



# 实战Java虚拟机

## JVM故障诊断与性能优化

葛一鸣 | 著

通过200余示例详解Java虚拟机的各种参数配置、故障排查、性能监控及优化技术全面，通俗易懂



中国工信出版集团



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>



4.2.4	标记压缩法 (Mark-Compact)	66
4.2.5	分代算法 (Generational Collecting)	67
4.2.6	分区算法 (Region)	68
4.3	谁才是真正的垃圾: 判断可触及性	69
4.3.1	对象的复活	69
4.3.2	引用和可触及性的强度	71
4.3.3	软引用——可被回收的引用	72
4.3.4	弱引用——发现即回收	76
4.3.5	虚引用——对象回收跟踪	77
4.4	垃圾回收时的停顿现象: Stop-The-World 案例实战	79
4.5	小结	83
<b>第 5 章</b>	<b>垃圾收集器和内存分配</b>	<b>84</b>
5.1	一心一意一件事: 串行回收器	85
5.1.1	新生代串行回收器	85
5.1.2	老年代串行回收器	86
5.2	人多力量大: 并行回收器	86
5.2.1	新生代 ParNew 回收器	87
5.2.2	新生代 ParallelGC 回收器	88
5.2.3	老年代 ParallelOldGC 回收器	89
5.3	一心多用都不落下: CMS 回收器	90
5.3.1	CMS 主要工作步骤	90
5.3.2	CMS 主要的设置参数	91
5.3.3	CMS 的日志分析	92
5.3.4	有关 Class 的回收	94
5.4	未来我做主: G1 回收器	95
5.4.1	G1 的内存划分和主要收集过程	95
5.4.2	G1 的新生代 GC	96
5.4.3	G1 的并发标记周期	97
5.4.4	混合回收	100
5.4.5	必要时的 Full GC	102
5.4.6	G1 日志	102

# 1

## 第 1 章

### 初探 Java 虚拟机

---

什么是 Java 虚拟机？什么是 Java 语言？两者又有何关系？作为本书开篇之章，本章将主要介绍有关 Java 虚拟机的基本概念、发展历史和实现概要。其中，将重点介绍支撑 Java 世界的两份重要规范——Java 语言规范和 Java 虚拟机规范，帮助读者更好地理解 Java 生态圈。

本章涉及的主要知识点有：

- 读懂 Java 的发展历史。
- 学习 Java 虚拟机的概念和种类。
- 接触 Java 语言规范。
- 了解 Java 虚拟机规范。
- 掌握单步调试 Java 虚拟机的方法。



即在一个 if 语句中，表示条件的表达式必须用小括号标示，同时在右小括号后，书写语句块，表示执行内容。而对于 Expression 和 Statement 的具体定义，在语言规范中也有十分详细的描述，这里就不一一展开了，有兴趣的读者可以参考 Java 语言规范，JDK 1.7 版第 14 章的内容“Blocks and Statements”。

### 1.3.3 数据类型的定义

Java 语言规范中还定义了 Java 的数据类型。根据 Java 1.7 的规范，Java 的数据类型分为原始数据类型和引用数据类型。原始数据类型又分为数字型和布尔型。数字型又有 byte、short、int、long、char、float、double。注意，在这里 char 被定义为整数型，并且在规范中明确定义：byte、short、int、long 分别是 8 位、16 位、32 位、64 位有符号整数，而 char 为 16 位无符号整数，表示 UTF-16 的字符。布尔型只有两种取值：true 和 false。而对于 float 和 double，规范中规定，它们是满足 IEEE 754 的 32 位浮点数和 64 位浮点数。

**注意：**在 Java 语言中，char 占 2 字节，而不是 C 语言中的 1 字节。从这点上看，Java 的国际化是在语言底层就提供了强有力的支持。

此外，规范还定义了各类数字的取值范围、初始值，以及能够支持的各种操作。以整数为例，比较运算、数值运算、位运算、自增自减运算等都在规范中有描述。

除了基本数据类型外，引用数据类型也是 Java 重要的组成部分，引用数据类型分为 3 种：类或接口、泛型类型以及数组类型。

**提醒：**引用类型和原始类型在 Java 的处理中是截然不同的，尤其对于它们的“相等”操作。

**【示例 1-1】**在 Java 语言规范中，有一个简短的示例，说明了引用类型和原始类型的区别：

```
class Value { int val; }
class Test {
    public static void main(String[] args) {
        int i1 = 3;
        int i2 = i1;
        i2 = 4;
        System.out.print("i1==" + i1);
        System.out.println(" but i2==" + i2);
        Value v1 = new Value();
        v1.val = 5;
        Value v2 = v1;
```



```

        v2.val = 6;
        System.out.print("v1.val==" + v1.val);
        System.out.println(" and v2.val==" + v2.val);
    }
}

