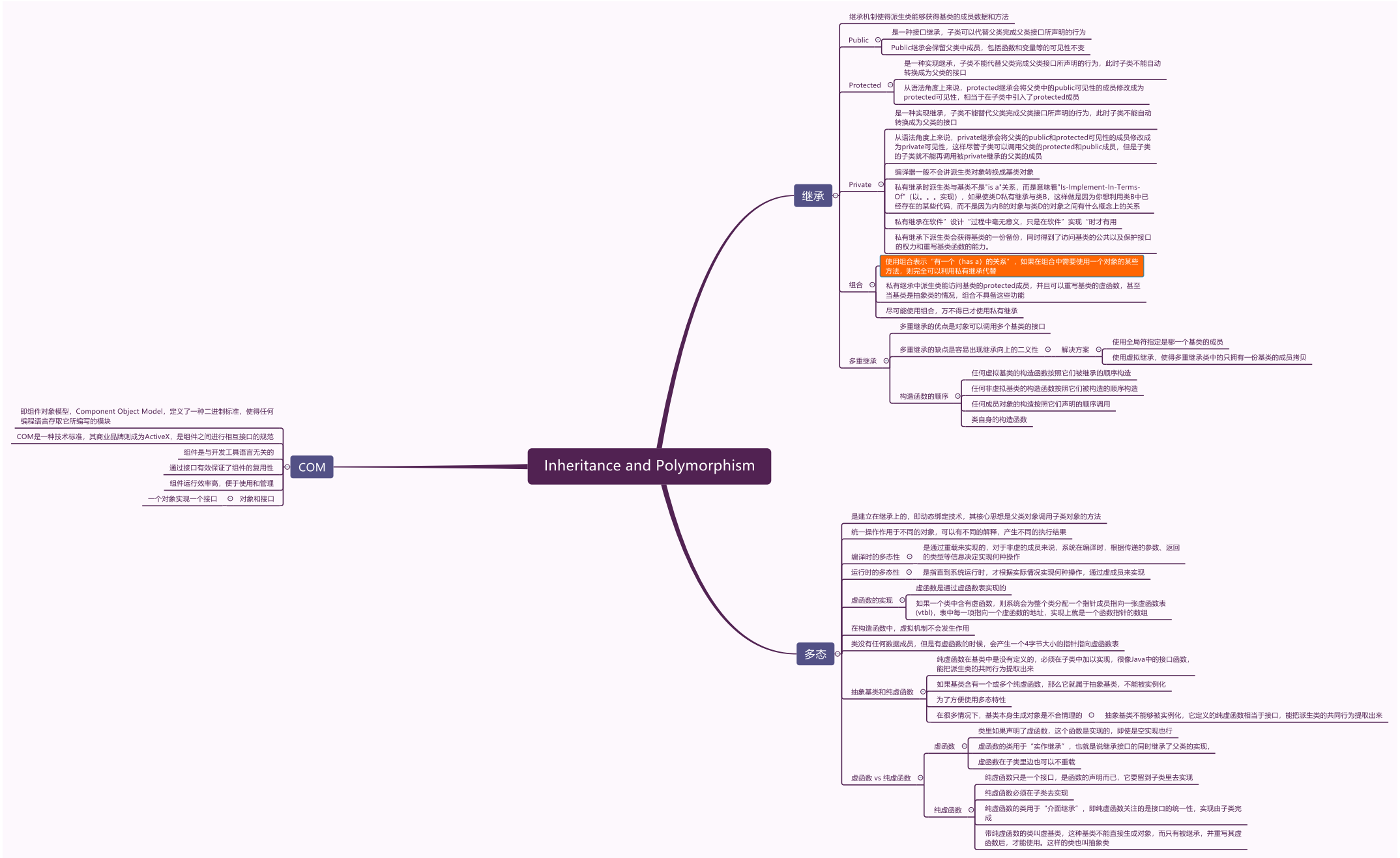
Inheritance and Polymorphism
继承
继承机制使得派生类能够获得基类的成员数据和方法
Public
是一种接口继承，子类可以代替父类完成父类接口所声明的行为
Public继承会保留父类中成员，包括函数和变量等的可见性不变
Protected
是一种实现继承，子类不能代替父类完成父类接口所声明的行为，此时子类不能自动转换成为父类的接口
从语法角度上来说，protected继承会将父类中的public可见性的成员修改成为protected可见性，相当于在子类中引入了protected成员
Private
是一种实现继承，子类不能替代父类完成父类接口所声明的行为，此时子类不能自动转换成为父类的接口
从语法角度上来说，private继承会将父类的public和protected可见性的成员修改成为private可见性，这样尽管子类可以调用父类的protected和public成员，但是子类的子类就不能再调用被private继承的父类的成员
编译器一般不会讲派生类对象转换成基类对象
私有继承时派生类与基类不是"is a"关系，而是意味着"Is-Implement-In-Terms-Of"（以。。。实现），如果使类D私有继承与类B，这样做是因为你想利用类B中已经存在的某些代码，而不是因为内B的对象与类D的对象之间有什么概念上的关系
私有继承在软件"设计"过程中毫无意义，只是在软件"实现"时才有用
私有继承下派生类会获得基类的一份备份，同时得到了访问基类的公共以及保护接口的权力和重写基类函数的能力。
组合
使用组合表示"有一个（has a）的关系"，如果在组合中需要使用一个对象的某些方法，则完全可以利用私有继承代替
私有继承中派生类能访问基类的protected成员，并且可以重写基类的虚函数，甚至当基类是抽象类的情况，组合不具备这些功能
尽可能使用组合，万不得已才使用私有继承
多重继承
多重继承的优点是对象可以调用多个基类的接口
多重继承的缺点是容易出现继承向上的二义性
解决方案
使用全局符指定是哪一个基类的成员
使用虚拟继承，使得多重继承类中的只拥有一份基类的成员拷贝
构造函数的顺序
任何虚拟基类的构造函数按照它们被继承的顺序构造
