关于默认构造方法的纠错

在创建新类的时候可以发现每一个类都是从Object类继承而来

**复习继承**

**Java多态（Polymorphism）**

  所谓多态就是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量倒底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。

比如在大学校园里看到三个人，他们看上去很年轻，其中一个是本科学生，一位是研究生，一位是年轻助教。但你无法确定他们分别的身份。这三个种可以表示成：

Person A, Person B, Person C

因为子类继承了相同父类，他们都可以进行向上转型。因为所有的学生，毕业生，老师都是Person。在创建实例的时候并不能确定他们的具体类型，可以把他们都声明成人。

只有在问了他的学历时才能知道他们分别是什么。

Person A = Student;

Person B = Graduate;

Person C = Teacher;

再次强调，这里是定义了三个引用类型的变量，类型均为Person型。

但是A,B,C 分别指向堆内存的一个Student，一个Graduate，一个Teacher对象。但是他们都向上转型了，所以他们都只能访问父类Person类中的方法和属性而对于子类中存在而父类中不存在的方法，该引用是不能使用的，尽管是重载该方法。若子类重写了父类中的某些方法，在调用该些方法的时候，必定是使用子类中定义的这些方法（动态连接、动态调用）。

例子：

      对于面向对象而已，多态分为编译时多态和运行时多态。其中编辑时多态是静态的，主要是指方法的重载，它是根据参数列表的不同来区分不同的函数，通过编辑之后会变成两个不同的函数，在运行时谈不上多态。而运行时多态是动态的，它是通过动态绑定来实现的，也就是我们所说的多态性。

原则：

Java实现多态有三个必要条件：继承、重写、向上转型。

         继承：在多态中必须存在有继承关系的子类和父类。

         重写：子类对父类中某些方法进行重新定义，在调用这些方法时就会调用子类的方法。

         向上转型：在多态中需要将子类的引用赋给父类对象，只有这样该引用才能够具备技能调用父类的方法和子类的方法。

         只有满足了上述三个条件，我们才能够在同一个继承结构中使用统一的逻辑实现代码处理不同的对象，从而达到执行不同的行为。

      对于Java而言，它多态的实现机制遵循一个原则：当超类对象引用变量引用子类对象时，被引用对象的类型而不是引用变量的类型决定了调用谁的成员方法，但是这个被调用的方法必须是在超类中定义过的，也就是说被子类覆盖的方法。

# **抽象类与接口**

类的修饰符号有：public 公共类；default 默认包内访问权限（即class前什么都不假）；abstract 抽象类； final 不能被继承的类； static 静态类； private 只能修饰内部类，一般不推荐使用。（回答毛子桓）

我们都知道在面向对象的领域一切都是对象，同时所有的对象都是通过类来描述的，但是并不是所有的类都是来描述对象的。如果一个类没有足够的信息来描述一个具体的对象，而需要其他具体的类来支撑它，那么这样的类我们称它为抽象类。比如new Animal()，我们都知道这个是产生一个动物Animal对象，但是这个Animal具体长成什么样子我们并不知道，它没有一个具体动物的概念，所以他就是一个抽象类，需要一个具体的动物，如狗、猫来对它进行特定的描述，我们才知道它长成啥样。

例子: abstract关键字定义抽象类

内存分析

从这里可以看出，抽象类就是为了继承而存在的，如果你定义了一个抽象类，却不去继承它，那么等于白白创建了这个抽象类，因为你不能用它来做任何事情。对于一个父类，如果它的某个方法在父类中实现出来没有任何意义，必须根据子类的实际需求来进行不同的实现，那么就可以将这个方法声明为abstract方法，此时这个类也就成为abstract类了。

包含抽象方法的类称为抽象类，但并不意味着抽象类中只能有抽象方法，它和普通类一样，同样可以拥有成员变量和普通的成员方法。注意，抽象类和普通类的主要有三点区别：

1）抽象方法必须为public或者protected（因为如果为private，则不能被子类继承，子类便无法实现该方法），缺省情况下默认为public。

2）抽象类不能用来创建对象；

3）如果一个类继承于一个抽象类，则子类必须实现父类的抽象方法。如果子类没有实现父类的抽象方法，则必须将子类也定义为为abstract类。

在其他方面，抽象类和普通的类并没有区别。因此，继承抽象类也只能单继承。所以这里不用Person类做例子。

注意：

1、抽象类**不能被实例化**，实例化的工作应该交由它的子类来完成，它只需要有一个引用即可。

2、抽象方法**必须**由子类来进行重写。

3、只要包含一个抽象方法的抽象类，该类**必须**要定义成抽象类，不管是否还包含有其他方法。

4、抽象类中可以包含具体的方法，当然也可以不包含抽象方法。

5、子类中的抽象方法不能与父类的抽象方法同名。

6、abstract不能与final并列修饰同一个类。

7、abstract 不能与private、static、final或native并列修饰同一个方法。

接口：

Interface是抽象方法和常值量的定义的集合。

从本质上讲，接口是一种特殊的抽象类，这种抽象类中只包含常量和方法的定义，而没有变量和方法的实现。不能有任何写好的方法！

为何使用接口？比如说，一个Student不止是个人，他可以是属于一个Sportsman类，可以是一个Volunteer类，可以是一个leader类中。由于java只能单继承，那怎么让一个Student可以表现出其他类的特点呢？使用接口

接口定义例子:

public interface Runner{

public static final int id = 1;

public abstract void start();

public abstract void run();

public void stop();

}

注意:

接口中所有的成员变量都必须是public static final的，即使不定义，也一定是这个。这是因为如果要实现多继承，每一个创建的子类对象都包含了一个父类对象，如果各个父类对象之间的成员变量还会发生联系，会造成混乱。Java避免了这种混乱，而是用了static final。静态的成员变量不属于特定对象，而是所有的共有的，是独立存在的，因此可以多继承。

所有的方法都必须是public abstract的，即使不写，也会自动是public abstract的。

接口可以继承其他接口，并添加新的属性和抽象方法，但不能继承类。为什么？

接口的实现：implements 关键字，所有方法都需要重写

内存分析

下节课：

1. 工厂模型
2. 命令模型