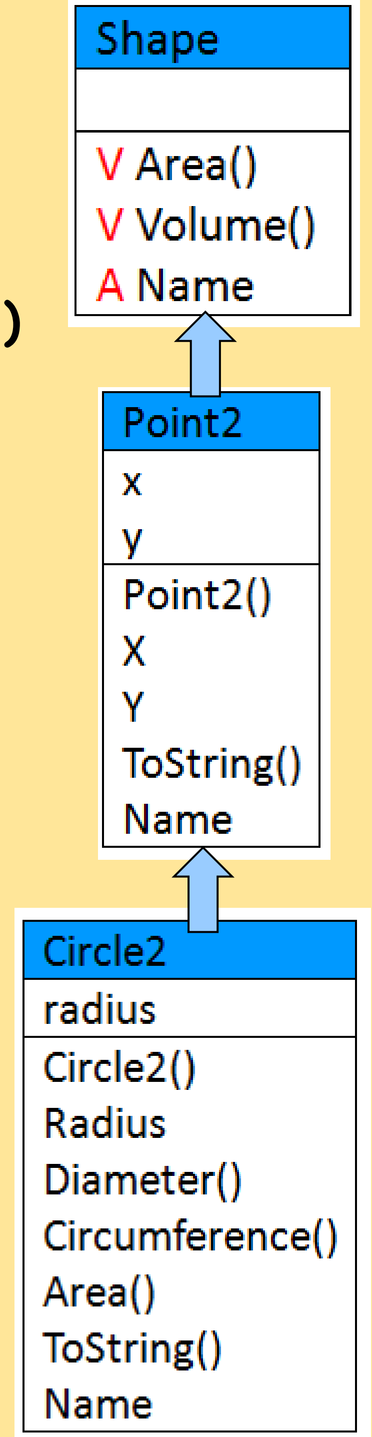
X

Y

ToString()

Name

```
using System;
public class Circle2 : Point2  {
   private double radius;
    public Circle2()  {
    }
    public Circle2( int xValue, int yValue, double radiusValue )
       : base( xValue, yValue )  {
       Radius = radiusValue;
    }
    public double Radius   {
       get   {
          return radius;
       }
       set   {
          if ( value >= 0 )
             radius = value;
       }
    }
    public double Diameter()   {
       return Radius * 2;
    }
```



Shape

V Area()
V Volume()
A Name

Point2

x
y

Point2()
X
Y
ToString()
Name

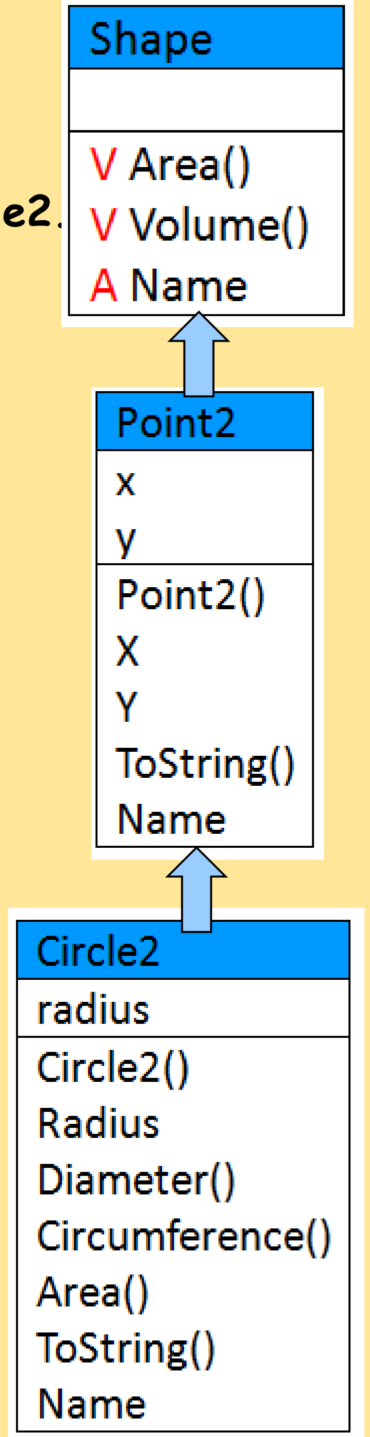
Circle2

radius

Circle2()
Radius
Diameter()
Circumference()
Area()
ToString()
Name

```
    public double Circumference()  {
        return Math.PI * Diameter();
    }
    public override double Area()  {
        return Math.PI * Math.Pow( Radius, 2 );
    }
    public override string ToString()  {
        return "Center = " + base.ToString() +
            "; Radius = " + Radius;
    }
    public override string Name  {
        get {
            return "Circle2";
        }
    }
}
```

Circle2.



