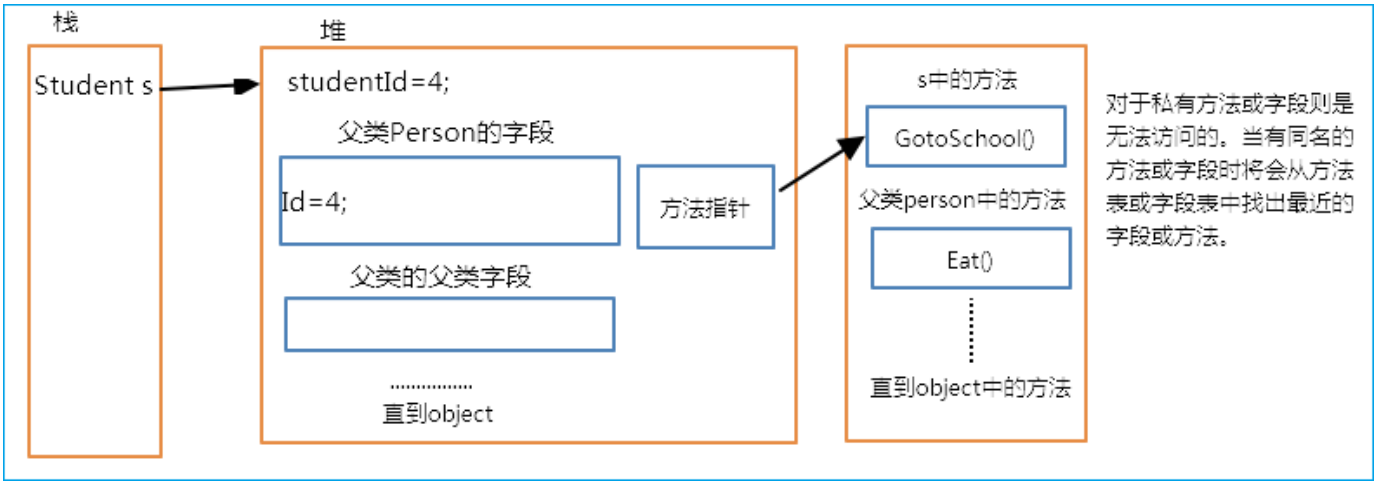
C#基础之内存分配 - 方小白

1.创建一个对象　　一个对象的创建过程主要分为内存分配和初始化两个环节。在.NET中CLR管理的内存区域主要有三部分：栈、GC堆、LOH堆，栈主要用来分配值类型数据。它的管理是有系统控制的，而不是像GC堆那样是由GC控制的。当线程执行完值类型实例所在方法后，这块空间将会被自动释放，一般栈的执行效率高不过容量有限。GC堆用来分配小对象实例，它是由GC完全控制内存的分配和回收。LOH堆则是为大对象实例准备的，它不会被压缩且只在GC完全回收时才会回收。在IL中可以看到newobj、ldstr(创建string对象)、newarr(用于分配新的数组对象)、box(装箱)等常见的创建对象的指令。当然在堆上也存在值类型，比如值类型作为类的字段时，它将存储在堆中的实例对象空间，还有装箱时也会让堆上存在值类型。好了接下来我们来看看创建一个对象的内存分配，现在有一个Person类和Student类。那么这句Student s = new Student() { studentId = 2, Id = 4 };执行完后s对象就被创建了，下面我画了张图来说明创建一个对象时内存的分配，其中s对象还有同步索引块与类型对象指针我没有画出来。

