今天主要学习：多态，抽象类，接口。很重要

多态----------

多态：一个类型有多种形态

比如：一个学生，也可以说成是一个人，也可以说成一个脊椎动物，也可以说成一个动物。。。。。。

多态前提

要有继承（extends）或实现(implements)关系

要有方法的重写或实现（没有也叫多态，但没意义）

要有父类（class）或父接口(interface)的引用指向子类对象

但我们一般把实现一个接口也看作是继承的一种，所以说老师里面讲的也是对的

多态中的成员访问特点之成员变量

用变量类型的成员变量

用实际类型的方法

静态的只看类型的类

基于这种情况：所有的属性都要私有（静态常量除外），然后对外提供访问方法。

多态中向上转型和向下转型

用父类变量指向子类---->向上转

用子类变量接收父类变量，要强转------>向下转

强转有可能出错：用instanceof进行判断，格式：对象 instanceof 类型;意思是：这个对象属于这个类型吗？返回的是 boolean

多态的好处和弊端

* A:多态的好处
  * a:提高了代码的维护性(继承保证)
  * b:提高了代码的扩展性(由多态保证)
* B:多态的弊端
  * 不能使用子类的特有属性和行为。

笔记：###09.08的代码是报错的。

抽象类----------

抽象类：把一类事物，相象的部分抽取出来，组成的类，因为这样的类不是具体的事物，所以不能实例化，用关键字：abstract修饰。

只是知道这部分很象，但一样不一样，还不知道。

抽象类格式：

abstract class 类名{ };

抽象类里面有抽象方法：(抽象类不一定有抽象方法，有抽象方法的类一定是抽象类或者是接口)

只知道子类里面一定有这个方法，但这个方法怎么实现的，还不知道

抽象方法格式

public abstract 返回值类型 方法名（参数列表）;

抽象类与普通类的功能区别是：抽象类里面可以定义抽象方法，抽象类不能直接实例化（创建实例，创建对象）

abstract不能和哪些关键字共存

abstract和static

被abstract修饰的方法没有方法体

被static修饰的可以用类名.调用,但是类名.调用抽象方法是没有意义的

abstract和final

被abstract修饰的方法强制子类重写

被final修饰的不让子类重写,所以他俩是矛盾

abstract和private

被abstract修饰的是为了让子类看到并强制重写

被private修饰不让子类访问,所以他俩是矛盾的

接口----------

接口:功能的扩展;接口只能按照多态的方式来实例化。

接口格式:

interface 接口名 {}

使用格式:

class 类名 implements 接口名 {}

接口成员特点

* 成员变量只能是常量，并且是静态的并公共的。
    * 默认修饰符：public static final
* 构造方法：接口没有构造方法。
* 成员方法：只能是抽象方法。
    * 默认修饰符：public abstract

类与类,类与接口,接口与接口的关系

* a:类与类：
    * 继承关系,只能单继承,可以多层继承。
* b:类与接口：
    * 实现关系,可以单实现,也可以多实现。
    * 并且还可以在继承一个类的同时实现多个接口。
* c:接口与接口：
    * 继承关系,可以单继承,也可以多继承。

emp:

类可只能单继承，但可以多实现！

```
interface 接口 extends 接口1,接口2 {}
class 类 extends 抽象类 implements 接口1, 接口2{}
```

接口里面的方法只能是抽象方法，属性都是常量，可以多继承接口。

抽象类，除了直接new对象，其它的和正常类一样。

抽象类是从类中抽出相似方法和属性，接口是对类的扩展。

熊明春：接口设计的是行为规范，类才是事物的抽象，事物才有多种形态，所以多态着重体现在类的继承和方法重写上