对于Java的初学者而言，很多时候我们只知道如何写一些基础的代码，但是却不知道为什么，对于一些概念而言，更是停留在一个很模糊的概念上，其实这样是不好的一种学习习惯。所以对于不太清楚的一些概念，自己多敲一敲，或许能更好的掌握。

今天要说的问题是，Java中代码的执行顺序，这里先定义几个类别：

* 构造方法（如：public className(){……}）
* 静态代码块（如：static {……}）
* 类的属性变量（如：ClassA a = new ClassA()）

这些它们在类被初始化时执行的顺序时怎么样的呢？

package test;

public class ClassOrderTest {

static class A{

C c =new C();

public A(){

System.out.println("A's Constructor");

}

static {

System.out.println("A's static ");

}

}

static class B extends A{

public B(){

System.out.println("B's Constructor");

}

D d = new D();

C c =new C();

static {

System.out.println("B's static ");

}

}

static class C {

public C(){

System.out.println("C's Constructor");

}

static {

System.out.println("C's static");

}

}

static class D{

public D(){

System.out.println("D's Constructor");

}

static {

System.out.println("D's static");

}

}

public static void main(String args[]){

System.out.println("start");

B b = new B();

}

}

上面的代码中，因为几个类都要在Main方法中运行，因此就声明为static类型了，这个大家都知道的吧，不声明为static时，编译器会报错：No enclosing instance of type ClassName is accessible.   
主要使用了四个静态内部类。代码比较简答，就不作过多的说明了。

运行结果

start

A's static

B's static

C's static

C's Constructor

A's Constructor

D's static

D's Constructor

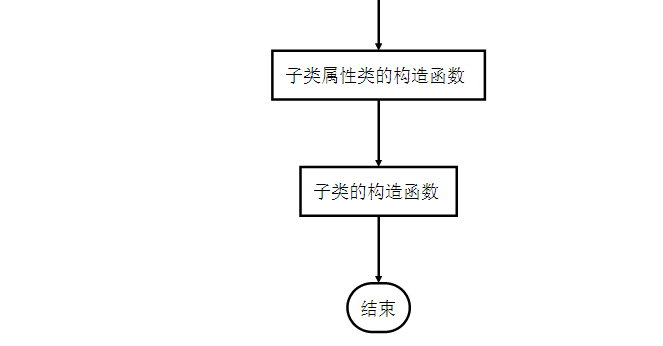
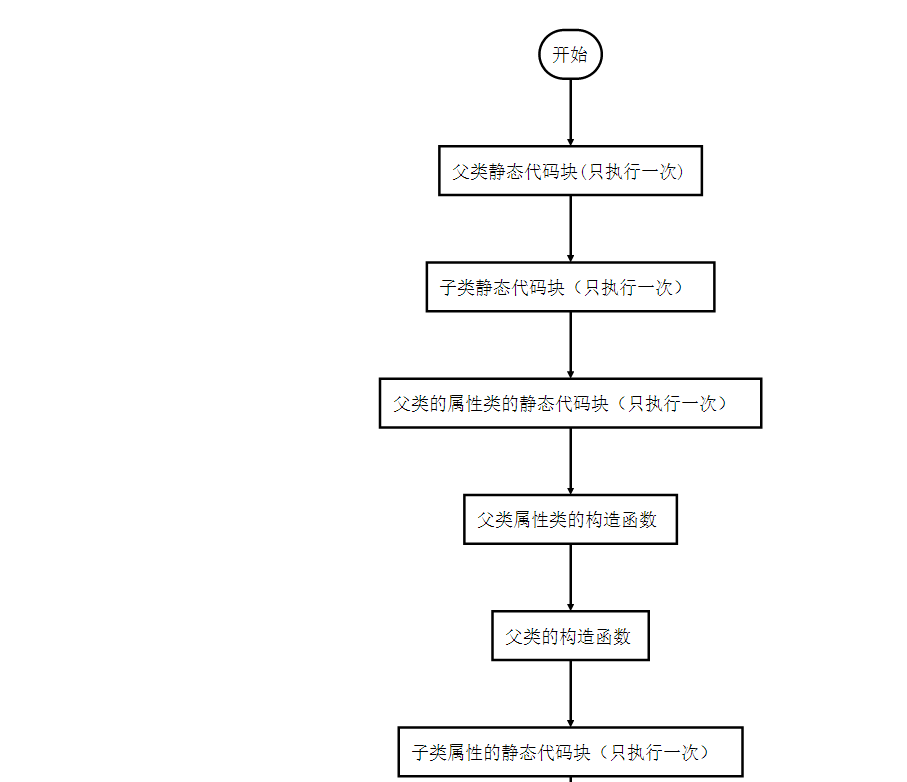
C's Constructor

B's Constructor

结果分析

所以可以得出结论：

开始父类静态代码块(只执行一次)子类静态代码块（只执行一次）父类的属性类的静态代码块（只执行一次）父类属性类的构造函数父类的构造函数子类属性的静态代码块（只执行一次）子类属性类的构造函数子类的构造函数结束



我们看main函数中的语句：

B b = new B();

首先，Java是先初始化类B，初识化的过程中先初始化B的父类，再初始化B本身，因为静态代码块是在类

初始化的时候加载，而类只初识化一次。   
初始化完B的时候：也就是执行B b执行完成。   
程序打印出：

start

A's static //父类初始化

B's static //子类初始化

接下来，执行new B();   
这个过程程序会先走到B的构造方法，如果B存在父类，则向上到父类的构造方法，也就是A的构造方法，此时会对A的属性类进行初始化，即

C c=new C();

就会对C进行相应的初始化、实例化的一些过程，如果只是类的属性类的声明，如：

C c;

则不对类C进行初始化，更不会实例化。   
因此按文章的代码，程序会打印出：

C's static //初识化过程

C's Constructor //实例化过程

完成之后，就开始执行A的构造方法，程序会打印出：

A's Constructor

父类完成之后，就开始对子类的属性初始化，以及属性构造方法调用，最后完成子类的实例化。   
依此打印出：

D's static //子类属性类初始化

D's Constructor//子类属性类实例化

C's Constructor//子类属性类实例化

B's Constructor//子类实例化

*ps：如果将A中的C c=new C();更改为static {C c;}系统也会先执行static代码块，只是，C c只对类C做了声明，因此这里也不会对C做初始化工作*