4.1 继承

面向对象程序设计语言有三大特性：封装、继承和多态性。继承是面向对象语言的重要特征之一，没有继承的语言只能被称作“使用对象的语言”。继承是非常简单而强大的设计思想，它提供了我们代码重用和程序组织的有力工具。

类是规则，用来制造对象的规则。我们不断地定义类，用定义的类制造一些对象。类定义了对象的属性和行为，就像图纸决定了房子要盖成什么样子。

一张图纸可以盖很多房子，它们都是相同的房子，但是坐落在不同的地方，会有不同的人住在里面。假如现在我们想盖一座新房子，和以前盖的房子很相似，但是稍微有点不同。任何一个建筑师都会拿以前盖的房子的图纸来，稍加修改，成为一张新图纸，然后盖这座新房子。所以一旦我们有了一张设计良好的图纸，我们就可以基于这张图纸设计出很多相似但不完全相同的房子的图纸来。

基于已有的设计创造新的设计，就是面向对象程序设计中的继承。在继承中，新的类不是凭空产生的，而是基于一个已经存在的类而定义出来的。通过继承，新的类自动获得了基础类中所有的成员，包括成员变量和方法，包括各种访问属性的成员，无论是public还是private。当然，在这之后，程序员还可以加入自己的新的成员，包括变量和方法。显然，通过继承来定义新的类，远比从头开始写一个新的类要简单快捷和方便。继承是支持代码重用的重要手段之一。

类这个词有分类的意思，具有相似特性的东西可以归为一类。比如所有的鸟都有一些共同的特性：有翅膀、下蛋等等。鸟的一个子类，比如鸡，具有鸟的所有的特性，同时又有它自己的特性，比如飞不太高等等；而另外一种鸟类，比如鸵鸟，同样也具有鸟类的全部特性，但是又有它自己的明显不同于鸡的特性。

如果我们用程序设计的语言来描述这个鸡和鸵鸟的关系问题，首先有一个类叫做“鸟”，它具有一些成员变量和方法，从而阐述了鸟所应该具有的特征和行为。然后一个“鸡”类可以从这个“鸟”类派生出来，它同样也具有“鸟”类所有的成员变量和方法，然后再加上自己特有的成员变量和方法。无论是从“鸟”那里继承来的变量和方法，还是它自己加上的，都是它的变量和方法。

本周的学习，建议按照视频中的操作，自己把媒体资料库从头到尾实现一遍，重点在于弄清楚子类和父类的关系，理解子类从父类继承得到了什么。

Source --- Generate Constructor using Fields可以帮助快速初始化成员函数

我们把用来做基础派生其它类的那个类叫做父类、超类或者基类，而派生出来的新类叫做子类。Java用关键字extends表示这种继承/派生关系：

class ThisClass extends SuperClass {      //… }

继承表达了一种is-a关系，就是说，子类的对象可以被看作是父类的对象。比如鸡是从鸟派生出来的，因此任何一只都可以被称作是一只鸟。但是反过来不行，有些鸟是鸡，但并不是所有的鸟都是鸡。如果你设计的继承关系，导致当你试图把一个子类的对象看作是父类的对象时显然很不合逻辑，比如你让鸡类从水果类得到继承，然后你试图说：这只本鸡是一种水果，所以这本鸡煲就像水果色拉。这显然不合逻辑，如果出现这样的问题，那就说明你的类的关系的设计是不正确的。Java的继承只允许单继承，即一个类只能有一个父类。

4.2 子类父类关系

对理解继承来说，最重要的事情是，知道哪些东西被继承了，或者说，子类从父类那里得到了什么。答案是：所有的东西，所有的父类的成员，包括变量和方法，都成为了子类的成员，除了构造方法。构造方法是父类所独有的，因为它们的名字就是类的名字，所以父类的构造方法在子类中不存在。除此之外，子类继承得到了父类所有的成员。

但是得到不等于可以随便使用。每个成员有不同的访问属性，子类继承得到了父类所有的成员，但是不同的访问属性使得子类在使用这些成员时有所不同：有些父类的成员直接成为子类的对外的界面，有些则被深深地隐藏起来，即使子类自己也不能直接访问。下表列出了不同访问属性的父类成员在子类中的访问属性：

|  |  |  |
| --- | --- | --- |
| **父类成员访问属性** | **在父类中的含义** | **在子类中的含义** |
| public | 对所有人开放 | 对所有人开放 |
| protected | 只有包内其它类、自己和子类可以访问 | 只有包内其它类、自己和子类可以访问 |
| 缺省 | 只有包内其它类可以访问 | 如果子类与父类在同一个包内：只有包内其它类可以访问  否则：相当于private，不能访问 |
| private | 只有自己可以访问 | 不能访问 |

public的成员直接成为子类的public的成员，protected的成员也直接成为子类的protected的成员。Java的protected的意思是包内和子类可访问，所以它比缺省的访问属性要宽一些。而对于父类的缺省的未定义访问属性的成员来说，他们是在父类所在的包内可见，如果子类不属于父类的包，那么在子类里面，这些缺省属性的成员和private的成员是一样的：不可见。父类的private的成员在子类里仍然是存在的，只是子类中不能直接访问。我们不可以在子类中重新定义继承得到的成员的访问属性。如果我们试图重新定义一个在父类中已经存在的成员变量，那么我们是在定义一个与父类的成员变量完全无关的变量，在子类中我们可以访问这个定义在子类中的变量，在父类的方法中访问父类的那个。尽管它们同名但是互不影响。

