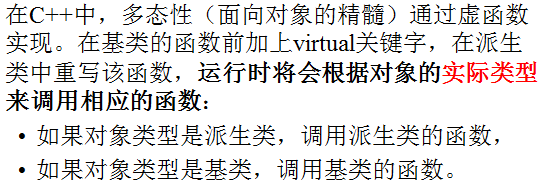
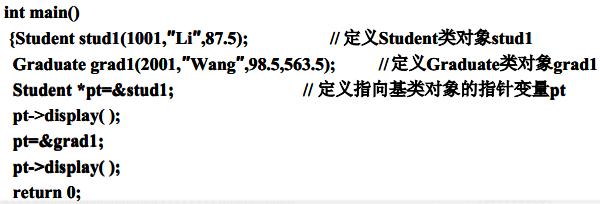
virtual functions(虚函数)和多态

意义：使程序更具有延展性和通用性

在设计模式上，对接口和抽象类编程，使结构更美观





【如果没有定义虚函数，因为display被重写，基类的display只能用域操作符调用，而不能直接写成

p.s 看代码比看文字直观多了……】

赋值兼容原则：派生类的对象可以被当成基类的对象来使用

【就是基类和派生类的指针可以互相指？例如和】

只有基类指针可以指向子类对象，子类指针是不能指向基类对象的，这个以前有讲过的…

虚函数与switch在使用上有点像，但switch费时且容易出错（可能没有列举完所有可能）

virtual functions

·代替了switch的作用

·在基类声明

就是函数前面加关键字“virtual”

·如果一个派生类没有定义virtual，就直接继承基类的函数（virtual就跟普通函数一样）

动态绑定and静态绑定

动态绑定（运行时候的多态性）

 编译到这条指令时，编译器不关联，在运行时根据类型选择调用哪个函数



【允许new类，在编译之后？？？】

可以把具体实现算法的派生类封装到加密的代码库中【不是很理解……】

静态绑定  就是只能运行特定的Draw

Pure Virtual Functions（纯虚函数）

格式：



虚函数：除非子类重写，否则跟一般函数一样

纯虚函数：没有实体，没有定义只有声明

Abstract and Concrete Classes

抽象基类：含有（或继承）一个或多个纯虚函数的类是抽象基类

不能实例化对象

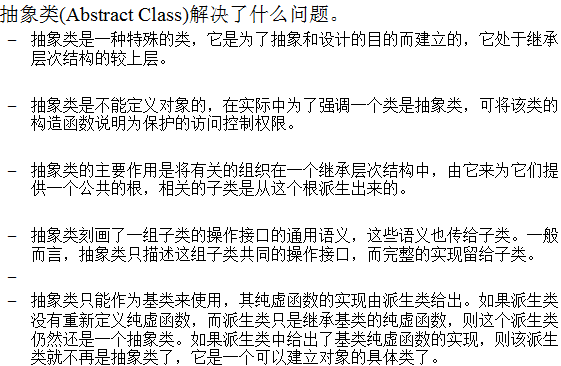
抽象类里面不需要给虚函数任何定义（虚函数只是接口）【这个虚函数应该是纯虚函数…】

可以有指针和引用，指针可以指向派生类，动态选择行为

具体类：没有纯虚函数的类

提供具体来创建一个真实的对象

除非在派生类中完全实现基类中所有的的纯虚函数，否则，派生类也变成了抽象类，不能实例化对象。



【就是将一般属性提取出来作为纯虚函数，然后下面派生类具体定义】

派生类与虚函数

派生类一般会重定义所继承的虚函数。派生类没有重定义某个虚函数，则使用基类中定义的版本。

派生类中虚函数的声明必须与基类中的定义方式完全匹配（函数名和参数列表完全相同）

但有一个例外：返回对基类型的引用（或指针）的虚函数。派生类中的虚函数可以返回派生类的引用（或指针）。

例如：Employee 类可以定义返回 Employee\* 的虚函数，如果这样，HourlyWorker 类中的虚函数重写可以定义为返回 Employee \* 或 HourlyWorker \*。

【这个在没有代码的情况下不是很理解……

就是首先，Employee\* 是基类里的虚函数

因为它是返回指针，所以在派生类里面函数名可以改变？】

返回值是指向该类对象的指针，所以只要是能指向该对象的指针类型就可以了，基类指针可以指向派生类对象，所以能返回基类指针，这样应该好理解一点吧…

虚析构函数

问题：如果用new运算符建立了临时对象，若基类中有析构函数，并且定义了一个指向该基类的指针变量。在程序用带指针参数的delete运算符撤销对象时，会发生一个情况: 系统会只执行基类的析构函数，而不执行派生类的析构函数。

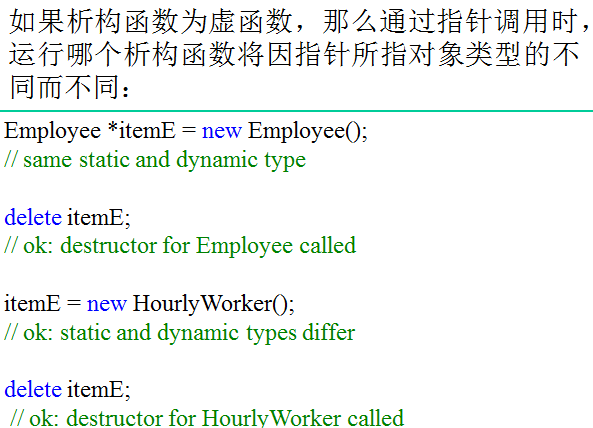
处理继承层次中的对象时，指针的静态类型可能与被删除对象的动态类型不同——删除实际指向派生类对象的基类类型指针。



这里希望detele释放p指向的内存空间，也就是 运行 Circle的析构函数

但是因为基类Point 定义了析构函数，所以只会调用基类的析构函数，而没有释放空间

解决办法：定义虚析构函数



即使析构函数没有工作要做，继承层次的根类也应该定义一个虚析构函数。

构造函数和赋值操作符不是虚函数

如果基类定义了虚析构函数，派生类的析构函数全部都为虚函数，即使函数名不同

虚基类

虚基类是相对于它的派生类而言的，它本身可以是一个普通的类。 只有它的派生类虚继承它的时候，它才称作虚基类，如果没有虚继承的话，就称为基类。

例如： class B ： virtual class A ，B虚继承A，A就是B的虚基类。如果没有虚继承，A是B的基类

