然而有的时候，如果能使用索引访问它们将会很方便，好像该实例是字段的数组一样。这正是索引器能做的事。如果为类 Employee 写一个索引器，方法 Main 看起来就像图 7-14 中的代码那样。请注意没有使用点运算符，相反，索引器使用索引运算符，它由一对方括号和中间的索引组成。

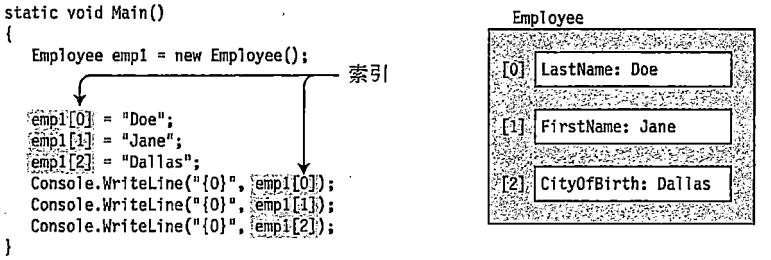

图 7-14　使用索引字段

7.17.1　什么是索引器

索引器是一组 get 和 set 访问器，与属性类似。图 7-15 展示了一个类的索引器的表现形式，该类可以获取和设置 string 型值。

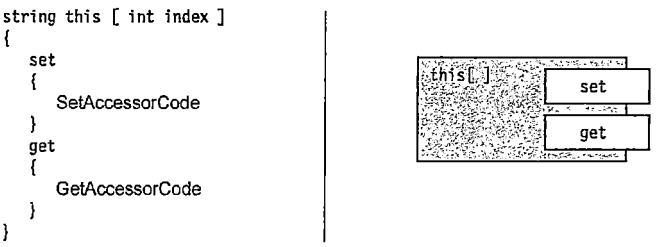

图 7-15　索引器的表现形式

7.17.2　索引器和属性

索引器和属性在很多方面是相似的。

- 和属性一样，索引器不用分配内存来存储。
- 索引器和属性都主要被用来访问其他数据成员，它们与这些成员关联，并为它们提供获取和设置访问。
  - 属性通常表示单个数据成员。
  - 索引器通常表示多个数据成员。

说明　可以认为索引器是为类的多个数据成员提供 get 和 set 访问的属性。通过提供索引器，可以在许多可能的数据成员中进行选择。索引本身可以是任何类型，而不仅仅是数值类型。

关于索引器，还有一些注意事项如下。

- 和属性一样，索引器可以只有一个访问器，也可以两个都有。
- 索引器总是实例成员，因此不能被声明为 static。
- 和属性一样，实现 get 和 set 访问器的代码不一定要关联到某个字段或属性。这段代码可以做任何事情也可以什么都不做，只要 get 访问器返回某个指定类型的值即可。

7.17.3 声明索引器

声明索引器的语法如下所示。请注意以下几点。

- 索引器没有名称。在名称的位置是关键字 this。
- 参数列表在方括号中间。
- 参数列表中必须至少声明一个参数。

```
          关键字        参数列表
            ↓          ↓
ReturnType this [ Type param1, ... ]
{              ↑              ↑
   get      方括号          方括号
   {
        ...
   }
   set
   {
        ...
   }
}
```

声明索引器类似于声明属性。图 7-16 阐明了它们在语法上的相似点和不同点。

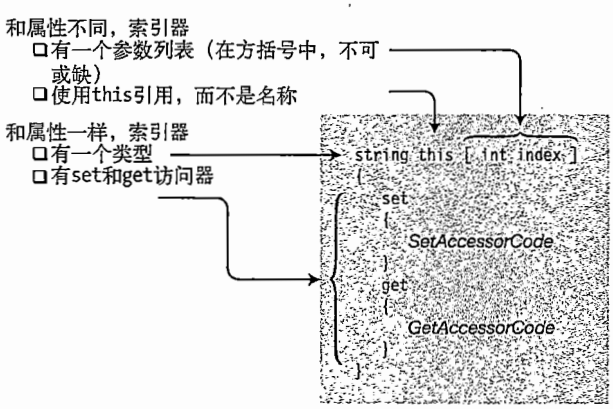

图 7-16 比较索引器声明和属性声明

7.17.4　索引器的 set 访问器

当索引器被用于赋值时，set 访问器被调用，并接受两项数据，如下：

- 一个名为 value 的隐式参数，其中持有要保存的数据；
- 一个或更多索引参数，表示数据应该保存到哪里。

```
emp[0] = "Doe";
   ↑      ↑
  索引    值
  参数
```

在 set 访问器中的代码必须检查索引参数，以确定数据应该存往何处，然后保存它。

set 访问器的语法和含义如图 7-17 所示。图的左边展示了访问器声明的实际语法。右边展示了访问器的语义，如果它是以普通方法的语法书写的。右边的图例表明 set 访问器有如下语义。