Shape

V Area()

V Volume()

A Name

Point2

x

y

Point2()

X

Y

ToString()

Name

Circle2

radius

Circle2()

Radius

Diameter()

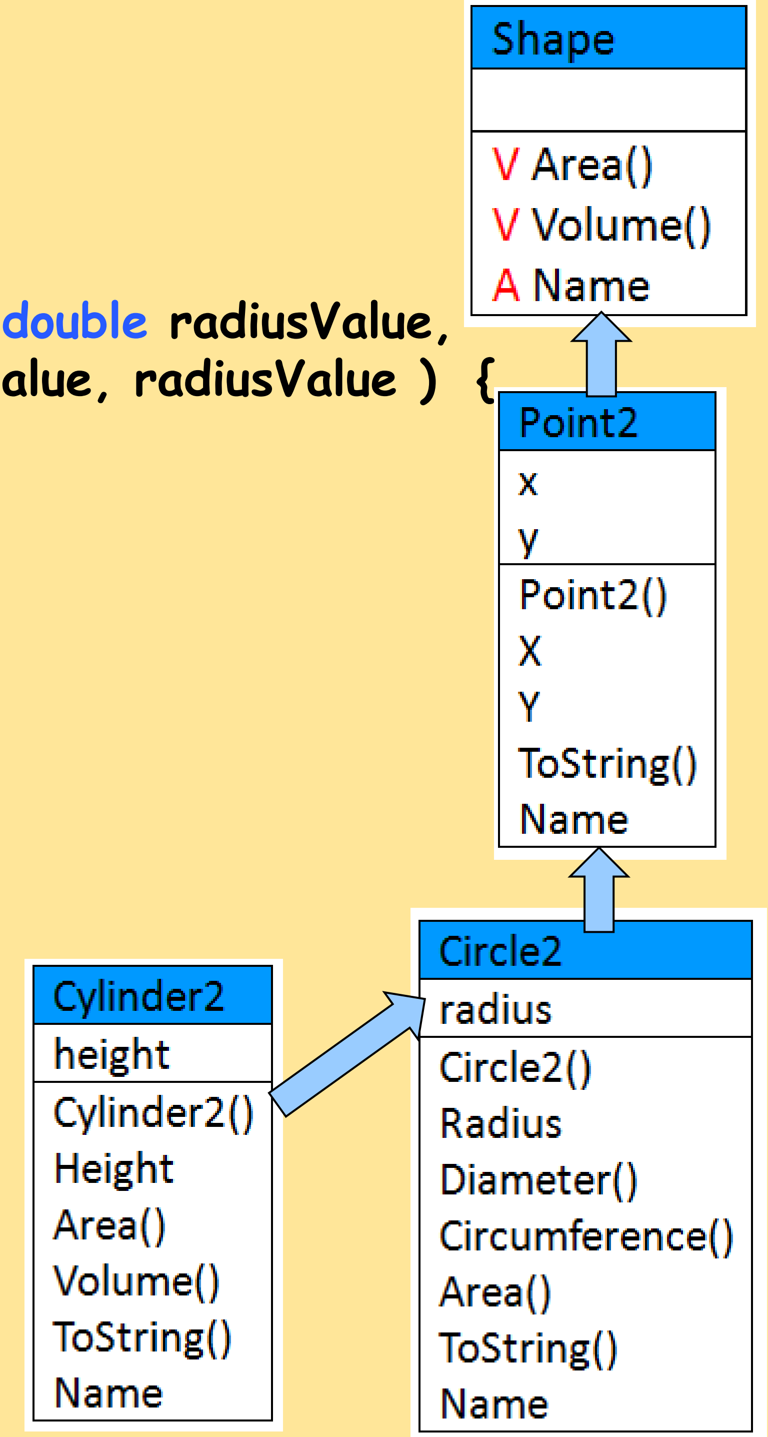
Circumference()

Area()

ToString()

