**·static关键字**

声明static是一个关键字，这个关键字组要用来生成属性和方法。

·static定义属性

在一个类中实际上所有的内容都交由各自的堆内存保存； 假设当我们已经产生了500w个对象 ，而我们想要修改其中所有对象的一个属性时，那么此时你面对的将是一场灾难！！因为对象实在是太多了！！那么此时最好的方法就是将该属性 设置为 （公共属性）如:static String country = "中华人民共和国";当你再次修改时，只需要修改一个对象的country那么所有的country值都会改变。

但时对于static属性的访问需要注意的一点是由于其本身是一个公共的属性，所以理论上虽然是可以通过对象进行访问，但是最好的方法还是通过 类名 进行访问: Person.country = "中华人民共和国";

\*static属性可以通过类名进行调用

eg: Person person = new person();

Person.country = "";

\*同时static属性虽然在类中但是，其并不受类实例化对象的控制，也就是说static属性可以在实例化类之前使用;

eg: //实例化类之前就可以使用

Person.country = "";

Person person = new person();

static也可以进行方法定义,其的特点是可以直接由 类名称在实例化之前调用;

eg:

public static void setCountry(String c){

country = c;

}

注意:

·static 方法只允许调用 static属性或 static方法；

·非static方法允许调用 static属性或 static方法；

原因:所有的static定义的方法和属性可以在实例化之前使用，而所有的非static方法和属性必须实例化之后使用。

static定义的方法和属性不是编写代码之初所需要考虑的，只有在需要回避实例化对象和使用公共属性的情况之下才会考虑使用static定义方法或属性。

**·代码块**

代码块分为以下三类:

·普通代码块

·构造代码块

·静态代码块

普通代码块用 {} 包裹，它主要用来进行一个方法中的分界，也就是说 使用 {} 会将代码进行分界使里面的变量等与外界进行分隔。防止相同的变量名称所带来的互相影响

构造块: 定义在一个类之中，构造块会优先于普通方法执行，并且每一次实例化都会调用构造块中的方法

eg:

clas Person{

public Person(){

System.out.println("【构造方法】Person类的构造方法被执行了");

}

{

System.out.println("【构造块】Person类构造块被执行了");

}

}

静态代码块: 静态代码块主要是指使用static关键字定义的代码块，主要考虑两种情况:①主类中定义静态代码块

②非主类定义静态块

eg:

clas Person{

public Person(){

System.out.println("【构造方法】Person类的构造方法被执行了");

}

static{

System.out.println("【非主类静态代码块】");

}

{

System.out.println("【构造块】Person类构造块被执行了");

}

}

此时可以发现静态代码块会优先于构造块执行，并且不管实例化多少次，静态代码块都只会

执行一次静态代码块的主要目的是为类中的静态属性进行初始化;（如登录数据库的密码 驱动等）

在主类中定义:

public class JavaDemo{

static{

System.out.println("主类中的静态代码块");

}

public static void main (String args[] ){

System.out.println("---主类123456---");

}

}

通过执行可知: 静态代码块优先于主类方法执行