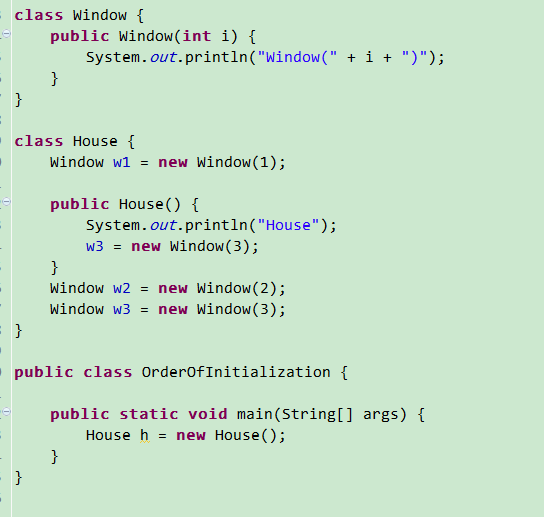
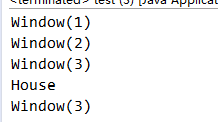
1. 总结：含有继承、静态变量、静态代码块、成员变量、构造代码块、构造函数的初始化顺序：
   1. **从父类—>到子类(连锁执行)：静态变量、静态代码块(即先从父类到子类依次执行其静态成分，仅执行一次)**
   2. **父类：成员变量、构造代码块🡪构造器(即先执行父类的成分)**
   3. **子类：成员变量、构造代码快🡪构造器(后执行子类的成分)**

**Note：静态变量、静态代码块；成员变量、构造代码块。这两组内部的执行顺序与声明顺序一致，即谁先写，谁先执行**

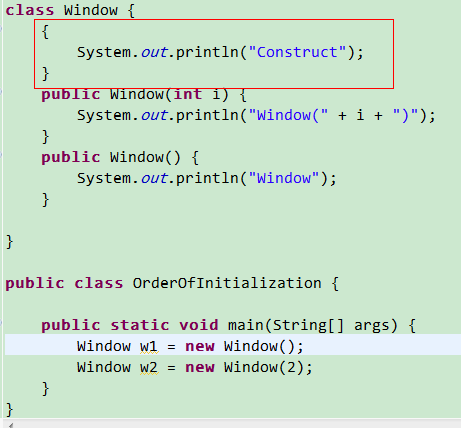
1. 构造器初始化顺序(成员变量自动初始化，无法阻止自动初始化的开始，它将在构造器被调用之前发生，成员变量定义的先后顺序决定了初始化的顺序)。



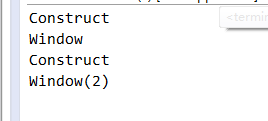
输出：



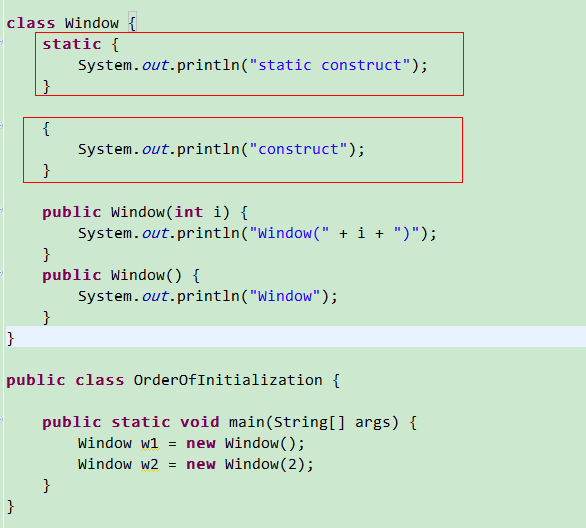
1. 构造代码块(保证无论调用哪个构造器，某些操作都会发生)



