- 前言
- 2.1 JVM概述
- 2.2 JVM内存结构篇
- 2.2 JVM内存结构
- 2.3 堆内存分配
- 2.4 堆内存调整
- 2.5 堆内存实战（实战）
- 2.6 JVM GC
- 2.7 垃圾收集
- 2.8 常用工具类
- 2.9 JVM调优
- 2.10 JVM实战
- 2.11 JVM实战
- 2.12 JVM最佳实践
- 2.13 JVM实战
- 2.14 JVM
- JVM内存模型
- 大白话带你认识JVM
- JVM面试必问的Java内存区域
- Java内存泄漏排查
- Java堆内存结构
- Java为字节码
- 性能优化的黄金指标和工具
- 字节码指令详解
- 第1条指令解析
- 第2条指令解析
- 第3条指令解析
- 第4条指令解析
- 第5条指令解析
- 字节码指令
- JVM内存模型
- JVM常见面试三问
- JVM常见面试之Arthas篇
- 内存泄漏排查实战
- CPU 100%排查实战实录
- JVM第三轮问题
- 2.2 JVM内存
- 3. MySQL
- 4. Redis
- 5. 分布式锁
- 6. 消息队列
- 7. 搜索引擎
- 8. 学习路线
- 9. 知识图谱
- 9. 联系作者

一文彻底搞懂 Java 类加载机制

二哥 发表于 2023-03-29 阅读量 1.2k 字数 12.9k 约阅读时长 42 分钟

上一个 编辑 JVM 的 Java 代码的时候，你肯定知道，JVM 要将最终编译成的字节码文件加载到内存中才能执行，这个过程也是个很复杂的东西，这个过程也是个很复杂的东西，所以我们来一起来搞懂一下 JVM 的类加载机制吧。

字节码的工作，一定是 .class 文件，它有点的玩意儿的东西是有点，明白大家都是什么东西。

这是我的大家是一个小技巧，可以通过 .txt 后 ls 长度看看字节码文件，为每个类生成的代码。

```
public class Test {
    public static void main(String[] args) {
        System.out.println("二哥来了");
    }
}
```

代码编译成功之后，会生成一个 Test.class 二哥文件，字节码文件是 .class 文件，它的内容是二进制文件，Windows 用户可以直接用 vim 或 notepad++ 直接打开，我们以 vim 直接来看看字节码的内容吧。

```
$ vim -b -c '%!xxd' -c 'set ft=xxd' -c 'w!' -c 'q!' Test.class
00000000: cafe babe 0000 0034 0021 0700 0201 0006 5465 7374  ......4.!......Test
00000010: 6347 6165 6368 6572 0700 2704 0025 5374 7269 6e67  cGacher..'..%String
00000020: 2063 6f6e 7374 2d70 6f6f 6c2d 3030 2e6e 6577 7465  const-pool-00.newte
00000030: 7374 2e6d 6169 6e2e 5465 7374 2e6d 6169 6e2e 6d61  st.main.Test.main.ma
00000040: 6e00 1b00 1a00 1b00 1b00 1b00 1b00 1b00 1b00 1b00  n...............
00000050: 2b01 1f00 2b00 0000 0000 0000 2000 0001 0000 0000  +...+.......
00000060: 0000 0000 0a01 0a00 0000 0000 0000 0000 0000 0000  ..
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000  ..
```

字节码文件的内容是这样的，看起来不太友好，我们还是分析分析吧。

类加载过程

加载过程是 JVM 将字节码文件从磁盘读入内存，并创建对应的 Class 对象，这个过程是整个类加载机制的起点。

字节码文件加载的内容主要包括：魔数与版本信息、常量池信息、类的基本信息等。

接下来就是验证阶段，这个阶段主要是对字节码文件的合法性进行检查，确保它符合 JVM 规范，防止恶意或损坏的字节码文件被加载。

验证阶段主要包括：文件格式验证、元数据验证、字节码验证、符号引用验证。

```
public class HelloWorldApp {
    public static String name = "二哥";
    public static final String name2 = "二哥来了";
}
```

charms 不是单纯的字符，而 manger 的 “的” 字是 “的” 的简化版本吗？“来了” 一样。

最后是准备阶段，Linux 的 ulimit 命令是最常用的参数之一，它主要是用来限制一个进程可以使用的资源上限，比如：文件句柄数、内存大小、CPU 时间、进程数等。

通过查看 ulimit -a 命令，可以看到当前系统对各项资源的具体限制，例如 open files 表示最大文件句柄数，stack size 表示栈内存大小，cpu time 表示 CPU 时间，max user processes 表示最大进程数。

通过设置 ulimit -n 2048 命令可以临时调整文件句柄数，但这种设置仅在当前会话中有效，重启后会恢复为默认值。

Java 中是单一继承语言，但是实现了多个接口的类，因此在进行继承关系的检查时，需要以实现了接口的类作为基准。

实现了多个接口的类，要在一个内存结构中记录多个接口的信息，这时可以参考 ClassInstanceKlass 的设计，将接口信息以数组的形式存储在内存中，这样可以保证类的结构完整性和访问效率。

这里要注意，接口信息的顺序，也可以理解，简单来个例子，要么是字母，要么是时间，要么是随机，这个顺序是 JVM 内部实现的一部分，通常会按照一定的规则来排列，以保证查找的效率和一致性。

```
public class Test {
    public static void main(String[] args) {
        System.out.println("二哥来了");
    }
}
```

编译成功之后，会生成一个名为 Test.class 的文件，这个文件就是 Class 对象的持久化形式，它包含了类的所有信息，包括：字段信息、方法信息、常量池、属性信息等。

上面这段代码编译成功之后，会生成一个名为 Test.class 的文件，这个文件就是 Class 对象的持久化形式，它包含了类的所有信息，包括：字段信息、方法信息、常量池、属性信息等。

```
public class Test {
    public static void main(String[] args) {
        System.out.println("二哥来了");
    }
}
```

上面这段代码编译成功之后，会生成一个名为 Test.class 的文件，这个文件就是 Class 对象的持久化形式，它包含了类的所有信息，包括：字段信息、方法信息、常量池、属性信息等。

51. Initialization（初始化）

这时候就是最后的阶段了，也是最后的阶段，也是最后的阶段，也是最后的阶段