昵称： Awecoder
园龄： 6年4个月
粉丝： 15
关注： 7
+加关注

< 2022年3月 >

日	一	二	三	四	五	六
27	28	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

最新随笔

1.使用注解管理spring bean生命周期

2.Java数组foreach赋值不生效的问题

3.合并两个有序链表

4.彻底搞清楚class常量池、运行时常量池、字符串常量池

5.JDK8打印并分析GC日志

6.如何查找对应的JVM参数

7.JVM故障运维工具的使用

8.对象内存分配过程

9.Java对象内存模型

10.Java对象创建流程

我的标签

JVM(13)

springboot(5)

mysql(3)

java(3)

图像处理(2)

彻底搞清楚class常量池、运行时常量池、字符串常量池

彻底搞清楚class常量池、运行时常量池、字符串常量池

常量池-静态常量池

也叫 **class文件常量池**，主要存放编译期生成的各种**字面量(Literal)和符号引用(Symbolic References)**。

- 字面量:例如文本字符串、fina修饰的常量。

```
int b = 2;
int c = "abcdefg";
```

- 符号引用:例如类和接口的全限定名、字段的名称和描述符、方法的名称和描述符

```
// 第3部分，常量池信息
Constant pool:
```

常量池-运行时常量池

- 当类加载到内存中后，JVM就会将class常量池中的内容存放到运行时常量池中；运行时常量池里面存储的主要是编译期间生成的字面量、符号引用等等。
- 类加载在链接环节的解析过程，会符号引用转换成直接引用（静态链接）。此处得到的**直接引用**也是放到运行时常量池中的。
- 运行期间可以动态放入新的常量。

常量池-字符串常量池

字符串常量池，也可以理解成运行时常量池分出来的一部分。类加载到内存的时候，字符串会存到字符串常量池里面。利用池的概念，避免大量频繁创建字符串。

- JDK6时字符串常量池位于运行时常量池，JDK7挪到堆中。

Hotspot8之前，使用持久代实现方法区，由于持久代内存不好估算，很容易到值OOM:Perm Gen异常。而元空间是本地内存，取决于操作系统分配内存。

字符串常量池位置变迁

Jdk1.6及之前： 有永久代, 运行时常量池在永久代，运行时常量池包含字符串常量池

Jdk1.7：有永久代，但已经逐步“去永久代”，字符串常量池从永久代里的运行时常量池分离到堆里

Jdk1.8及之后： 无永久代，运行时常量池在元空间，字符串常量池里依然在堆里

创建字符串操作

- 字面量赋值

```
String s = "lzp";
```

创建字符串对象，存放到字符串常量池中。s指向常量池中对象引用。

- new String对象

```
String c = new String("lzp");
```

new 新字符串对象，会在堆和字符串常量池中都创建对象。

- intern方法

递归(1)
简单(1)
链表(1)
leetcode(1)
G1收集器(1)
更多

积分与排名
积分 - 42089
排名 - 32855

随笔分类 (67)
codeinterview(1)
Java(4)
Java并发编程(4)
JVM(14)
mongoDB(1)
Mybatis(1)
MySQL(3)
Nginx(12)
Redis(1)
SLAM(5)
SpringBoot(11)
Tomcat(1)
web应用部署(4)
博客(1)
个人管理(1)
更多

随笔档案 (71)
2022年3月(1)
2022年2月(6)
2022年1月(6)
2021年12月(7)

String中的intern方法是一个 native 的方法，当调用 intern方法时，如果池已经包含一个等于此String对象的字符串（用equals(oject)方法确定），则返回池中的字符串。**否则，返回堆中String对象的引用(jdk1.6是将 堆中的String对象 复制到字符串常量池，再返回常量池中的引用)。**

```
String c = new String("lzp");
String d = c.intern();
System.out.println(c == d); // false
```

c指向堆对象，d指向常量池对象，因此必然不相等。

```
String s1 = new String("he") + new String("llo");
String s2 = s1.intern();
System.out.println(s1 == s2); // true
// 在 JDK 1.6 下输出是 false，创建了 6 个对象
```

JDK7以后会创建2个字符串常量池对象 "he","llo" ，new 3个堆对象 "he","llo","hello" ，字符串常量池没有hello对象引用。调s1的intern方法，hello指向new出来的hello对象。因此JDK7版本创建了5个对象。s1调intern()方法，返回堆中对象引用。

当然，很多博客中也说字符串常量池中保存的是堆对象的引用，即堆中有5个对象 2个he，2个llo，1个hello 。字符串常量池底层是hotspot的C++实现的，底层类似一个HashTable， 保存的本质上是**字符串对象的引用。**

众说纷纭，不好确定。但是两种情况的外在表现是一致的，字符串字面量对象在常量池中。

编译期优化

```
String a = "awecoder";
String b = "awe" + "coder";
System.out.println(a == b); // true

// 下面的也可以优化
"a" + 1 == "a1"
"a" + 3.4 = "a3.4"
```

b也是字面量，由于"awe"和"coder"在编译期已确定，JVM编译期将其优化为一个字符串字面量。

```
String a = "awecoder";
String b = "awe";
final String finalb = "awe";
System.out.println(a == b + "coder"); // false
System.out.println(a == finalb + "coder"); // true
```

编译期确定不了，例如new对象便不能优化。对于连接符"+"周围是否有变量，能够优化还是取决于变量是否确定。两者底层实现不同，一个是编译期优化成一个字面量，另一个底层使用StringBuilder的append()方法实现（反编译字节码文件可以观察到）。

分类: JVM

标签: JVM

Awecoder
关注 - 7
粉丝 - 15
+加关注

0 0

« 上一篇： JDK8打印并分析GC日志
» 下一篇： 合并两个有序链表

posted @ 2022-02-10 23:06 Awecoder 阅读(569) 评论(1) 编辑 收藏 举报