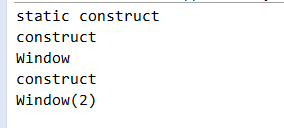
输出：



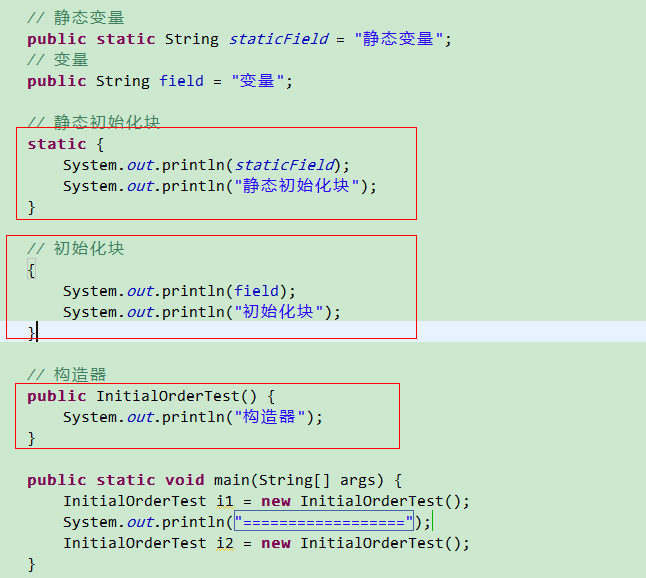
1. 静态语句块(当首次生成这个类的一个对象的时候，或者首次访问属于该类的静态数据成员的时候调用一次，仅调用一次)



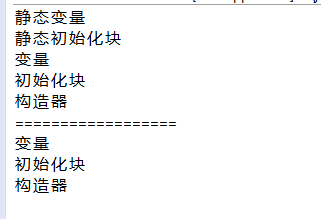
输出



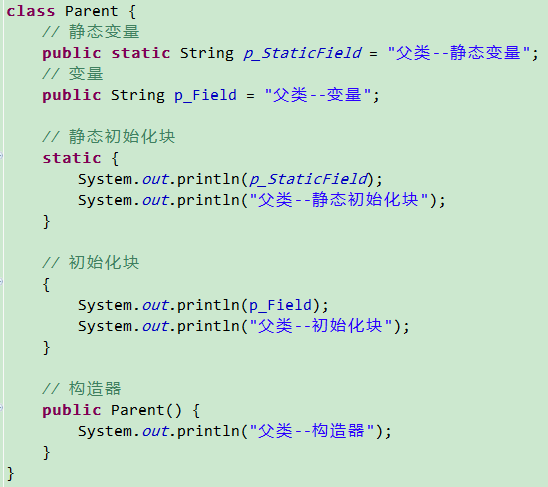
1. 对于静态变量、静态初始化块、变量、初始化块、构造器，它们的初始化顺序依次是（静态变量、静态初始化块）>（变量、初始化块）>构造器

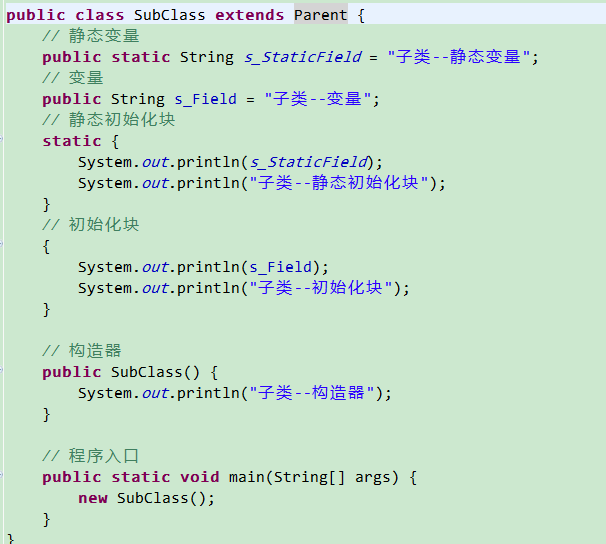


输出：

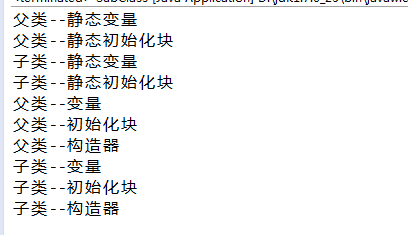


1. 继承情况下的初始化顺序(并不是父类完全初始化完毕后才进行子类的初始化，实际上子类的静态变量和静态初始化块的初始化是在父类的变量、初始化块和构造器初始化之前就完成了)

