HenCoder Plus 讲义

深入理解 JVM

字节码结构

> 使用 hexdump 命令可以用来查看我们的 16 进制字节码。

如下图所示，字节码文件不使用分隔符

```
ClassFile {
    u4             magic;
    u2             minor_version;
    u2             major_version;
    u2             constant_pool_count;
    cp_info        constant_pool[constant_pool_count-1];
    u2             access_flags;
    u2             this_class;
    u2             super_class;
    u2             interfaces_count;
    u2             interfaces[interfaces_count];
    u2             fields_count;
    field_info     fields[fields_count];
    u2             methods_count;
    method_info    methods[methods_count];
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

官方链接

反编译字节码

1. 使用 `javap` 反编译可以看到文件基本结构

```
pdog@10 ~/IdeaProjects/jvm/out/production/jvm javap Hello
Compiled from "Hello.java"
public class Hello {
  public Hello();
  public static void main(java.lang.String[]);
}
```

2. 使用 `javap -c` 会得到每个方法的 `Code` 信息

```
✘ pdog@10 ~/IdeaProjects/jvm/out/production/jvm javap -c Hello
Compiled from "Hello.java"
public class Hello {
  public Hello();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2                  // Field java/lang/System.ou
       3: ldc           #3                  // String hello world!
       5: invokevirtual #4                  // Method java/io/PrintStrea
```

3. 使用 `javap -v` 可以看到更详细的内容

```
✘ pdog@10 ~/IdeaProjects/jvm/out/production/jvm javap -v Hello
Classfile /Users/pdog/IdeaProjects/jvm/out/production/jvm/Hello.class
  Last modified Dec 10, 2019; size 519 bytes
  MD5 checksum 94668c1ec8f3d8cc302526911222e6be
  Compiled from "Hello.java"
public class Hello
  minor version: 0
  major version: 52
  flags: ACC_PUBLIC, ACC_SUPER
Constant pool:
   #1 = Methodref          #6.#20         // java/lang/Object."<init>":()V
   #2 = Fieldref           #21.#22        // java/lang/System.out:Ljava/io/Pr
```

4. 在 Code 信息中可以看到我们的操作数栈的大小,本地变量表的大小和传入参数数量

