```
public Panda (string n) => name = n;
```

3.1.3.1 重载构造器

类或者结构体可以重载构造器。为了避免重复代码，构造器可以用 this 关键字调用另一个构造器：

```
using System;

public class Wine
{
  public decimal Price;
  public int Year;
  public Wine (decimal price) { Price = price; }
  public Wine (decimal price, int year) : this (price) { Year = year; }
}
```

当构造器调用另一个构造器时，被调用的构造器先执行。

还可以向另一个构造器传递表达式：

```
public Wine (decimal price, DateTime year) : this (price, year.Year) { }
```

表达式内不能使用 this 引用，例如，不能调用实例方法（这是强制性的。由于这个对象当前还没有通过构造器初始化完毕，因此调用任何方法都有可能失败）。但是表达式可以调用静态方法。

3.1.3.2 隐式无参数构造器

C# 编译器会自动为没有显式定义构造器的类生成无参数公有构造器。但是，一旦显式定义了至少一个构造器，系统就不再自动生成无参数的构造器。

3.1.3.3 构造器和字段的初始化顺序

之前提到，字段可以在声明时初始化为其默认值：

```
class Player
{
  int shields = 50;    // Initialized first
  int health = 100;    // Initialized second
}
```

字段的初始化按声明的先后顺序，在构造器之前执行。

3.1.3.4 非公有构造器

构造器不一定都是公有的。通常，定义非公有的构造器的原因是为了通过一个静态方法调用来控制类实例的创建。静态方法可以从一个池中返回对象，而不必每次创建一个新