**一：继承的特性**

1. **继承的单跟性**：一个子类中只能有一个父类
2. **继承的传递性**：派生类会以直接或间接的方式继承基类(包括基类所继承的另一个根基类)的所有的合法的成员，包括基类所重写的抽象成员、所实现的接口等等

**二：继承的注意事项**

1. 子类继承了父类的所有属性和方法，但是却没有继承私有字段，因为我们无法通过实例名.字段名来访问。（这只是字面上的意思，因为是私有的，我们访问不到，实际是继承了的）
2. 在c#中，所有的类都直接或者间接的继承了object类，例如：类名.ToString()这个方法，ToString()方法是怎么来的呢？ 就是继承object类产生的

**三：继承中，子类跟父类的构造函数的关系**

1. **继承中并没有继承父类中的构造方法，因为构造方法是通过传参的方式传进来的**

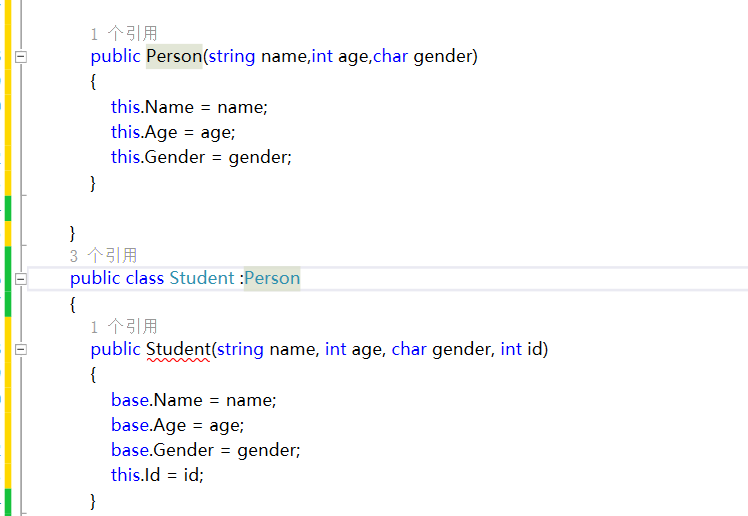
如图：

微信截图_20171210232524

所以当我们想继承父类中的构造方法是不行的

1. **当我们在父类中定义了有参数的构造方法，那么在子类中就会报错**

如图：



(Person是父类 Student是子类)

**为什么会发生这样的情况？**

**首先我们需要明白一个类的实例化必须回完成以下三种事情**

1. **在内存的堆中开辟一块新的内存空间**
2. **在堆中开辟的空间创建一个对象**
3. **调用对象的构造函数（默认为无参，并且如果写了其他构造函数，那么无参的构造函数将不会使用）**