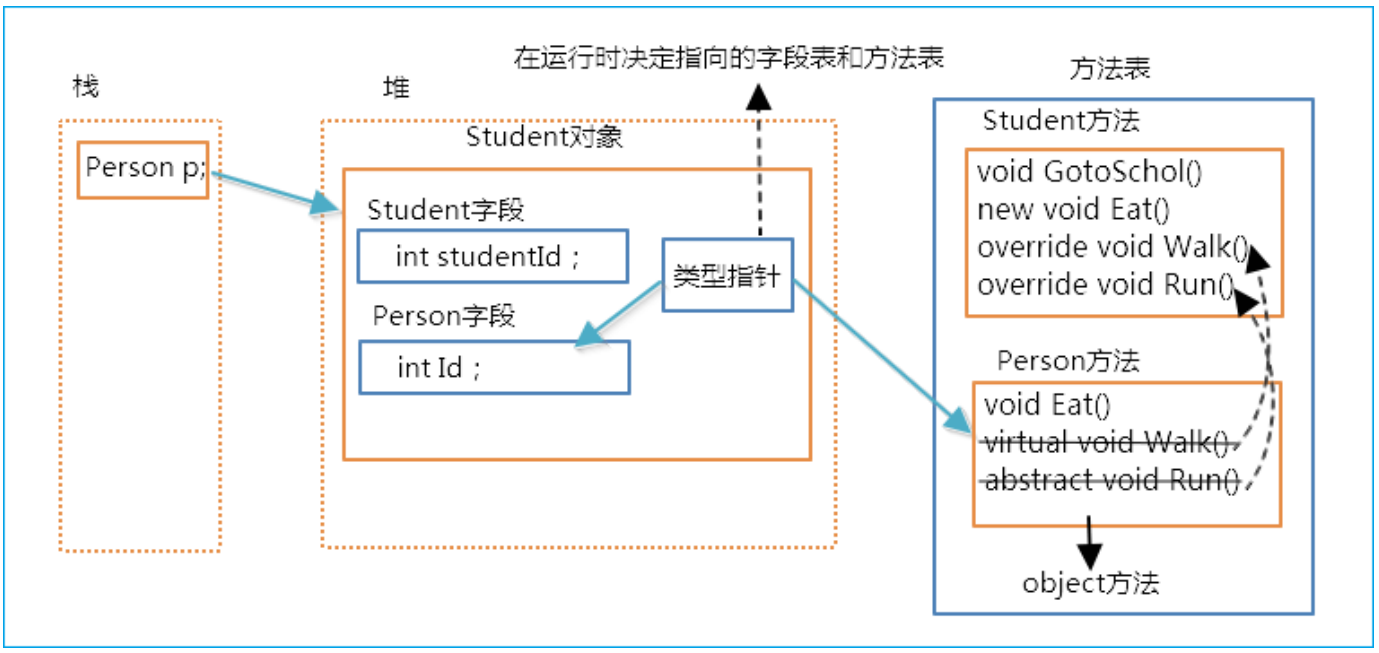
View Code



2.父类对象指向子类　　我们在写程序时为了实现多态一般都会使用父类对象指向子类。那么当我写入Person p=new Student();时便在堆中创建了一个子类对象，下面是关于父类对象指向子类的内存分配图。我在Person中添加了虚方法和抽象方法，并在Student子类重写了方法。从图中可以看出一旦子类重写了父类的虚方法或抽象方法，则Person方法表中的2个方法将会被子类覆盖，我们可根据它来实现多态。另外在Student方法表中还有一个new void Eat()方法，不过它是无法被p调用的因为此时的new Eat()属于子类。也就是说除了被覆盖的方法外，p只能调用Person方法表中的方法，如果找不到则会继续寻找Person父类的方法直到object。注意是不会往回找的，它不会去Student方法表中寻找方法。

