多态传参：引用传递，修改内容形参影响实参，但形参指向其他的对象不影响实参

编程思想：

1 一段代码完成一个小功能

2 先找不正确的条件（过关斩将，过了就继续，过不了就打道回府）

3 项目较复杂，采用分层模式（明确系统需要哪些类、类鱼类之间的调用关系）

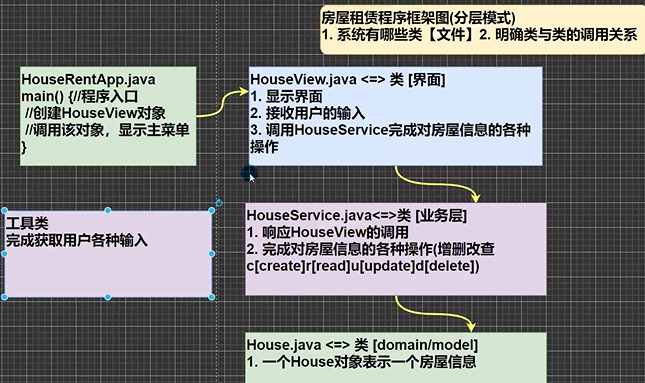
4 完成功能三部曲：明确功能、思路分析、代码实现

**零钱通项目：**

# 第九章 房屋出租系统

分层模式

增删改查（crud）



静态方法：方法是static(静态)时，为静态方法（类方法），可直接通过类名调用

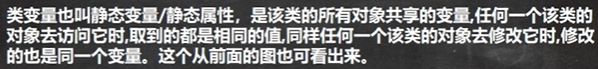
# 第二阶段 面向对象高级

**目录**



**类变量（静态变量）**

概念：



定义方法：



调用方法：

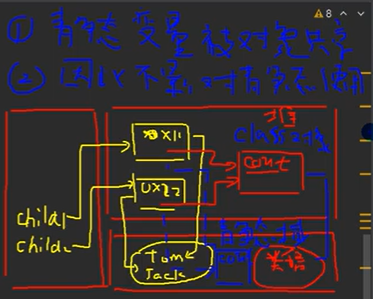


特点：

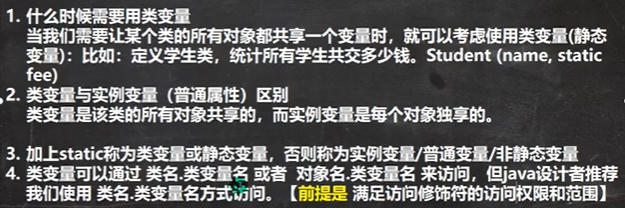
1 被同一个类的所有实例共享

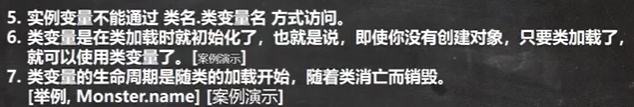
2 在类加载的时候就生成

内存调用机制：



使用细节：





问题1：类何时加载到内存中

**类方法（静态方法）**

定义方法：



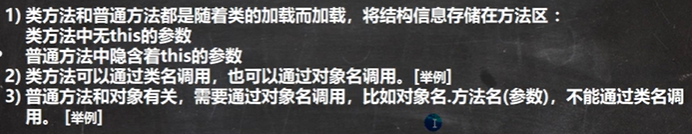
调用：

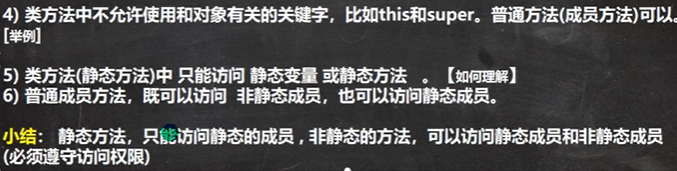


使用场景（当作工具使用）：



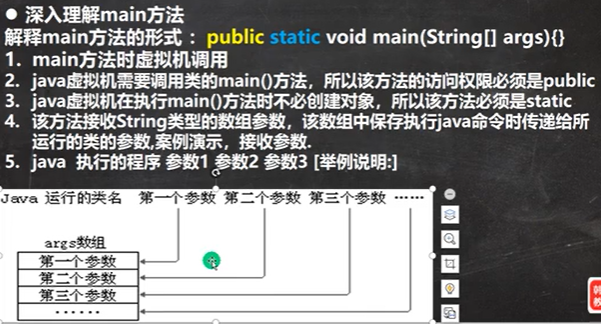
