.net基础知识

1. 类型
2. 简介

几乎每种编程语言都有某种形式的类型系统，这种类型系统被分为强/弱、安全/不安全、静态/动态以及其他一些让人更不好懂的说法；

C#的类型系统是静态的、显示的、安全的；

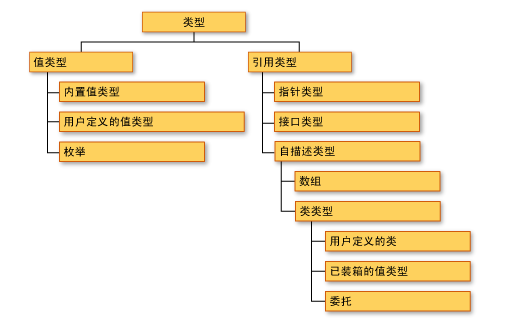
C#是静态类型的：每个变量都有特定的类型，而且该类型在编译时是可知的；

C#是显示类型的：每个变量的类型都必须在声明中显示指明：

C#是安全的：例如：类型转换时，编译器会检查这样转换是否可行，如果不可行，就会触发一个编译时错误，抛出一个异常提示：

1. 分类

现实世界中的值和引用：一份报纸，复制一份给朋友一份，朋友会得到一份属于他自己的报纸（你的报纸添加注释，朋友会不到你的注释），这种行为属于值类型的行为；假如读的是一个网页，需要把URL给到朋友，朋友打开URL地址读取该网页，这种类型属于引用类型（网页添加注释，朋友会看得到添加的注释）；





对于值类型的表达式，它的值就是表达式的值；

对于引用类型表达式，它的值就是一个引用，而不是该引用所指的对象；

差异：

1. 存储位置：局部变量的值总是存储在栈（Stack）中;引用类型（对象）总是存储在堆中（GC Heap），静态变量也是存储在堆中；
2. 可否派生：值类型不可以派生出其他类型，值不需要额外的信息来描述值实际是什么类型；引用类型来说，每个对象的开头都包含一个数据块，它标识了对象的实际类型；
3. 值类型

1、应用场景

2、判等

3、声明规则

1. 引用类型

1、应用场景

1、判等

3、声明规则

String恒定不变特性：两个String，如果赋值为同一个值，在内存中只有一个字符串存在，两个引用的地址相同。由此引出String的不变性。

效率低下的原因：

每次赋值都不一样会产生一个新的引用地址，从而需要到内存中去开辟一个地址，所以String会导致效率低下；

Eg:Sting s1=’’windy” string S2=S1; S1=”peeta” S1=”Peeta” S2 =”windy”;

特殊：动态创建字符串 str1+str2 str1+字符串两种类型；驻留机制对于动态创建的字符串是不起作用的；

1. 类型转化
2. 三大特性

封装

继承

一切皆为对象，一切皆于继承；继承是指一个对象直接使用另一对象的属性和方法；通过继承，可以创建重用、扩展和修改在其他类中定义的行为的新类；

重用特性：子类可以调用父类方法和字段，而父类不能调用子类方法和字段；

扩展特性：继承是可传递的，子类是对父类的扩展，必须继承父类方法，同时可以添加新方法。

修改特性：子类实现虚方法覆写操作，使得父类指针可以指向子类对象成员。

对象的创建过程：

字段的创建过程：分配空间和对象的初始化过程，首先找到父类，并为其字段分配存储空间，一次类推纸质递归结束，也就是完成System.Object内存分配为止；

方发表的创建过程：方法表的创建是类第一次加载到CLR时完成的，在对象创建时只是将其附加成员TypeHandle指向方法列表在Loader Heap上的地址，将对象与其动态方法列表相关联起来，因此方法表是先于对象而存在的。

Eg:Bird bird = new Chicken()

关注对象原则：调用子类或是父类的成员，取决于创建的对象是子类对象还是父类对象，而不是他的引用类型；例如Bird bird2 = new Chicken()时，我们关注的是其创建对象为Chicken类型，因此子类将继承父类的字段和方法，或者覆写父类的虚方法，而不用关注bird2的引用类型是否为Bird。引用类型不同的区别决定了不同的对象在方法表中不同的访问权限。

多态

同一对象呈现多种不同的状态；例如一个人在学校是学生有上课的技能，在工地就是工人有搬砖的技能；

实现多态的途径：虚方法实现多态；抽象类实现多态；接口是实现多态；

里氏转换（Liskov Substitution Principle）：派生子类对象能够替换其基类对象被使用：Bird bird =new Checkin(); 实际上显示的是父类类型指针指向的子类类型对象重写父类Eat后的方法, 这就是多态。

开放封闭原则（Open Closed Principle）：封装变化，降低耦合；对扩展开放，意味着有新的需求或变化时，可以对现有代码进行扩展，以适应新的情况。对修改封闭，意味着类一旦设计完成，就可以独立完成其工作，而不要对类进行任何修改。

虚方法基类可以实例化对象，抽象类不可以；区别在于基类是否需要实例化；

抽象类应主要用于关系密切的对象，而接口最适合为不相关的类提供通用功能。

1. 类库分析