Name

```
using System;
public class Cylinder2 : Circle2  {
   private double height;
    public Cylinder2()   {
    }
    public Cylinder2( int xValue, int yValue, double radiusValue,
       double heightValue): base( xValue, yValue, radiusValue )  {
       Height = heightValue;
    }
    public double Height   {
       get      {
          return height;
       }
       set  {
          if ( value >= 0 )
             height = value;
       }
    }
    public override double Area()   {
       return 2 * base.Area() +
         base.Circumference() * Height;
    }
```



Shape

V Area()
V Volume()
A Name

Point2

x
y

Point2()
X
Y
ToString()
Name

Circle2

radius

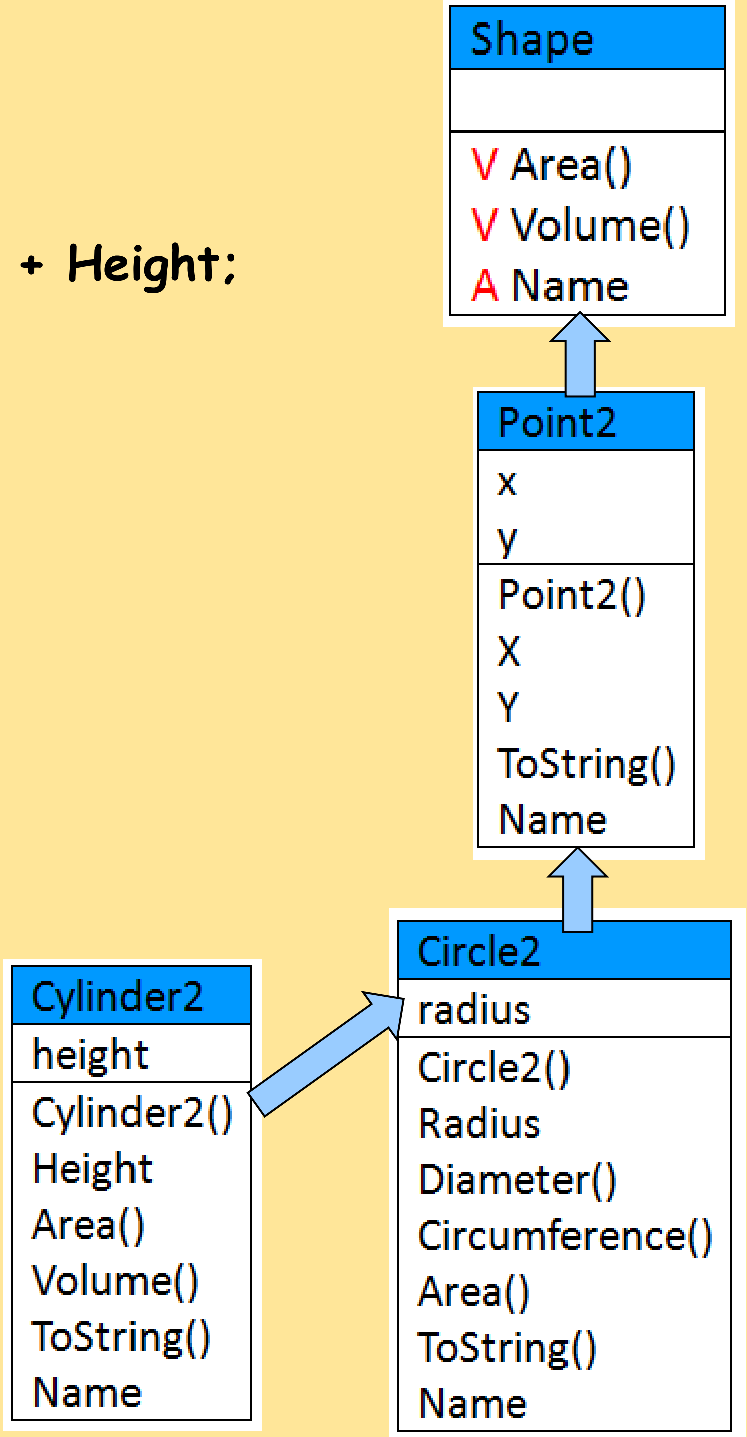
Circle2()
Radius
Diameter()
Circumference()
Area()
ToString()
Name

Cylinder2

height

Cylinder2()
Height
Area()
Volume()
ToString()
Name

```
    public override double Volume()   {
        return base.Area() * Height;
    }
    public override string ToString()   {
        return base.ToString() + "; Height = " + Height;
    }
    public override string Name  {
        get      {
            return "Cylinder2";
        }
    }
}
```