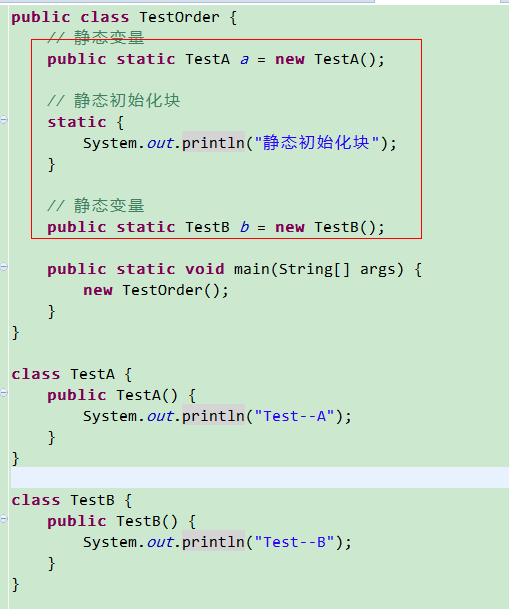




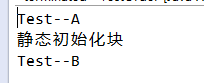
输出：

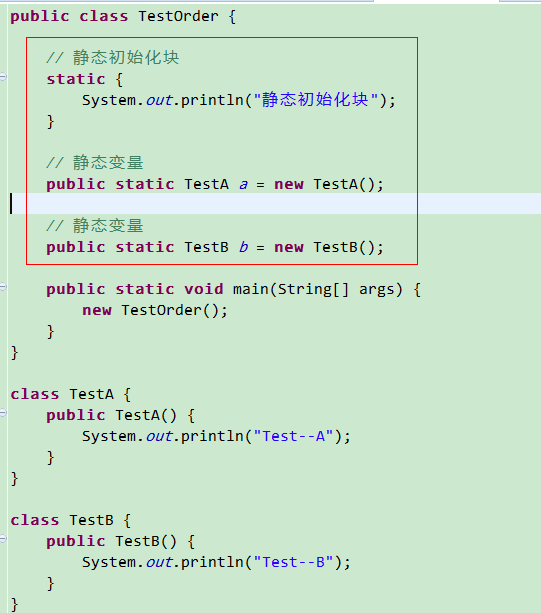


1. 对于静态变量和静态初始化块之间、变量和初始化块之间的先后顺序又是怎样呢？是否静态变量总是先于静态初始化块，变量总是先于初始化块就被初始化了呢？实际上这取决于它们在类中出现的先后顺序

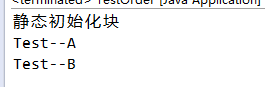


输出：

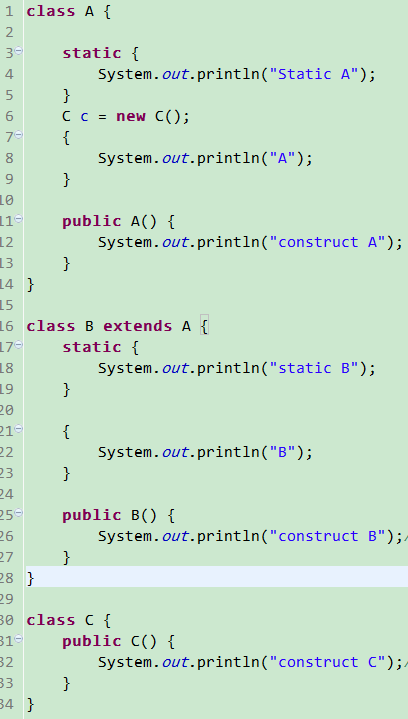




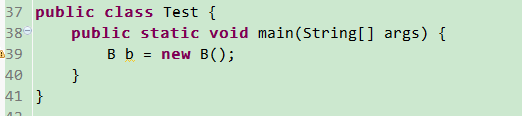
输出：



1. 复合例子



测试



输出：

