



对象初始化的顺序

写 `new 对象()` 的时候, Java 在背后从类的静态部分到对象的成员变量, 全部初始化了一遍

- 新建对象的唯一途径
 - 在堆上分配空间
 - 执行必要的初始化工作
 - 执行构造器函数

▼ 初始化块 Initialiser Block

- Initializer Block 是 Java 中一种用于在对象构造前进行“通用初始化”的代码块, 它有两种类型:

▼ 普通初始化块 (对象初始化块)

```
{  
    // 初始化代码  
    System.out.println("对象初始化块");  
}
```

- 不写在任何方法里, 也不写在构造器里
- 每次 **创建对象 (new)** 时都会执行一次
- 它会在 **构造方法之前执行**
- 多个构造器**共用这段初始化逻辑**

```
public class Cat {  
    int age;        // 成员变量  
  
    {
```

```
        System.out.println("对象初始化块执行");
        age = 1;
    }

    Cat() {
        System.out.println("构造器执行");
    }

    public static void main(String[] args) {
        new Cat();
    }
}
```

输出：

```
对象初始化块执行
构造器执行
```

- 💡 它的作用 ➡ 如果你有一些**所有构造器都需要执行的代码**，可以写在实例初始化块中，避免复制粘贴

▼ 静态初始化块（Static Initialiser Block）

```
static {
    // 只执行一次
    System.out.println("静态初始化块");
}
```

- 用 `static` 修饰
- 属于 **类本身**
- **只执行一次**：在类**第一次加载**到内存时执行
- 通常用于初始化静态变量、加载资源

```
public class Cat {
    static {
        System.out.println("静态初始化块执行");
    }

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

输出：



```
静态初始化块执行
main方法执行
```

特性	普通初始化块	静态初始化块
关键字	无	<code>static</code>
属于谁	对象	类
执行时机	每次 new 时	类加载时（一次）
是否可以访问实例变量	可以	✗ 不可以（没有 this）


初始化块 = 不在构造器里的初始化代码

用来简化逻辑、避免重复、控制执行顺序

▼ 初始化的详细顺序

-  必要的初始化工作
 - 就是类加载进内存后，要做的一系列准备，比如 变量分配内存、初始化默认值
-  静态成员变量的初始化（只做一次）
 - 包括 `static int x = 10;` 这种静态变量的初始化
 - 是属于类的，不属于对象

- **类加载时就会初始化**

-  **静态初始化块** `static {}` （也是只执行一次）


```
static {  
    System.out.println("静态初始化块执行了");  
}
```

- 它会在**类被加载时执行**
- 用于复杂的静态初始化逻辑

-  **成员变量**的初始化


```
int age = 1;
```

- 这是给对象属性指定初始值
- **每次 new 一个对象，都会初始化这些成员变量**

-  **实例初始化块** `{}`

```
{  
    System.out.println("普通初始化块执行");  
}
```

- 没有 `static`，是对象专用的
- **每次创建对象都会执行**
- **会比构造器先执行**

-  **构造器的执行 Constructor**

- **构造器是最后一步**，它可以使用前面初始化好的变量，再做一次**最终定制化的**初始化

▼ 总结

当你写：

```
new Cat();
```

Java 执行顺序是：

1. 加载类 → 静态成员初始化 + 静态初始化块（只执行一次）
2. 分配内存 → 非静态成员默认赋值
3. 成员变量显示初始化（比如 `int x = 10;`）
4. 实例初始化块（每次 new 时都执行）
5. **构造方法**（最后执行）

▼ 示例

```
public class Demo {  
    static int x = initStaticVar();  
    static {  
        System.out.println("静态初始化块");  
    }  
  
    int y = initMemberVar();  
  
    {  
        System.out.println("实例初始化块");  
    }  
  
    Demo() {  
        System.out.println("构造方法");  
    }  
  
    static int initStaticVar() {  
        System.out.println("初始化静态变量");  
        return 100;  
    }  
  
    int initMemberVar() {
```

```

        System.out.println("初始化成员变量");
        return 200;
    }

    public static void main(String[] args) {
        System.out.println("第一次创建对象：");
        new Demo();

        System.out.println("\n第二次创建对象：");
        new Demo();
    }
}

```



输出结果：

```

初始化静态变量
静态初始化块
第一次创建对象：
初始化成员变量
实例初始化块
构造方法

第二次创建对象：
初始化成员变量
实例初始化块
构造方法

```

类只初始化一次（静态部分），对象每 new 一次都会走成员变量 → 初始化块 → 构造器的流程