```
public Hello();
  descriptor: ()V
  flags: ACC_PUBLIC
  Code:
    stack=1, locals=1, args_size=1
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return
    LineNumberTable:
      line 1: 0
    LocalVariableTable:
      Start  Length  Slot  Name   Signature
          0       5     0  this   LHello;
```

> 操作数栈和本地变量表的大小的单位为 Slot，`double` 类型和 long 类型会占用 2 个 Slot

> 每个非静态方法的传入参数都会有一个 `this`

5. 虚拟机字节码指令表

   在字节码中每个指令都用一个字节来表示

6. 通过字节码来学习

   1. 内部类

      内部类会持有外部类的原因是在内部类中用一个成员变量记录了外部类对象在 Kotlin 中，如果在内部类中没有使用到外部类，那么不会持有外部类

   2. 泛型

      Java 的泛型擦除并不是将所有泛型信息全部都擦除了，会将类上和方法上声明的泛型信息保 存在字节码中的 Signature 属性中，这也是反射能够获取泛型的原因。但是在方法中的泛型 信息是完全擦除了

   3. synchronized 关键字

      synchronized 会在方法调用前后通过 monitor 来进入和退出锁

JVM 运行时数据区

- 被所有线程共享的
  - 堆
  - 方法区

    - 运行时常量池
- 每个线程私有的
    - 本地方法栈
    - 虚拟机栈
    - 程序计数器
- 虚拟机栈中的栈帧
    - 局部变量表
    - 操作数栈
    - 动态连接
    - 返回地址
- 对象在堆中：
    - 对象头
        - MarkWord
        - 类型指针
        - 数组长度（对象是数组时）
    - 具体数据
    - 对齐填充
- MarkWord 的结构（32位虚拟机）

状态	23	2	4	1	2
无锁	hashcode		age	0	01
偏向锁	thread	时间戳	age	1	01
轻量级锁	指向栈中锁的指针				00
重量级锁	指向重量级锁的指针				10
GC					11

无锁状态中的 hashcode 是懒加载的，一个对象一旦计算过 hashcode 就被不会成为偏向锁。 而一个偏向锁状态的对象，一旦计算过 hashcode (Object 默认的或者是 System 类提供的)，那 么会立即升级到重量级锁。 锁只能升级不能降级，无锁 -> 偏向锁 -> 轻量级锁 -> 重量级锁

垃圾回收

- JVM 堆的结构划分
  - 新生代
    - eden (80%)
    - from (10%)
    - To (10%)
  - 老年代

  对象会在新生代的 eden 区域中创建（大对象会直接进入老年代），第一次 eden 区满了以后进行 minor GC 将存货对象 age + 1，然后放入 from 区域，将 Eden 区域内存清空。

  以后每次 minor GC 都将 eden 和 from 中的存活对象 age + 1 ，然后放入 to 区域，然后将 to 区 域和 from 区域互相调换。

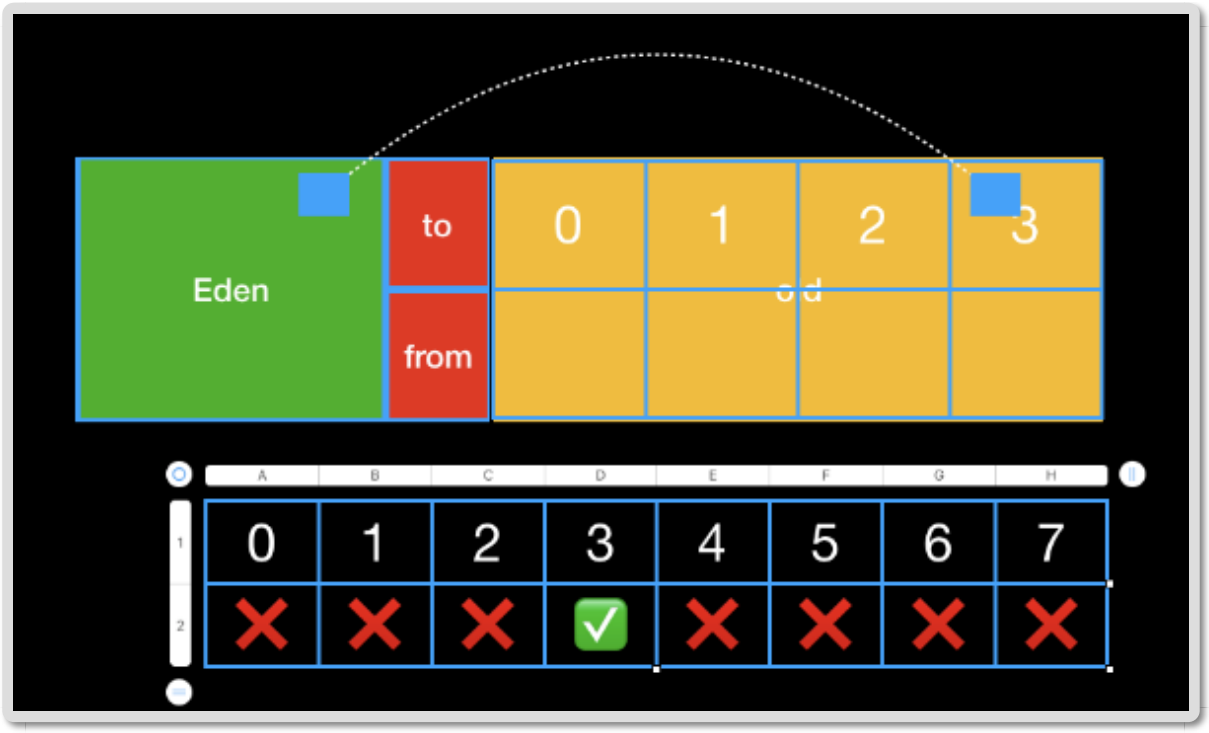
  - 在 age 达到一定值时会移动到老年代
  - 在 minor GC 时，存活对象大于 to 区域，那么会直接进入老年代
- 垃圾回收算法
  - 标记-清除算法 （会产生空闲内存碎片
  - 标记-整理算法（防止产生内存碎片）
  - 复制算法（效率最高，但是内存利用率低）

  > JVM 中新生代使用复制算法，老年代使用标记整理算法

- 关于跨代引用

  为了防止不能确定新生代的对象是否被老年代的对象引用而需要进行 full GC 。

  通过 card table 将老年代分成若干个区域，所以在 minor GC 时只需要对表中记录的老年代区 域进行扫描就可以了。



问题和建议?

课上技术相关的问题，都可以去群里和大家讨论，对于比较通用的、有价值的问题，可以去我们的知识星球提问。

具体技术之外的问题和建议，都可以找丢物线（微信：diuwuxian），丢丢会为你解答技术以外的一切。

觉得好?

如果你觉得课程很棒，欢迎给我们好评呀！https://ke.qq.com/comment/index.html?cid=381952

> 一定要是你真的觉得好，再给我们好评。不要仅仅因为对扔物线的支持而好评（报名课程已经是你最大的支持了，再不够的话 B 站多来点三连我也很开心），另外我们也坚决不做好评返现等任何的交易。我们只希望，在课程对你有帮助的前提下，可以看到你温暖的评价。

更多内容：

- 网站：https://hencoder.com；https://kaixue.io
- 各大搜索引擎、微信公众号、微博、知乎、掘金、哔哩哔哩、YouTube、西瓜视频、抖音、快手、微视：统一账号「**扔物线**」，我会持续输出优质的技术内容，欢迎大家关注。
- 哔哩哔哩快捷传送门：https://space.bilibili.com/27559447

大家如果喜欢我们的课程，还请去扔物线的哔哩哔哩，帮我素质三连，感谢大家！