在构造一个子类的对象时，父类的构造方法也是会被调用的，而且父类的构造方法在子类的构造方法之前被调用。在程序运行过程中，子类对象的一部分空间存放的是父类对象。因为子类从父类得到继承，在子类对象初始化过程中可能会使用到父类的成员。所以父类的空间正是要先被初始化的，然后子类的空间才得到初始化。在这个过程中，如果父类的构造方法需要参数，如何传递参数就很重要了。

关于super()

1. 能在一个构造函数里调用两次super()吗？
2. super()必须在构造函数的第一行吗？

Ans:

一个构造函数里面不能调用两次super（），super（）必须写在第一行，因为程序初始化顺序为：1）父类定义初始化；2）父类构造器初始化；3）子类定义初始化；4）子类构造器初始化。

如果不是这样，会出现“Constructor call must be the first statement in a constructor”错误。

4.3 多态变量和向上造型

类定义了类型,DVD类所创建的对象的类型就是DVD。类可以有子类,所以由那些类定义的类型可以有子类型。在DoME的例子中,DVD类型就是Item类型的子类型。

子类型类似于类的层次,类型也构成了类型层次。子类所定义的类型是其超类的类型的子类型。

当把一个对象赋值给一个变量时,对象的类型必须与变量的类型相匹配,如:

    Car myCar = new Car();

是一个有效的赋值,因为Car类型的对象被赋值给声明为保存Car类型对象的变量。但是由于引入 了继承,这里的类型规则就得叙述得更完整些:

**一个变量可以保存其所声明的类型或该类型的任何子类型。**

对象变量可以保存其声明的类型的对象,或该类型的任何子类型的对象。

Java中保存对象类型的变量是多态变量。“多态”这个术语(字面意思是许多形态)是指一个变量可以保存不同类型(即其声明的类型或任何子类型)的对象。

子类的对象可以赋值给父类的变量；

例：

Vehicle v1 = new Vehicle();

Vehicle v2 = new Car();

Vehicle v3 = new Bicycle();

其中 Car 和 Bicycle 是Vehicle的子类。

子类的对象可以传递给需要父类对象的函数。

子类对象可以赋值给父类变量。

多态变量：一个父类变量可以被赋值任一子类对象。

！！！Java 中 不存在 对象对对象的赋值，而是使得新变量和原来变量都成为 对象的管理者！！！

造型：将一个对象强制换成其可以转换的类型赋值给一个变量（例如父类转为子类）

例子：

其中item为 CD的父类

Item item = new Item();

CD cd = new CD();

Item = cd;

CD cc = item;( 编译报错 )

CD cc= (CD) item (编译通过，此处为造型)

造型并不总能通过编译，要看原来变量的类型。

向上造型： 将一个子类对象赋给父类变脸

类型转换和造型不同

Int i = (int)10.2;

4.4 多态

如果子类的方法覆盖了父类的方法，我们也说父类的那个方法在子类有了新的版本或者新的实现。覆盖的新版本具有与老版本相同的方法签名：相同的方法名称和参数表。因此，对于外界来说，子类并没有增加新的方法，仍然是在父类中定义过的那个方法。不同的是，这是一个新版本，所以通过子类的对象调用这个方法，执行的是子类自己的方法。

覆盖关系并不说明父类中的方法已经不存在了，而是当通过一个子类的对象调用这个方法时，子类中的方法取代了父类的方法，父类的这个方法被“覆盖”起来而看不见了。而当通过父类的对象调用这个方法时，实际上执行的仍然是父类中的这个方法。注意我们这里说的是对象而不是变量，因为一个类型为父类的变量有可能实际指向的是一个子类的对象。

当调用一个方法时，究竟应该调用哪个方法，这件事情叫做绑定。绑定表明了调用一个方法的时候，我们使用的是哪个方法。绑定有两种：一种是早绑定，又称静态绑定，这种绑定在编译的时候就确定了；另一种是晚绑定，即动态绑定。动态绑定在运行的时候根据变量当时实际所指的对象的类型动态决定调用的方法。Java缺省使用动态绑定。

多态：一个父类变量可以赋不同类型的类的对象

当通过对象变量调用函数的时候，调用哪个函数就叫做绑定。

静态绑定：根据变量的声明类型来绑定

动态绑定：根据变量的动态类型来绑定

覆盖 Override

子类和父类中存在名称和参数完全相同的函数，这一对函数构成覆盖关系。

通过父类的变量调用该函数时，会调用当时变量所管理的对象所属的类的函数。

单根系统：

Java中为object类

已学过的object类中可继承的函数有toString, equals

Equals只能判断两个变量是否管理同一个对象，而不能判断两个对象是否完全相同。