Shape

V Area()
V Volume()
A Name

Point2

x
y

Point2()
X
Y
ToString()
Name

Circle2

radius

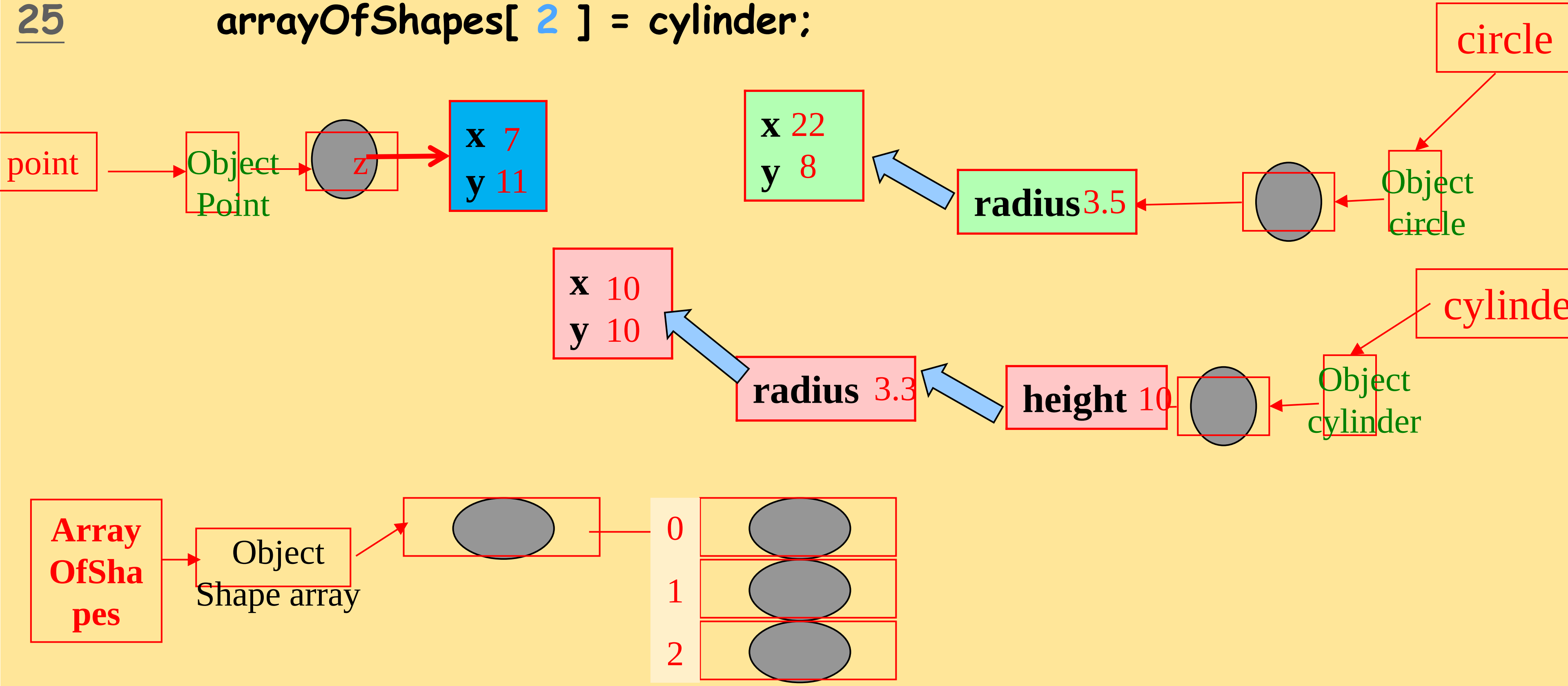
Circle2()
Radius
Diameter()
Circumference()
Area()
ToString()
Name