**通过以上三个条件，才能够获取对象中的属性 字段 构造方法等等**

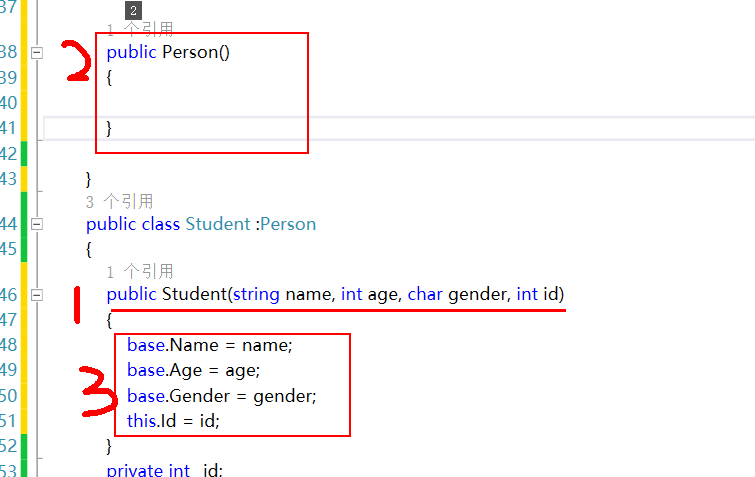
**那么当子类继承父类的时候，子类是通过什么方法来获取父类的属性和方法呢？**

我们先看下流程：

1. 子类继承了父类

2.我们对子类进行类的实例化

3.子类在内存中开辟新的内存空间，创建一个对象并调用构造方法，这时候构造方法不会继续走下去，会调用父类中的无参的构造方法



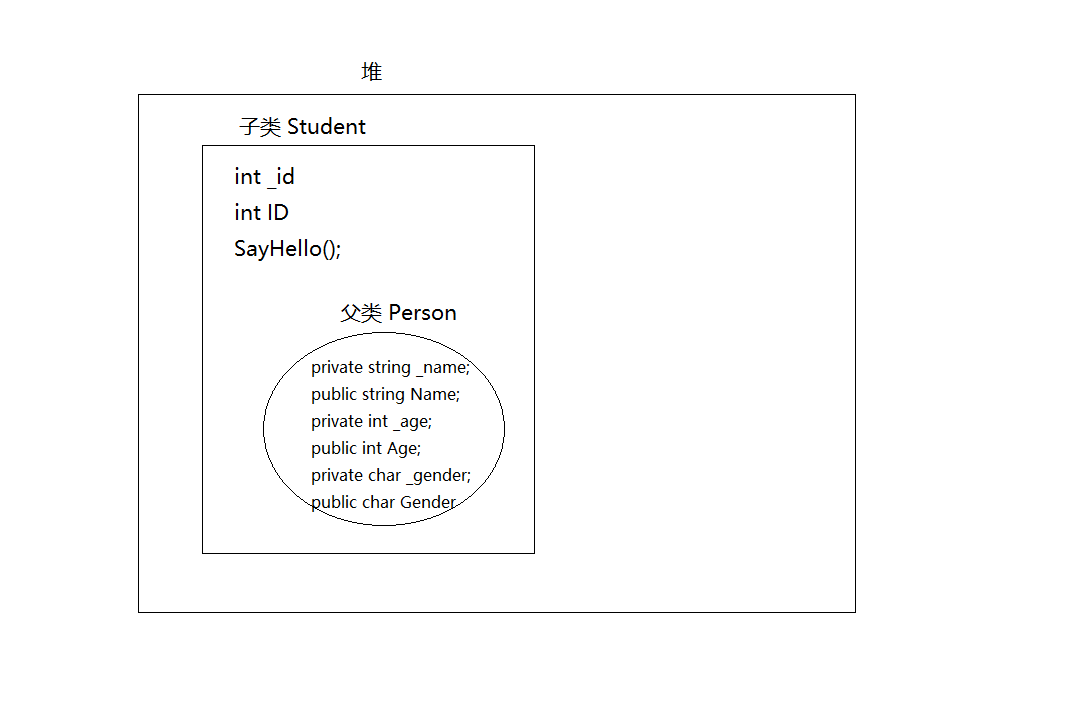
(步骤为1、2、3，如果第2步没有写无参的构造方法的话，系统也会自动调用)

1. 子类调用父类构造方法（调用默认的无参构造方法），父类在子类的内存空间中开辟一块新的内存空间，创建一个对象实例化，目的是为了让子类能够读取父类中的属性和方法

5.程序回到子类，父类中的所有属性字段和方法都实例化完毕，开始对子类中的内容进行实例化(由此我们可以得出，是因为父类在子类的内存空间中隐式完成了一次类的实例化，使得子类能够获取父类中的属性和方法)

我们来看下当在继承的时候 父类和子类的内存空间情况

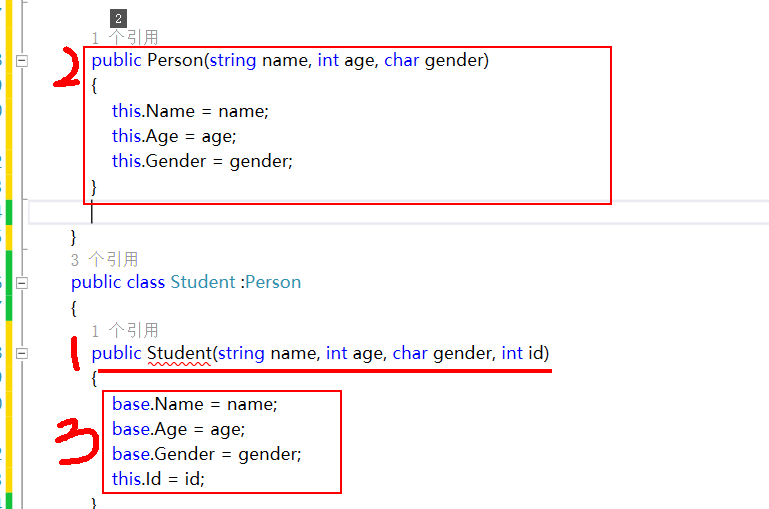
如图：



由图中可以看出，因为在继承的时候，父类由系统自动完成了1.开辟新的内存空间 2.在内存空间创建新的对象 3.子类调用父类的无参的构造方法（这样子类才能读取到了父类中的所有属性和方法）