View Code

View Code



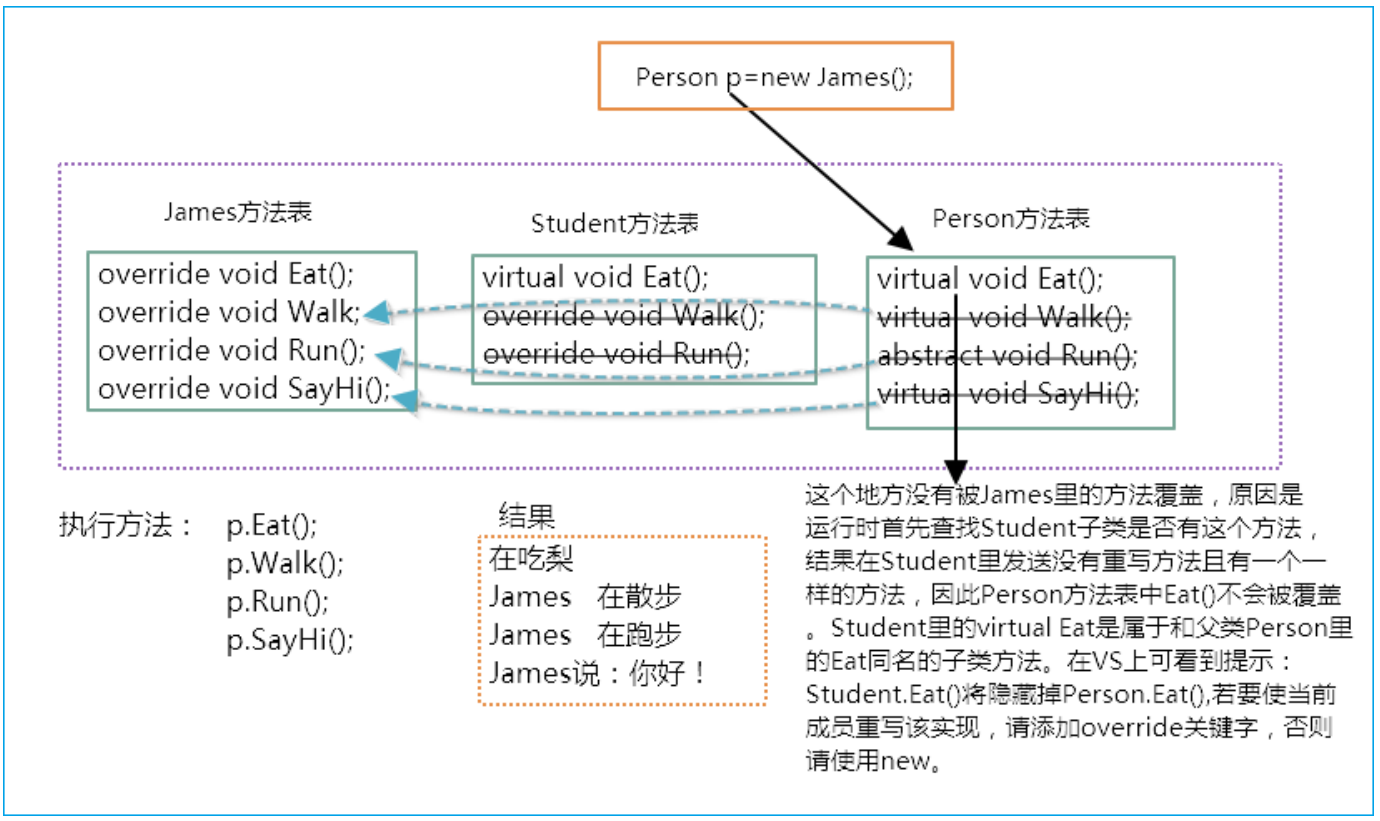
3.指向孙类对象　　现在我再添加一个Student的子类James，从上一个例子中已经知道只有override关键字重写的方法父类才会调用，因此我将普通方法全部删除。执行代码为Person p = new James() { name = "James", studentId = 2, Id = 4 };代码和内存分配图如下，为了突出重点，图中我就没有画字段了。从结果可以看到SayHi方法最后是被孙类的SayHi覆盖了，从这里可以看出继承的传递性！

```
public  abstract class Person
    {
        public  int Id;
        public virtual void Eat()
        {
            Console.WriteLine( "在吃梨");
        }
        public virtual void Walk()
        {
            Console.WriteLine("在散步");
        }
        //抽象方法只能在抽象类中声明，因此要在Person前加abstract，且只能声明并必须在子类中实现。
        public abstract void Run();
        public virtual void SayHi()
        {
            Console.WriteLine("人说：你好！");
        }
    }
```

```
public class Student:Person
    {
        public int studentId;
```

```
        public virtual void Eat()
        {
            Console.WriteLine("学生  在吃梨");
        }
        public override void Walk()
        {
            Console.WriteLine("学生  在散步");
        }
        public override void Run()
        {
            Console.WriteLine("学生  在跑步");
        }
    }
```

```
public class James:Student
    {
        public string name;
        public override void Eat()
        {
            Console.WriteLine("James  在吃梨");
        }
        public override void Walk()
        {
            Console.WriteLine("James  在散步");
        }
        public override void Run()
        {
            Console.WriteLine("James  在跑步");
        }
        public override  void SayHi()
        {
            Console.WriteLine("James说：你好！");
        }
    }
```


Person p=new James();
James方法表
override void Eat();
override void Walk;
override void Run();
override void SayHi();
Student方法表
virtual void Eat();
override void Walk();
override void Run();
Person方法表
virtual void Eat();
virtual void Walk();
abstract void Run();
virtual void SayHi();
这个地方没有被James里的方法覆盖，原因是运行时首先查找Student子类是否有这个方法，结果在Student里发送没有重写方法且有一个一样的方法，因此Person方法表中Eat()不会被覆盖。Student里的virtual Eat是属于和父类Person里的Eat同名的子类方法。在VS上可看到提示：Student.Eat()将隐藏掉Person.Eat(),若要使当前成员重写该实现，请添加override关键字，否则请使用new。
执行方法：
p.Eat();
p.Walk();
p.Run();
p.SayHi();
结果
在吃梨
James 在散步
James 在跑步
James说：你好！