- 它的返回类型为 void。
- 它使用的参数列表和索引器声明中的相同。
- 它有一个名为 value 的隐式参数，值参类型和索引器类型相同。

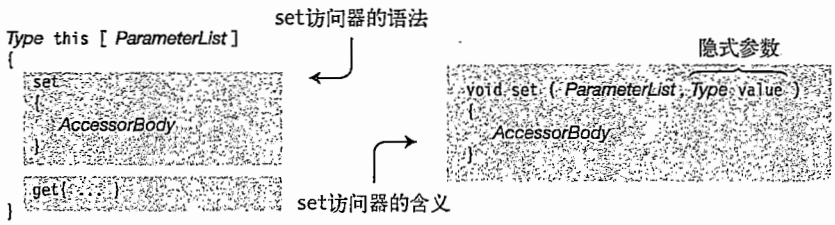

图 7-17　set 访问器声明的语法和含义

7.17.5　索引器的 get 访问器

当使用索引器获取值时，可以通过一个或多个索引参数调用 get 访问器。索引参数指示获取哪个值。

```
string s = emp[0];
             ↑
          索引参数
```

get 访问器方法体内的代码必须检查索引参数，确定它表示的是哪个字段，并返回该字段的值。

get 访问器的语法和含义如图 7-18 所示。图的左边展示了访问器声明的实际语法。右边展示了访问器的语义，如果它是以普通方法的语法书写的。get 访问器有如下语义。

- 它的参数列表和索引器声明中的相同。
- 它返回与索引器类型相同的值。

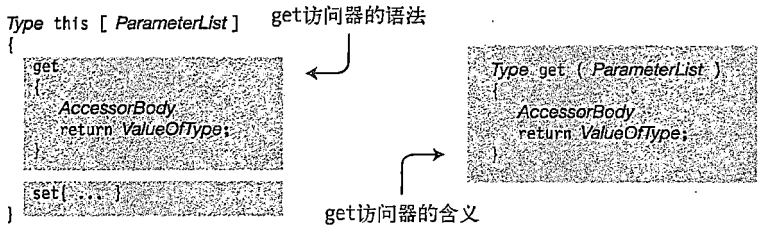

图 7-18 get 访问器声明的语法和含义

7.17.6 关于索引器的更多内容

和属性一样，不能显式调用 get 和 set 访问器。取而代之，当索引器用在表达式中取值时，将自动调用 get 访问器。当使用赋值语句对索引器赋值时，将自动调用 set 访问器。

在“调用”索引器时，要在方括号中提供参数。

```
   索引   值
    ↓     ↓
emp[0] = "Doe";                          //调用 set 访问器
string NewName = emp[0];                 //调用 get 访问器
                  ↑
                索引
```

7.17.7 为 Employee 示例声明索引器

下面的代码为先前示例中的类 Employee 声明了一个索引器。

- 索引器需要读写 string 类型的值，所以 string 必须声明为索引器的类型。它必须声明为 public，以便从类的外部访问。
- 3 个字段被随意地索引为整数 0~2，所以本例中方括号中间名为 index 的形参必须为 int 型。
- 在 set 访问器方法体内，代码确定索引指的是哪个字段，并把隐式变量 value 的值赋给它。在 get 访问器方法体内，代码确定索引指的是哪个字段，并返回该字段的值。

```
class Employee {
   public string LastName;                //调用字段 0
   public string FirstName;               //调用字段 1
   public string CityOfBirth;             //调用字段 2

   public string this[int index]          //索引器声明
   {
      set                                 //set 访问器声明
      {
         switch (index) {
            case 0: LastName = value;
               break;
            case 1: FirstName = value;
               break;
```

```
            case 2: CityOfBirth = value;
               break;

            default:                                    //(第23章中的异常)
               throw new ArgumentOutOfRangeException("index");
         }
      }

      get                                               //get访问器声明
      {
         switch (index) {
            case 0: return LastName;
            case 1: return FirstName;
            case 2: return CityOfBirth;

            default:                                    //(第23章中的异常)
               throw new ArgumentOutOfRangeException("index");
         }
      }
   }
}
```

7.17.8　另一个索引器示例

下面的示例为类 Class1 的两个 int 字段设置索引。

```
class Class1
{
   int Temp0;                          //私有字段
   int Temp1;                          //私有字段
   public int this [ int index ]       //索引
   {
      get
      {
         return ( 0 == index )         //返回Temp0或Temp1的值
                  ? Temp0
                  : Temp1;
      }

      set
      {
         if( 0 == index )
            Temp0 = value;             //注意隐式变量"value"
         else
            Temp1 = value;             //注意隐式变量"value"
      }
   }
}

class Example
{
   static void Main()
   {
```

```
        Class1 a = new Class1();

        Console.WriteLine("Values -- T0: {0},  T1: {1}", a[0], a[1]);
        a[0] = 15;
        a[1] = 20;
        Console.WriteLine($"Values -- T0: { a[0] }, T1: { a[1] }");
    }
}
```

这段代码产生以下输出：

```
Values -- T0: 0,  T1: 0
Values -- T0: 15, T1: 20
```

7.17.9 索引器重载

只要索引器的参数列表不同，类就可以有任意多个索引器。索引器类型不同是不够的。这叫作索引器重载，因为所有的索引器都有相同的“名称”：this 访问引用。

例如，下面的代码有 3 个索引器：两个 string 类型的和一个 int 类型的。两个 string 类型的索引中，一个带一个 int 参数，另一个带两个 int 参数。

```
class MyClass
{
    public string this [ int index ]
    {
        get { ... }
        set { ... }
    }

    public string this [ int index1, int index2 ]
    {
        get { ... }
        set { ... }
    }

    public int this [ float index1 ]
    {
        get { ... }
        set { ... }
    }

    ...
}
```

说明 请记住，类中重载的索引器必须有不同的参数列表。