Cylinder2

height

Cylinder2()
Height
Area()
Volume()
ToString()
Name

```
using System;
using System.Windows.Forms;
public class AbstractShapesTest  {
   public static void Main( string[] args ) {
      Point2 point = new Point2( 7, 11 );
      Circle2 circle = new Circle2( 22, 8, 3.5 );
      Cylinder2 cylinder = new Cylinder2( 10, 10, 3.3, 10 );
      Shape[] arrayOfShapes = new Shape[ 3 ];
      arrayOfShapes[ 0 ] = point;
      arrayOfShapes[ 1 ] = circle;
      arrayOfShapes[ 2 ] = cylinder;
```



circle

point

Object
Point

z

x 7
y 11

x 22
y 8

radius 3.5

Object
circle

x 10
y 10

cylinde

radius 3.3

height 10

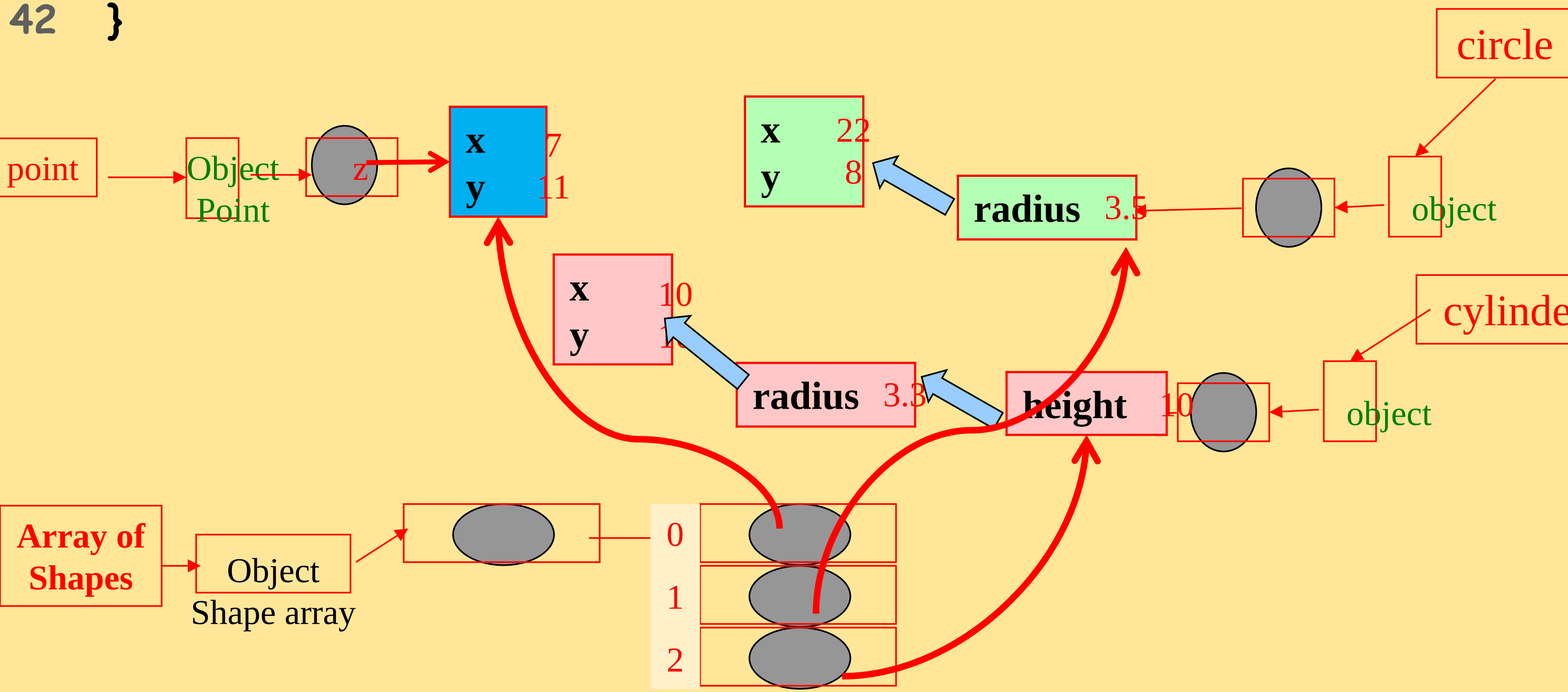
Object
cylinder

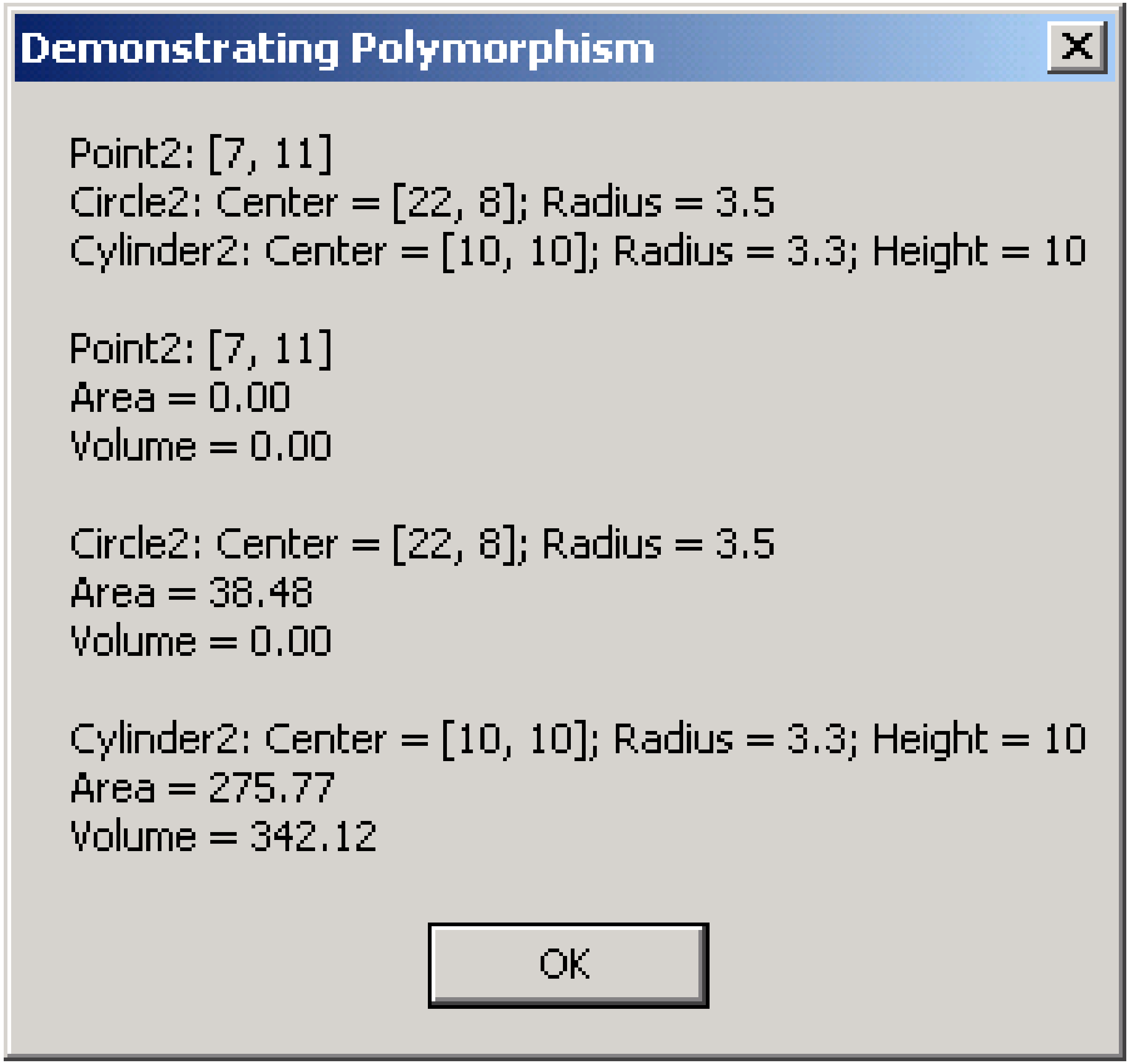
Array
OfSha
pes

Object
Shape array

0
1
2

```
        string output = point.Name + ": " + point + "\n" +
           circle.Name + ": " + circle + "\n" +
           cylinder.Name + ": " + cylinder;
        foreach( Shape shape in arrayOfShapes ) {
           output += "\n\n" + shape.Name + ": " + shape +
              "\nArea = " + shape.Area().ToString( "F" ) +
              "\nVolume = " + shape.Volume().ToString( "F" );
        }
        MessageBox.Show( output, "Demonstrating Polymorphism" );
    }
  }
```





```
Point2: [7, 11]
Circle2: Center = [22, 8]; Radius = 3.5
Cylinder2: Center = [10, 10]; Radius = 3.3; Height = 10

Point2: [7, 11]
Area = 0.00
Volume = 0.00

Circle2: Center = [22, 8]; Radius = 3.5
Area = 38.48
Volume = 0.00

Cylinder2: Center = [10, 10]; Radius = 3.3; Height = 10
Area = 275.77
Volume = 342.12
```