任何非虚拟基类的构造函数按照它们被构造的顺序构造
任何成员对象的构造按照它们声明的顺序调用
类自身的构造函数
多态
是建立在继承上的，即动态绑定技术，其核心思想是父类对象调用子类对象的方法
统一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果
编译时的多态性
是通过重载来实现的，对于非虚的成员来说，系统在编译时，根据传递的参数、返回的类型等信息决定实现何种操作
运行时的多态性
是指直到系统运行时，才根据实际情况实现何种操作，通过虚成员来实现
虚函数的实现
虚函数是通过虚函数表实现的
如果一个类中含有虚函数，则系统会为整个类分配一个指针成员指向一张虚函数表(vtbl)，表中每一项指向一个虚函数的地址，实现上就是一个函数指针的数组
在构造函数中，虚拟机制不会发生作用
类没有任何数据成员，但是有虚函数的时候，会产生一个4字节大小的指针指向虚函数表
抽象基类和纯虚函数
纯虚函数在基类中是没有定义的，必须在子类中加以实现，很像Java中的接口函数，能把派生类的共同行为提取出来
如果基类含有一个或多个纯虚函数，那么它就属于抽象基类，不能被实例化
为了方便使用多态特性
在很多情况下，基类本身生成对象是不合情理的
抽象基类不能够被实例化，它定义的纯虚函数相当于接口，能把派生类的共同行为提取出来
虚函数 vs 纯虚函数
虚函数
类里如果声明了虚函数，这个函数是实现的，即使是空实现也行
虚函数的类用于"实作继承"，也就是说继承接口的同时继承了父类的实现，
虚函数在子类里边也可以不重载
纯虚函数
纯虚函数只是一个接口，是函数的声明而已，它要留到子类里去实现
纯虚函数必须在子类去实现
纯虚函数的类用于"介面继承"，即纯虚函数关注的是接口的统一性，实现由子类完成
带纯虚函数的类叫虚基类，这种基类不能直接生成对象，而只有被继承，并重写其虚函数后，才能使用。这样的类也叫抽象类
COM
即组件对象模型，Component Object Model，定义了一种二进制标准，使得任何编程语言存取它所编写的模块
COM是一种技术标准，其商业品牌则成为ActiveX，是组件之间进行相互接口的规范
组件是与开发工具语言无关的
通过接口有效保证了组件的复用性
组件运行效率高，便于使用和管理
一个对象实现一个接口
对象和接口