# 一. 高级特性： 关键字： static final abstract

在开始之前，我们有必要来复习一下‘多态’相关的知识点：

1. 多态的原则： 满足开----闭原则。(后面会再讲到此原则，先有个概念)：在不影响原有功能的基础上有可扩展性。
2. 多态的应用： 把共性都抽取出来，而个性才分到具体的子类中。
3. 注意： 我们在设计的时候，尽可能地在方法调用，返回值类型，形参，都应该应用多态。这样程序的复用性，可扩展性都很好，还可以降低耦合性。

如：

```
package day05;
public class TestPoly {
   public static void main(String[] args) {
      // TODO Auto-generated method stub
      Vehicle v = getVehicle(Integer.parseInt(args[0]));
      goHome(v);
   }
   /************
    * 注:此处为了方便,我把以下两个方法都写成了静态的.

    * 此方法是为了获得一种交通工具,它属于一种工厂,专门用来生产'交通

工具'

    * @param type: 根据此类型参数,来确定生产哪种'交通工具'
    */
   public static Vehicle getVehicle(int type) {
      if(type == 0) return new Bike();
      else if(type == 1) return new Car();
      else return new Plane();
   }
   /*********************************
    * 此方法使用 Vehicle(交通工具) 来做为形参.在实现在使用它的

run()方法.
    * @param v
    */
   public static void goHome(Vehicle v) {
      v.run();
   }
}
```

```
class Vehicle {
   public void run() { }
}
class Bike extends Vehicle {
   public void run() {
      System.out.println("go home by Bike");
   }
}
class Car extends Vehicle {
   public void run() {
      System.out.println("go home by car");
   }
}
class Plane extends Vehicle {
   public void run() {
      System.out.println("go home by plane");
   }
}
```

以上是我们对于多态的一个例子，希望能让大家对多态的理解有所帮助。

---

## 关键字： static

可以修饰： 属性， 方法， 代码块 （局部变量不能做静态）

修饰属性： static 属性： 表示此属性是属于整个类的，不属于某一对象，所以它也叫类变量，存放在代码空间。

概念： 类加载的过程?

现在我们思考一个问题，JAVA虚拟机怎么来加载类文件的?

： JAVA虚拟机通过输入流来读入JAVA字节码文件的过程。就是类加载的过程。

那么我们说，在类加载时，JAVA虚拟机就会为类静态属性初始化，初始值为0

或null;

注：静态实例变量可以用类名直接来调用。如： Student.id; 当然也可以用对象来拿，但不建议这么做。

现在，我们来比较一下： 静态实例变量， 实例变量， 局部变量的区别?

1. 静态实例变量也叫类变量，类加载时就开始初始化，类卸载时消亡，存放在代码区。
2. 实例变量，有了对象之后才有，有初始值，存放在堆区，随对象而存在
3. 局部变量，只在定义它的代码块内有效，存放在栈区.

修饰方法： 静态方法：

特点：在静态方法中不能访问非静态成员（含属性和方法）

静态方法可以被覆盖，但没有多态。所以，JAVA虚拟机对于静态方法的选择，看编译时类型，而不是运行时类型。

注： 父类静态的方法只能由子类静态的方法覆盖，非静态的也只能由非静态的覆盖。

记住：静态，只有类的概念，不存在对象的概念，也不能使用 this 关键字．可以用类名直接调用静态方法。

思考：主方法为什么一定要是静态方法?

答：在用JAVA命令启动JVM（虚拟机）时，由于MAIN方法是一个很特殊的方法，这是一个程序的入口方法，根据JVM加载

类文件时的要求，在加载的时候还不存在任何对象，那如何来调用MINA方法呢？除非MAIN是静态的。

Static 代码块： 静态初始化块

静态代码块一定得放在类里面，但不在任何的方法里面，它在：当类被加载时就会被执行，它只会被执行一次。

根据静态初始化块的这个特性：我们可以写出一个没有MAIN方法的程序。

```
Public class TestStatic {
   Static {
      System.out.pritnln(“HELLO,WORLD!!!”);
      System.exit(0); //执行完后就中止程序。
   }
}
```

