·**代码块**

1. 代码块分为以下三类:

·普通代码块

·构造代码块

·静态代码块

1. 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(就是方法体)

1. 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");  }  } |

1. 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. 非主类定义静态代码块

|  |
| --- |
| 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 |