使用细节：



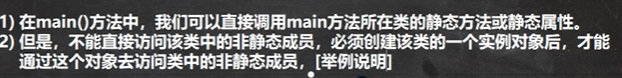


**Main方法**

语法：

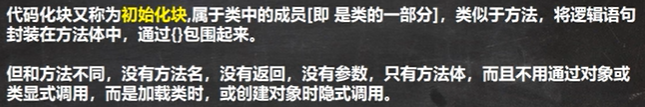


注意：



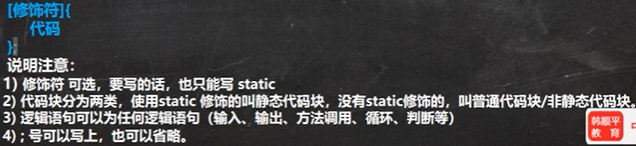
**代码块（初始化块，可理解为对构造器的补充说明）**

概念：

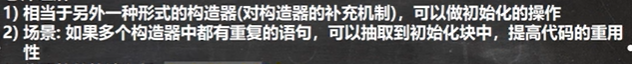


代码块调用顺序优先于构造器

基本语法：



好处（应用场景）：



**细节**

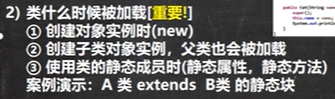
1 静态代码块与普通代码块加载顺序的区别：



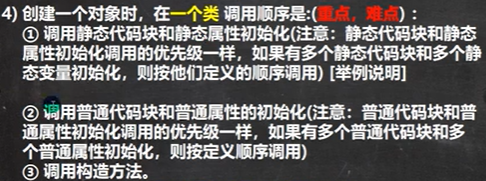


普通代码块可视为构造器的补充

2 类何时被加载：



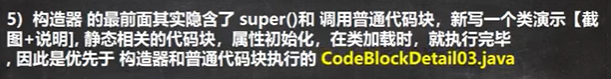
3 创建一个对象时，在一个类的调用（不是加载）顺序：



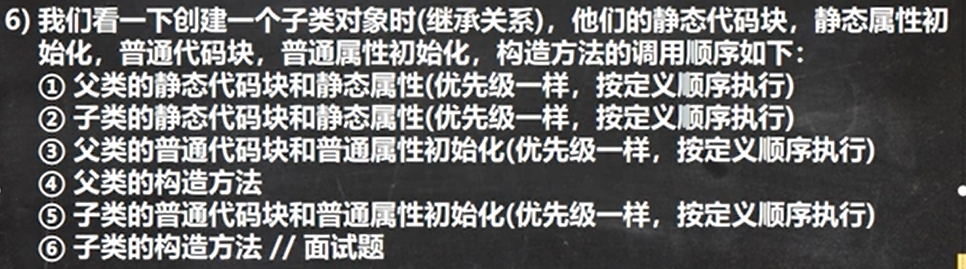
问题2：方法不会初始化吗

（方法需被显式调用）

5 super()、普通代码块、构造器执行顺序



6 创建子类对象时，静态代码块、静态属性初始化、普通代码块、普通属性初始化、构造方法调用顺序：（先进行类加载，在进行对象初始化）

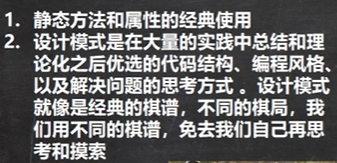


7 代码块调用属性、方法



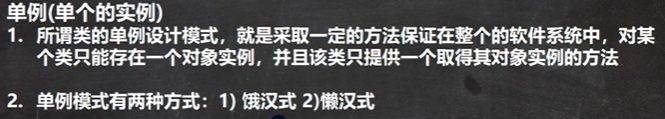
