## 1.2什么是.NET Framework

C#编写的程序需要编译和运行的，.NET Framework就是应用程序运行时的执行环境，为应用程序提供以下几种服务。

* 全面的类库 .NET Framework提供了丰富的类库以供调用，这使我们不必再去编写大量的代码去处理常见的操作。
* 内存管理 在许多其他的编程语言中，开发人员需要负责内存的分配和释放，以及对象生命周期的处理等任务。.NET Framework为了减轻开发人员的工作，提供了内存管理服务。
* 通用类型系统（CTS） CTS定义了可以在中间语言中使用的预定义数据类型。
* 开发结构和技术 .NET Framework提供了开发特定应用程序所需要的库，如Web应用程序的ASP.NET技术。
* 语言互操作性 面向.NET Fremework的语言编译器都提供了生成中间语言代码的机制，这种机制使得不同语言之间进行互操作成为可能。

### 1.2.1 .NET Framework的组成

.NET Framework包括公共语言进行时（CLR）和.NET Framework类库（FCL）。

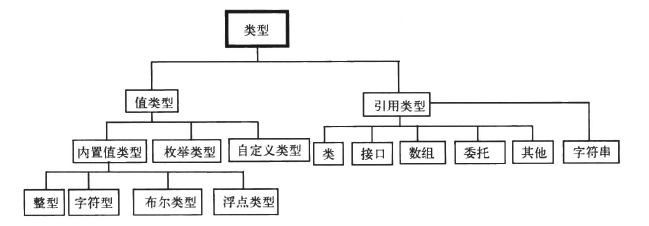
#### 1.CLR

公共语言进行时是.NET Framework的核心基础。我们可以将CLR看成一个在执行时管理代码的代理，它提供了内存管理、线程管理和异常处理等服务，而且还负责对代码实施严格的类型安全检查，保证代码的正确性。我们将手CLR管理的代码成为托管代码，将不受CLR管理的代码成为非托管代码。

* CTS

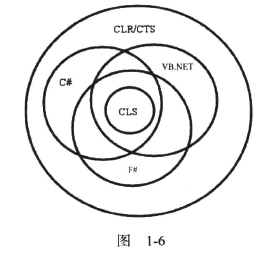
CTS用于解决不用语言之间数据类型不同的问题，如C#中整形是int，而VB.NET中的整形确实Integer，通过CTS可以吧这两个类型变成通用类型Int32。所有.NET语言都共享CTS这一类型系统，从而使他们之间能够实现无缝互操作。

CTS类型主要分为两大类：引用类型和值类型。



* CLS

CLS是一种最底的语言标准，它制定了以.NET平台为目标的语言所必须支持的最小特征，以及某一语言与其他.NET语言之间实现互操作所需要的完备特征。凡是遵守这个标准的语言，在.NET框架下都可以实现互相调用；不符合公共语言规范的代码就不能被其他语言调用，继而不能与其他语言进行互操作。

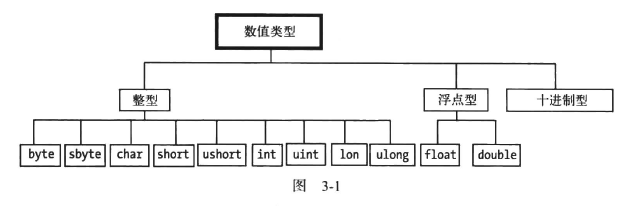


#### 2. .NET Framework

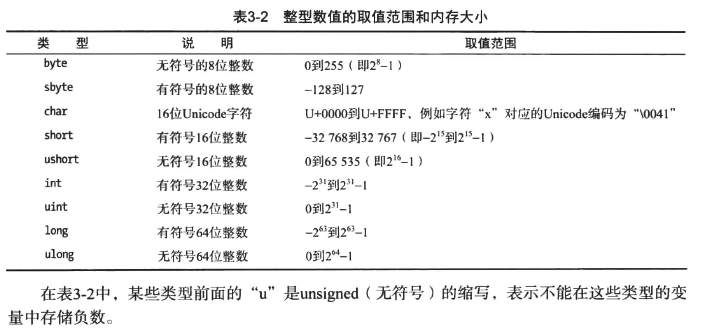
.NET Framework类库就是一组DLL程序集的集合，其中包含了大量定义好的类型，这些类型都公开了一些功能。

由于FCL中包含了数量极多的类型，因此有必要将相关的一组类型放到一个单独的命名空间中加以区别，例如System.IO命名空间中就包含了用于执行I/O操作的类型。因此，在使用FCL中某个类时，还必须知道该类所在的命名空间。

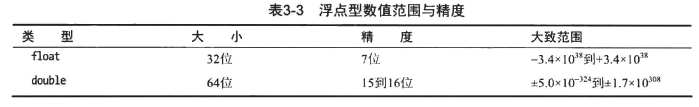
### 3.2.1 数值类型



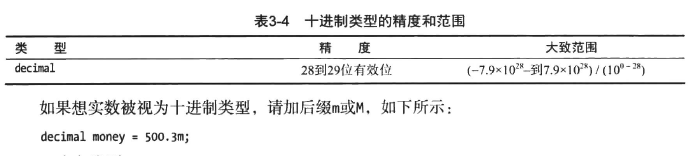
#### 1.整型



#### 2.浮点型



#### 3.十进制



## 4.5 类与结构体的区别

* 语法上的区别在于，定义类要使用关键字class，而定义结构体则使用关键字struct。
* 结构体中不可对声明字段进行初始化，但类可以。
* 如果没有为类显示地定义构造函数，C#编译器会自动生成一个无参数的实例构造函数，我们称之为隐式构造函数；而一旦我们为类显示地定义了一个构造函数，C#编译器就不会再自动生成隐式构造函数了。与此不同的是，在结构体中，无论你是否显式地定义了构造函数，隐式构造函数都是一直存在的。
* 结构体重不能显式地定义无参数的构造函数，这也说明无参构造函数是一直存在的，所以不能在显式地为结构体添加一个无参的构造函数；而类中则可以显式地定义一个无参数的构造函数。
* 在结构体的构造函数中，必须要为结构体中的所有字段赋值。
* 创建结构体对象可以不使用new关键字，但此时结构体对象中的字段是没有初始值的；而类必须使用new关键字来创建对象。
* 结构体不能继承结构或者类，但可以实现接口；而类可以继承类但不能继承结构，它也可以实现接口。
* 类是引用类型，而结构体是值类型。
* 结构体不能定义析构函数，而类可以有析构函数。
* 不能用abstract和sealed关键字修饰结构体，而类可以。

### 5.2.3 子类的初始化顺序

子类的初始化顺序：

1. 初始化类的实例字段；
2. 调用基类的构造函数，如果没有指明基类，则调用System.Object的构造函数；
3. 调用子类的构造函数。

## 5.3 多态

由于可以继承基类的所有成员，子类就都有了相同的行为，但是有时子类的某些行为需要相互区别，子类需要覆写父类的方法来实现子类特有的行为，这样的技术在面向对象的编程中就是多态。

### 5.3.1使用virtual和override关键字实现方法重写

### 5.3.2 阻止派生类重写虚成员

用sealed关键字可以防止一个类被其他类继承。同样，也可以用sealed关键字来阻止派生类重写虚成员。

### 5.3.3 使用新成员隐藏基类成员

如果在派生类中定义了与基类成员同名的成员，可以使用new关键字把基类成员隐藏起来。

## 5.4所有类的父类：System.Object

所有类都派生自System.Object类。