```

上述程序将输出:

```

i1==3 but i2==4
v1.val==6 and v2.val==6

```

从上述输出可以看出, 对于原始数据类型 `int`, `i1` 和 `i2` 表示不同的变量, 两者毫无关系, 但是对于 `v1` 和 `v2`, 它们都指向唯一一个由 `new` 关键字创建的 `Value` 对象。

由于本书并非讲解 Java 语言, 因此对于这部分内容点到即止, 有兴趣的读者可以参考 Java 语言规范的第 4 章 “Types, Values, and Variables”。

### 1.3.4 Java 语言规范总结

除上述基本内容外, Java 语言规范还定义了各种不同类型间的转换规则、方法的可见性定义、有关接口的使用、注释等。

总之, Java 语言规范完整定义和描述了 Java 语言的所有特性, 因为 Java 语言本身不属于本书的讨论重点, 故在此只做简要介绍。

## 1.4 一切听我的: Java 虚拟机规范

虽然 Java 语言和 Java 虚拟机有着密切的联系, 但两者是完全不同的内容。Java 虚拟机是一台执行 Java 字节码的虚拟计算机, 它拥有独立的运行机制, 其运行的 Java 字节码也未必由 Java 语言编译而成, 像 Groovy、Scala 等语言生成的 Java 字节码也可以由 Java 虚拟机执行。立足于 Java 虚拟机, 可以产生各种各样的跨平台语言。除了语言特性各不相同外, 它们可以共享 Java 虚拟机带来的跨平台性、优秀的垃圾回收器, 以及可靠的即时编译器。

因此, 与 Java 语言不同, Java 虚拟机是一个高效的、性能优异的、商用级别的软件运行和开发平台, 而这也是本书讨论的重点。

Java 虚拟机规范的主要内容大概有以下几个部分:

- 定义了虚拟机的内部结构 (将在第 2 章中详细介绍)。

- 定义了虚拟机执行的字节码类型和功能（将在第 11 章中详细介绍）。
- 定义了 Class 文件的结构（将在第 9 章中详细介绍）。
- 定义了类的装载、连接和初始化（将在第 10 章中详细介绍）。

以 Java 1.7 为例，读者可以在 <http://docs.oracle.com/javase/specs/jvms/se7/html/> 浏览虚拟机规范全文。这份规范可以说是开发 Java 虚拟机的指导性文件，如果要实现自定义的 Java 虚拟机，则需要参考和熟悉这份规范，同时这份规范对于了解现存的流行 Java 虚拟机（如 Hotspot、IBM J9 等），也有十分重要的意义。

## 1.5 数字编码就是计算机世界的水和电

数字是计算机内最直接、最基础的表现类型。了解数字在计算机内的表示，对于了解整个计算机系统具有相当重要的作用，数字也是专业计算机从业人员的基本功之一。本节将主要介绍整数以及浮点数在 Java 虚拟机中的支持情况。

### 1.5.1 整数在 Java 虚拟机中的表示

在 Java 虚拟机中，整数有 byte、short、int、long 四种，分别表示 8 位、16 位、32 位、64 位有符号整数。整数在计算机中使用补码表示，在 Java 虚拟机中也不例外。在学习补码之前，必须先理解原码和反码。

所谓原码，就是符号位加上数字的二进制表示。以 int 为例，第 1 位表示符号位（正数或者负数），其余 31 位表示该数字的二进制值。

10 的原码为：00000000 00000000 00000000 00001010

-10 的原码为：10000000 00000000 00000000 00001010

对于原码来说，绝对值相同的正数和负数只有符号位不同。

反码就是在原码的基础上，符号位不变，其余位取反，以 -10 为例，其反码为：

11111111 11111111 11111111 11110101

负数的补码就是反码加 1，整数的补码就是原码本身。

因此，10 的补码为：

00000000 00000000 00000000 00001010