C# 类 (Class)　　C# 多态性

C# 继承

继承是面向对象程序设计中最重要的概念之一。继承允许我们根据一个类来定义另一个类，这使得创建和维护应用程序变得更容易。同时也有利于重用代码和节省开发时间。

当创建一个类时，程序员不需要完全重新编写新的数据成员和成员函数，只需要设计一个新的类，继承了已有的类的成员即可。这个已有的类被称为的**基类**，这个新的类被称为**派生类**。

继承的思想实现了 **属于（IS-A）** 关系。例如，哺乳动物 **属于（IS-A）** 动物，狗 **属于（IS-A）** 哺乳动物，因此狗 **属于（IS-A）** 动物。

基类和派生类

一个类可以派生自多个类或接口，这意味着它可以从多个基类或接口继承数据和函数。

C# 中创建派生类的语法如下：

```
<访问修饰符符> class <基类>
{
 ...
}
class <派生类> : <基类>
{
 ...
}
```

假设，有一个基类 Shape，它的派生类是 Rectangle：

实例

```
using System;
namespace InheritanceApplication
{
   class Shape
   {
      public void setWidth(int w)
      {
         width = w;
      }
      public void setHeight(int h)
      {
         height = h;
      }
      protected int width;
      protected int height;
   }

   // 派生类
   class Rectangle: Shape
   {
      public int getArea()
```

```
      {
         return (width * height);
      }
   }

   class RectangleTester
   {
      static void Main(string[] args)
      {
         Rectangle Rect = new Rectangle();

         Rect.setWidth(5);
         Rect.setHeight(7);

         // 打印对象的面积
         Console.WriteLine("总面积:  {0}",  Rect.getArea());
         Console.ReadKey();
      }
   }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
总面积:  35
```

基类的初始化

派生类继承了基类的成员变量和成员方法。因此父类对象应在子类对象创建之前被创建。您可以在成员初始化列表中进行父类的初始化。

下面的程序演示了这点：

实例

```
using System;
namespace RectangleApplication
{
   class Rectangle
   {
      // 成员变量
      protected double length;
      protected double width;
      public Rectangle(double l, double w)
      {
         length = l;
         width = w;
      }
      public double GetArea()
      {
         return length * width;
      }
      public void Display()
      {
```

```
            Console.WriteLine("长度:  {0}", length);
            Console.WriteLine("宽度:  {0}", width);
            Console.WriteLine("面积:  {0}", GetArea());
        }
    }//end class Rectangle
    class Tabletop : Rectangle
    {
        private double cost;
        public Tabletop(double l, double w) : base(l, w)
        { }
        public double GetCost()
        {
            double cost;
            cost = GetArea() * 70;
            return cost;
        }
        public void Display()
        {
            base.Display();
            Console.WriteLine("成本:  {0}", GetCost());
        }
    }
    class ExecuteRectangle
    {
        static void Main(string[] args)
        {
            Tabletop t = new Tabletop(4.5, 7.5);
            t.Display();
            Console.ReadLine();
        }
    }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
长度:  4.5
宽度:  7.5
面积:  33.75
成本:  2362.5
```

C# 多重继承

多重继承指的是一个类别可以同时从多于一个父类继承行为与特征的功能。与单一继承相对，单一继承指一个类别只可以继承自一个父类。

C# 不支持多重继承。但是，您可以使用接口来实现多重继承。下面的程序演示了这点：

实例

```
using System;
namespace InheritanceApplication
{
```

```
   class Shape
   {
      public void setWidth(int w)
      {
         width = w;
      }
      public void setHeight(int h)
      {
         height = h;
      }
      protected int width;
      protected int height;
   }

   // 基类 PaintCost
   public interface PaintCost
   {
      int getCost(int area);

   }
   // 派生类
   class Rectangle : Shape, PaintCost
   {
      public int getArea()
      {
         return (width * height);
      }
      public int getCost(int area)
      {
         return area * 70;
      }
   }
   class RectangleTester
   {
      static void Main(string[] args)
      {
         Rectangle Rect = new Rectangle();
         int area;
         Rect.setWidth(5);
         Rect.setHeight(7);
         area = Rect.getArea();
         // 打印对象的面积
         Console.WriteLine("总面积:  {0}",  Rect.getArea());
         Console.WriteLine("油漆总成本:  ${0}" , Rect.getCost(area));
         Console.ReadKey();
      }
   }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
总面积:  35
油漆总成本:  $2450
```

C# 类 (Class)

C# 多态性

3 篇笔记

写笔记