**设计模式（套路，公认23种设计模式）**

概念：



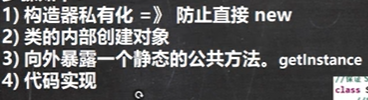
**单例设计模式（单个的实例）（静态方法和属性经典使用场景）**

概念：



**饿汉式（对象在类加载时即创建，可能还没用到，造成资源浪费）：**

步骤：

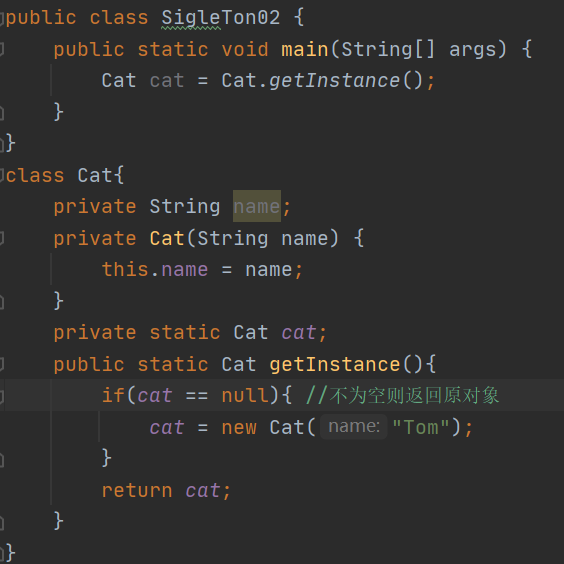




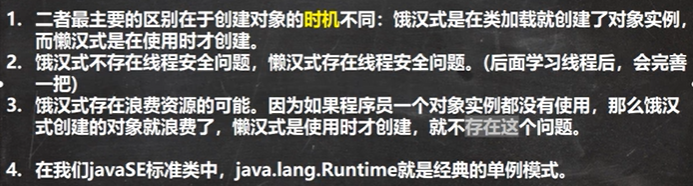
**懒汉式：**

步骤：





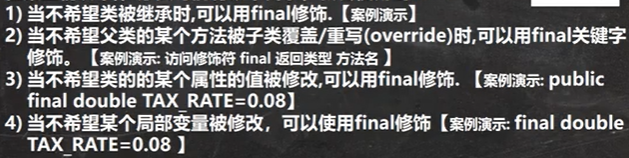
饿汉式与懒汉式区别：



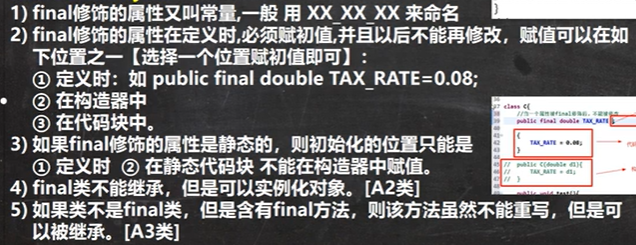
**Final关键字**

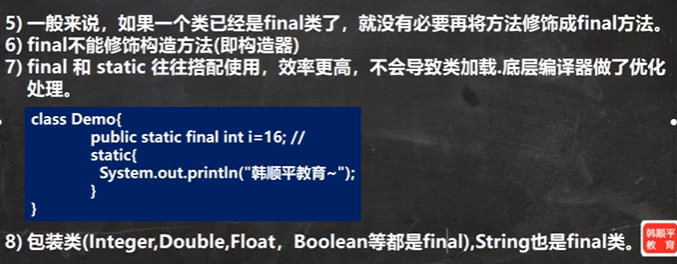
作用：修饰类、属性、方法、局部变量（一般大写字母）

使用场景：



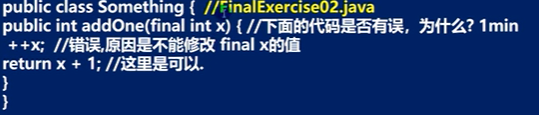
使用细节：





问题：为什么static+final修饰的变量引用时不引起类加载？？

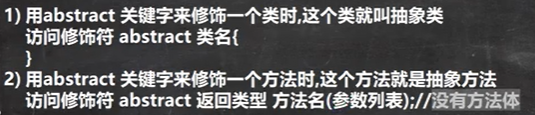
传入的形参可以final修饰并且不赋初值（在实例对象调用方法的时候赋值）：



**抽象类（abstract，本质还是类）：**

概念：用abstract关键字修饰的类

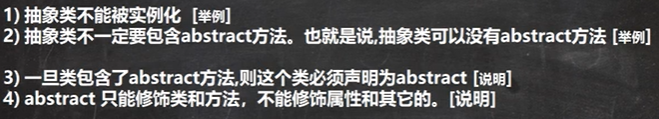
抽象方法：用abstract关键字修饰的方法（没有方法体）

