8.7.1　构造函数初始化语句

默认情况下，在构造对象时，将调用基类的无参数构造函数。但构造函数可以重载，所以基类可能有一个以上的构造函数。如果希望派生类使用一个指定的基类构造函数而不是无参数构造函数，必须在构造函数初始化语句中指定它。

有两种形式的构造函数初始化语句。

- 第一种形式使用关键字 base 并指明使用哪一个基类构造函数。
- 第二种形式使用关键字 this 并指明应该使用当前类的哪一个构造函数。

基类构造函数初始化语句放在冒号后面，跟在类的构造函数声明的参数列表后面。构造函数初始化语句由关键字 base 和要调用的基类构造函数的参数列表组成。

例如，下面的代码展示了类 MyDerivedClass 的构造函数。

- 构造函数初始化语句指明要使用有两个参数的基类构造函数，并且第一个参数是一个 int，第二个参数是一个 string。
- 基类参数列表中的参数必须在类型和顺序方面与已定的基类构造函数的参数列表相匹配。

```
                                    构造函数初始化语句
                                          ↓
public MyDerivedClass( int x, string s ) : base( s, x )
{                                          ↑
   ...                                   关键字
```

当声明一个不带构造函数初始化语句的构造函数时，它实际上是带有 base()构造函数初始化语句的简写形式，如图 8-12 所示。这两种形式是语义等价的。

```
class MyDerived: MyBase
{
   MyDerived()
   {
      ...
   }
   ...

隐式使用基类构造函数
MyBase()的构造函数
```

```
class MyDerived: MyBase
{
   MyDerived() : base()
   {
      ...
   }
   ...

显式使用基类构造函数
MyBase()的构造函数
```

图 8-12　等价的构造函数形式

另外一种形式的构造函数初始化语句可以让构造过程（实际上是编译器）使用当前类中其他的构造函数。例如，如下代码所示的 Myclass 类包含带有一个参数的构造函数。但这个单参数的构造函数使用了同一个类中具有两个参数的构造函数，为第二个参数提供了一个默认值。

```
                                  构造函数初始化语句
                                          ↓
public MyClass(int x): this(x, "Using Default String")
{                        ↑
   ...                 关键字
}
```

这种语法很有用的另一种情况是，一个类有好几个构造函数，并且它们都需要在对象构造的过程开始时执行一些公共的代码。对于这种情况，可以把公共代码提取出来作为一个构造函数，被其他所有的构造函数用作构造函数初始化语句。由于减少了重复的代码，实际上这也是推荐的做法。

你可能会觉得还可以声明另外一个方法来执行这些公共的初始化，并让所有构造函数来调用这个方法。由于种种原因这不是一个好办法。首先，编译器在知道方法是构造函数后能够做一些优化。其次，有些事情必须在构造函数中进行，在其他地方则不行。比如之前我们学到的 readonly 字段只可以在构造函数中初始化。如果尝试在其他方法（即使这个方法只被构造函数调用）中初始化一个 readonly 字段，会得到编译错误。不过要注意，这一限制仅适用于 readonly 字段，不适用于 readonly 属性。

回到公共构造函数，如果这个构造函数可以用作一个有效的构造函数，能够初始化类中所有需要初始化的东西，那么完全可以把它设置为 public 的构造函数。

但是如果它不能完全初始化一个对象怎么办？此时，必须禁止从类的外部调用构造函数，因为那样的话它只会初始化对象的一部分。要避免这个问题，可以把构造函数声明为 private，而不是 public，然后只让其他构造函数使用它，如以下代码所示：

```
class MyClass
{
   readonly int    firstVar;
   readonly double secondVar;

   public string UserName;
   public int UserIdNumber;

   private MyClass( )              //私有构造函数执行其他构造
   {                               //函数共用的初始化
      firstVar  = 20;
      secondVar = 30.5;
   }

   public MyClass( string firstName ) : this() //使用构造函数初始化语句
   {
      UserName     = firstName;
      UserIdNumber = -1;
   }

   public MyClass( int idNumber ) : this( )    //使用构造函数初始化语句
   {
      UserName     = "Anonymous";
      UserIdNumber = idNumber;
   }
}
```

8.7.2 类访问修饰符

类可以被系统中其他类看到并访问。这一节阐述类的可访问性。虽然我们会在解说和示例中