静态代码块

构造方法用于对象的初始化。

静态初始化块（方法中不能存在），用于**类的初始化操作**。在静态初始化块中不能直接访问非static成员。

静态代码块在类被加载时运行，且只自动运行一次，要优于构造函数。一般用于加载配置资源。

构造（实例）代码块--在java类中使用{}声明的代码块。

创建对象时被调用，一次创建调用一次。插入到构造函数的最前面，不是在构造函数之前运行,而是依托于构造函数。特性：1.在每个构造函数中都运行。2.在构造函数中它会首先运行。

构造器代码块的作用是提取侯构造器中的相同部分，减少重复代码。

```
    public class CodeBlock {
    {
        System.out.println("构造代码块");
    }

    public CodeBlock(){
        System.out.println("无参构造函数");
    }
    public CodeBlock(String str){
        System.out.println("有参构造函数");
    }
}
```

反编译生成的class文件：

```
import java.io.PrintStream;

public class CodeBlock
{

        public CodeBlock()
        {
                System.out.println("构造代码块");
                System.out.println("无参构造函数");
        }

        public CodeBlock(String str)
        {
                System.out.println("构造代码块");
                System.out.println("有参构造函数");
        }
}
```

构造函数

构造函数的功能主要用于在类的对象创建时（实例化）定义初始化的状态。它没有返回值，也不能用void来修饰。

必须通过new运算符在创建对象时才会自动调用。如果没有显式声明，Java编译器会提供一个隐式默认的无参构造函数。

普通代码块

普通代码块和构造代码块的区别是，构造代码块是在类中定义的，而普通代码块是在方法体中定义的。

执行顺序：静态代码块>（main方法>）构造代码块>构造函数>普通代码块

加载类-->构造器生成对象（先初始化成员变量，再构造函数）(父类再子类)-->实例化对象

实例变量的初始化语句经过编译器处理后，都会合并到构造器中去。其中定义变量语句转换得到的赋值语句、初始化块中的语句转化得到的赋值语句，**总是位于构造器的所有语句之前。**

经典实例：

```
public class CodeBlock {
    static{
        System.out.println("静态代码块");
    }
    {
        System.out.println("构造代码块");
    }
    public CodeBlock(){
        System.out.println("无参构造函数");
    }

    public void sayHello(){
        {
            System.out.println("普通代码块");
        }
    }

    public static void main(String[] args) {
        System.out.println("执行了main方法");

        new CodeBlock().sayHello();;
        System.out.println("---------------");
        new CodeBlock().sayHello();;
    }
}
```

反编译生成的class文件：

```
public class CodeBlock
{

        public CodeBlock()
        {
                System.out.println("构造代码块");
                System.out.println("无参构造函数");
        }

        public void sayHello()
        {
                System.out.println("普通代码块");
        }

        public static void main(String args[])
        {
                System.out.println("执行了main方法");
                (new CodeBlock()).sayHello();
                System.out.println("---------------");
                (new CodeBlock()).sayHello();
        }

        static
        {
                System.out.println("静态代码块");
        }
}
```

执行结果：

```
静态代码块
执行了main方法
构造代码块
无参构造函数
普通代码块
---------------
构造代码块
无参构造函数
普通代码块
```

对象的初始化实例化顺序：

父类静态代码块
子类静态代码块
父类构造代码块
父类构造函数
子类构造代码块
子类构造函数
- - - - - - - - - - - -
父类构造代码块
父类构造函数
子类构造代码块
子类构造函数