注：在什么情况下JVM会去加载类字节码文件呢？

想想： Student s1 = null; //此种情况JVM 是不会加载 Student 类的。

Student s2 = new Student();//只有 new 了以后，才会真正去加载这个类文件。

这是一种JVM优化算法---延迟加载，以勉生成一个不用的对象，造成空间

浪费。

注： 如果有了继承，要加载子类，必先会加载父类。

- static 设计模式之应用： 单态模式： Singleton

```
package day06;
public class SingletonDemo {
   public static void main(String[] args) {
      // TODO Auto-generated method stub
      Husband h1 = Husband.newInstance();
      Husband h2 = Husband.newInstance();
      System.out.println(h1);
      System.out.println(h2);
      System.out.println(h1 == h2);
   }
}
/**********************
 * 单态模式演示程序. 这是'懒汉式'的单态.
 * @author yejf
 */
class Husband { // java Husband
   //此实例变量一定要是 static, 它是类变量,不属于某一个对象.
   private static Husband instance = new Husband("韦小宝");
   private String name;
   /*************
    * @param name
    * 此处一定要做成'私有'的,防止外面自己来构造对象.
    */
   private Husband(String name) {
      this.name = name;
   }
   /*****************
    * 此方法是提供给外面唯一的接口,去获得一个 Husband 对象.
    * @return
    */
   Public static Husband newInstance() {
         rerutn instance;
   }
}
```

单态模式非常重要，大家一定要掌握。记住三点：

1．提供一个私有静态本类型的类变量，如：private static 类名 instance = new 类名();

2．构造方法一定是 私有; private

3．提供一个公共的对外接口，一般格式：public static 类名 newStance() { }

## 关键字： final

Final 修饰： 变量（包含成员变量和局部变量）， 方法， 类

1. 修饰局部变量： 表示此局部变量是一个常量，不能更改；在定义时必须赋值。
2. 修饰实例变量： 也表示一个常量，但它的赋值有两种方式：
   i. 在定义的时候直接初始化
   ii. 在构造方法中初始化。 注：这两次不能同时都做，但又不能都不做。如果此实例变量又是静态的，则要么直接初始化，要么在静态代码块中再赋值。
3. 修饰方法： 表示此方法是一个不能覆盖的方法，这样可以提高程序的稳定性。
4. 修饰类： 表示此类是一个最终类，不能被继承。

记住： 一个 final 类中，所有的方法默认都是 final 的。

Final 模式之应用。 不变模式：

String 类就是一个 final 类， 它不能被继承。所以我们在使用 String 类时， 最好这样定义：

String s1 = “123”; 而不是： String s2 = new String(“123”);

因为第一种情况，JVM会把它放到串池中，给大家共享； 第二种情况存放在堆区，不能共享。

依据不变对象的特性，所以 String 对象的互想操作可能会造成大量的垃圾对象，影响效率。

所以，如果涉及到大量字符串的+（连接）操作，我们建议使用 StringBuffer 来替代 String.

## 关键字： abstract

修饰： 类， 方法；

Abstract class：叫做抽象类，抽象类不能实例化一个对象。

Abstract method：叫做抽象方法，抽象方法只有定义，没有实现，用；结束，

没有{}

记住： 抽象类中不一定要有抽象方法，但有一个抽象方法的类就一定是抽象类。

抽象类被设计用来被其它类继承，抽象方法用于被子类来覆盖，如果一个类继承了一个抽象类，则又没有实现抽象类的所有
方法，则这个类还将是一个抽象类。
所以我们说，如果子类覆盖了父类抽象方法，也叫做子类实现了抽象的方法；
由于抽象类不能实例化一个对象，所以它不能做为运行时类型，而只能做为编译时类型。
如：

```
abstract class Animal {
   public abstract void eat();
}
Class Dog extends Animal {
   Public void eat() { }
}
```

在 main 中， 可以：

```
Animal a1 = new Animal(); // error.
Animal a = new Dog(); //ok
a.eat(); //会有多态特性。
```

* 在我们设计多个类或对象时，应尽量地把它们的共性抽取出来，放到一个基类中，可使得类或对象之间的耦合性降低。这样也
可以更好地满足多态。

注： static , final, abstract 都不能用来修饰 构造方法。

对于： private, static, final 它们三个之间可以任意混用，但都不能与 abstract 一起使用。

思考： final 和 abstract 为什么不能一起使用?

答： final 决定了这个类或方法再也不能被继承或覆盖，而 abstract 就是

为了类继承或方法被覆盖而定义的，
它们本来就自相矛盾，所以绝不能一起使用。

作者：叶加飞 (steven ye)
mailto: yejf@tarena.com.cn
加拿大.达内科技 (上海中心)