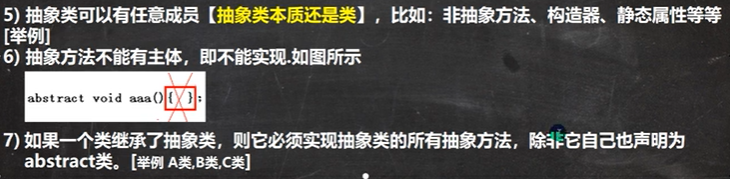


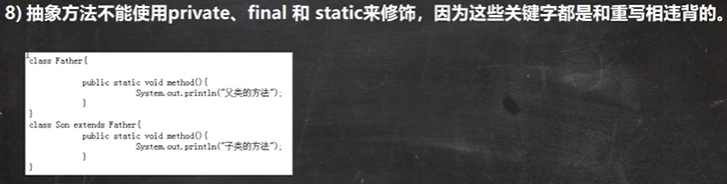
抽象方法的实现：



使用细节：





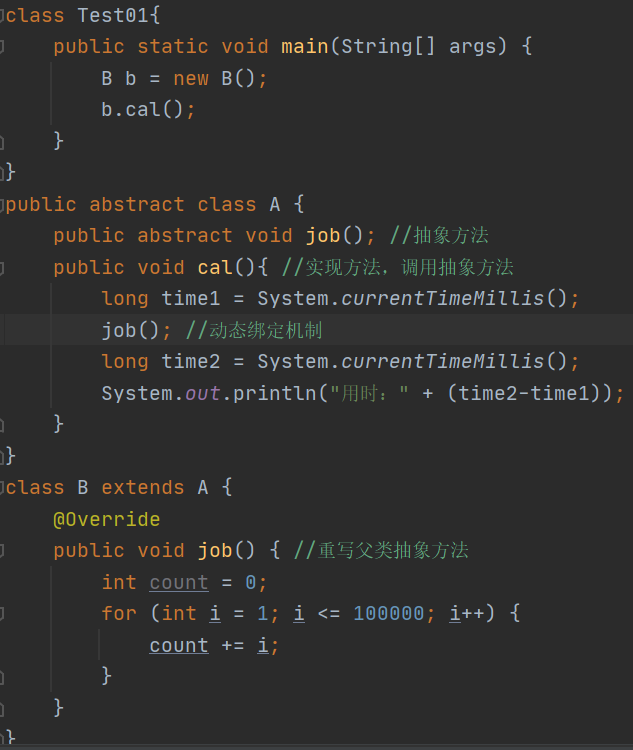


问题：静态方法不可以被重写

**模板设计模式（抽象类最佳应用场景）**

**概念：父类中普通方法调用抽象方法，子类只需重写抽象方法**

返回当前时间(毫秒数)：long time = System.curretTimeMillis()



**接口（抽象类究极进化体，包含了一些默认的规则（public、abstract））**

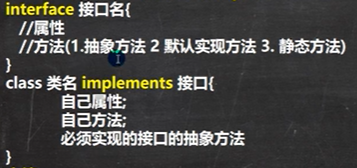
**（对java单继承的一种补充）**

**（继承解决复用性，接口解决拓展性）**

概念（制订规范）：

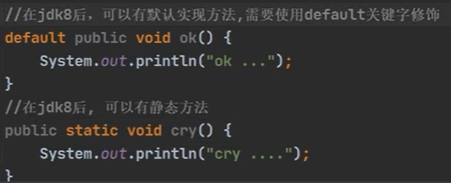


语法：

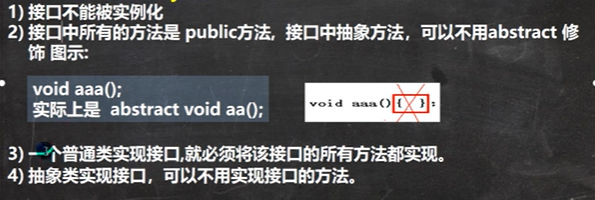


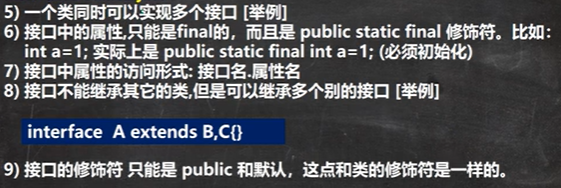
接口中的方法（抽象方法可省略abstract关键字）：





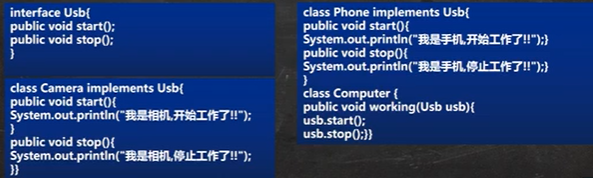
接口细节：





实现类重写接口抽象方法快捷键：**alt+enter**

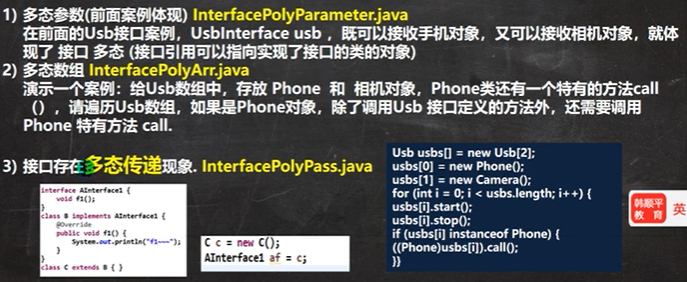
案例：



**接口与继承的关系：**



**接口的多态**



调用接口属性：接口名.方法名；调用父类属性：super.属性名