所以，当我们这样写的时候是不行的

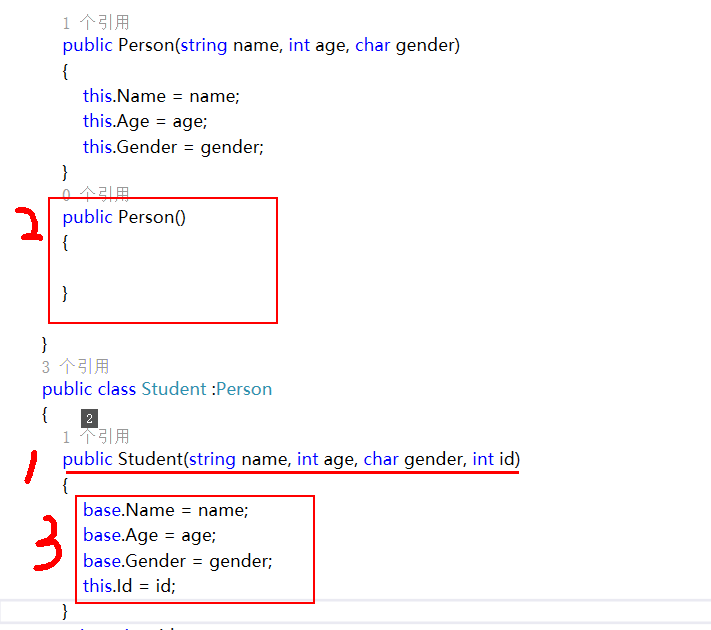
如下图：



程序运行步骤为1、2、3，当走到第二步的时候，因为子类实例化的时候，子类会默认优先的的调用无参的构造方法（目的是为了获取父类中的属性和函数等），而这个是有参的，这样父类就只完成了开辟新的内存空间、在内存空间中创建对象，而最后一步 子类调用默认的无参的构造方法就不能完成了，也就是无法完成实例化，子类无法获取父类中的属性和方法

除非，我们自己手动添加一个无参的构造方法写在下面，然后父类在实例化的时候有无参的构造方法给子类调用

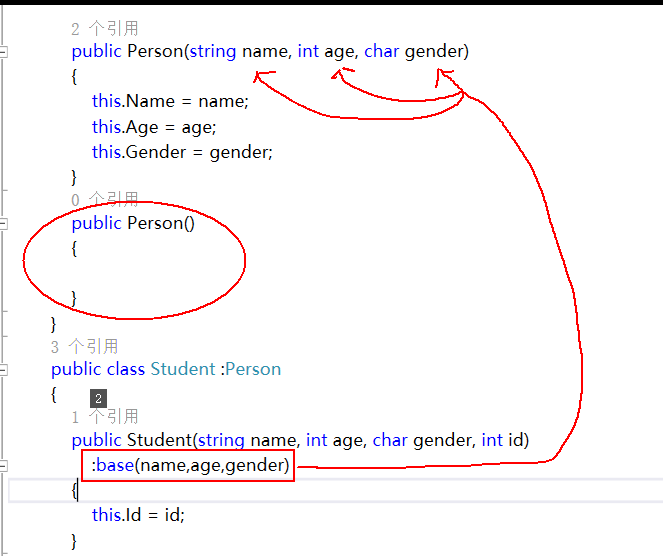
如图：



注意：**我们这样写了之后，父类就能调用默认的无参的构造方法了，但是一旦父类调用默认的无参构造方法，父类中的其他构造方法就不会调用了(步骤4)，当我们写子类的时候还是一样要重新赋值属性，这样的话当我们写多个子类的时候就会发生代码冗余的现象，而且父类的作用也没很好的体现出来**

我们可以通过 **:base()** 来让子类不再默认的调用无参的构造方法，而去调用我们需要的构造方法

如下图：

微信截图_20171210232524

1. 我们先对Studnet类进行实例化，然后在新建内存空间的时候把参数”吴鹏”，21，’男’，0423 的四个参数传给Student的构造方法
2. 通过 **:base()** 让父类不再调用默认的无参的构造方法，并且把Student类中构造方法接收到的4个参数，传了3个给父类中的构造方法
3. 父类中的构造方法再对属性进行赋值

**我们还能在子类中通过base.父类的属性 来访问或者调用父类中的属性**