

Java方法区、栈及堆

原创

非 | 空

于 2017-11-19 15:45:19 发布

15403

★ 收藏 169

版权

分类专栏:

Thinking in Java

文章标签:

java内存

java方法区

java堆栈



Thinking in Java 专栏收录该内容

0 订阅

4 篇文章

订阅专栏

一 方法区 (Method Area)

1. 什么是方法区 (Method Area) ?

《深入理解JVM》书中对方法区 (Method Area) 描述如下:

方法区 (Method Area) 与Java堆一样, 是各个线程共享的 **内存** 区域。

2.方法区 (Method Area) 存储什么?

《深入理解JVM》书中对方法区 (Method Area) 存储内容描述如下:

它存储已被Java虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等

2.1 方法区 (Method Area) 存储的类信息

对每个加载的类型 (类class、接口interface、枚举enum、注解annotation) , JVM必须在方法区中存储以下类型信息:

- 这个类型的**完整有效名称** (全名=包名.类名)
 - 这个类型**直接父类**的完整有效名称(`java.lang.Object` 除外, 其他类型若没有声明父类, 默认父类是Object)
 - 这个类型的修饰符(`public`、`abstract`、`final` 的某个子集)
 - 这个类型**直接接口**的一个有序列表
- 除此之外还方法区 (Method Area) 存储类信息还有
- 类型的常量池(constant pool)
 - 域(Field)信息
 - 方法(Method)信息
 - 除了常量外的所有静态(static)变量

方法区 (Method Area) 存储类信息请参考: [参考博客](#)

2.2 方法区 (Method Area) 存储的常量

- `static final`修饰的**成员变量**都存储于 方法区 (Method Area) 中

2.3 方法区 (Method Area) 存储的静态变量

- 静态变量又称为**类变量**, 类中被`static`修饰的**成员变量**都是静态变量 (类变量)
- 静态变量之所以又称为**类变量**, 是因为静态变量和类关联在一起, 随着类的加载而存在于方法区 (而不是堆中)
- 八种基本数据类型 (byte、short、int、long、float、double、char、boolean) 变量会在方法区开辟空间, 并将对应



非 | 空

关注

55

5

★ 169

如果未用 `new` 关键字为引用类型的静态变量分配对象（如：`static Object obj;`）那么对象的引用obj会存储在方法区中，并为其指定默认值 `null`；若，对于引用类型的静态变量如果用 `new` 关键字为引用类型的静态变量分配对象（如：`static Person person = new Person();`），那么对象的引用person 会存储在方法区中，并且该对象在堆中的地址也会存储在方法区中（注意此时静态变量只存储了对象的堆地址，而对象本身仍在堆内存中）；这个过程还涉及到静态变量初始化问题，可以参考博客：[静态变量初始化相关](#)

2.4 方法区（Method Area）存储的方法（Method）

- 程序运行时会加载类编译生成的字节码，这个过程中静态变量（类变量）和静态方法及普通方法对应的字节码加载到方法区。
- 但是！！方法区中没有实例变量，这是因为，类加载先于对应类对象的产生，而实例变量是和对象关联在一起的，没有对象就不存在实例变量，类加载时没有对象，所以方法区中没有实例变量
- 静态变量（类变量）和静态方法及普通方法在方法区（Method Area）存储方式是有区别的

二 栈（Stack）

栈（Stack）：线程私有的内存区域

- 每个方法（Method）执行时，都会创建一个栈帧，用于存储局部变量表、操作数栈、动态链接、方法出口信息等
- 栈中所存储的变量和引用都是局部的（即：定义在方法体中的变量或者引用），局部变量和引用都在栈中（包括final的局部变量）
- 八种基本数据类型（byte、short、int、long、float、double、char、boolean）的**局部变量（定义在方法体中的基本数据类型的变量）**在栈中存储的是它们对应的值
- 栈中还存储局部的对象的引用（定义在方法体中的引用类型的变量），对象的引用并不是对象本身，而是对象在堆中的地址，换句话说，局部的对象的引用所指对象在堆中的地址在存储在了栈中。当然，如果对象的引用没有指向具体的对象，对象的引用则是 `null`

三 Java堆（Java Heap）

Java堆（Java Heap）：被所有线程共享的一块内存区域，在虚拟机启动时创建。Java堆（Java Heap）唯一目的就是存放对象实例。所有的对象实例及数组都要在**Java堆（Java Heap）**上分配内存空间。

- 由关键字 `new` 产生的所有对象都存储于Java堆（Java Heap）
- ！！！**实例变量（非static修饰的成员变量）**和对象关联在一起，所以实例变量也在堆中
- java数组也在堆中开辟内存空间

四 示例

```
1 public class PersonDemo
2 {
3     public static void main(String[] args)
4     { //局部变量p和形参args都在main方法的栈帧中
5         //new Person()对象在堆
6         Person p = new Person()
7         //sum在栈中, new int[]
```

非 | 空

关注

```
8         int[] sum = new int[10];
9     }
10 }
11
12
13 class Person
14 {    //实例变量name和age在堆(Heap)中分配空间
15     private String name;
16     private int age;
17     //类变量(引用类型)name1和"cn"都在方法区(Method Area)
18     private static String name1 = "cn";
19     //类变量(引用类型)name2在方法区(Method Area)
20     //new String("cn")对象在堆(Heap)中分配空间
21     private static String name2 = new String("cn");
22     //num在堆中, new int[10]也在堆中
23     private int[] num = new int[10];
24
25
26     Person(String name,int age)
27     {
28         //this及形参name、age在构造方法被调用时
29         //会在构造方法的栈帧中开辟空间
30         this.name = name;
31         this.age = age;
32     }
33
34     //setName()方法在方法区中
35     public void setName(String name)
36     {
37         this.name = name;
38     }
39
40     //speak()方法在方法区中
41     public void speak()
42     {
43         System.out.println(this.name+"..."+this.age);
44     }
45
46     //showCountry()方法在方法区中
47     public static void showCountry()
48     {
49         System.out.println("country="+country);
50     }
51 }
52
```

五 参考资料

- 【1】周志明. 深入理解Java虚拟机[M]. 机械工业出版社, 2011.
- 【5】毕向东.Java基础教学视频
- 【3】[官方文档](#)
- 【4】<http://blog.csdn.net/u013241673/article/details/78221857>
- 【5】<http://blog.csdn.net/zzhangxiaoyun/article/details/7518917>

六 不足之处，请不吝指教



欢迎关注公众号（微信搜索: non_null）,



非 | 空 关注

👍 55



💬 5



★ 169

