深入理解Java Runtime Area Java运行时数据区

六尺帐篷 (/u/f8e9b1c246f1)
2017.08.05 11:57* 字数 1473 阅读 81 评论 0 喜欢 4
(/u/f8e9b1c246f1)

编辑文章 (/writer#/notebooks/15069334/notes/15407046)

- Java Runtime Area的分类
- 从线程的角度理解Java Runtime Area
- 从存储内容理解Java Runtime Area
- 方法区中究竟存储了哪些信息?
- 基本数据类型的成员变量放在jvm的哪块内存区域里?

Java Runtime Area的分类

Java Runtime Area主要可以分为六部分：

- Program Counter (PC) Register **程序计数器**
- Java Virtual Machine Stacks **Java虚拟机栈**
- Heap Memory **Java堆**
- Method Area **方法区**
- Run-time Constant Pool **运行时常量池**
- Native Method Stacks **本地方法栈**

具体的每个区域的内容和特点可以参考《深入理解Java虚拟机》,此书已经讲的很详细了。
下面我们对这几个数据区域进行分类，分别从不同的视角来分析，加深我们的理解

从线程的角度理解Java Runtime Area

首先，我们从区域是否是线程私有的还是所有线程共享的来分类：

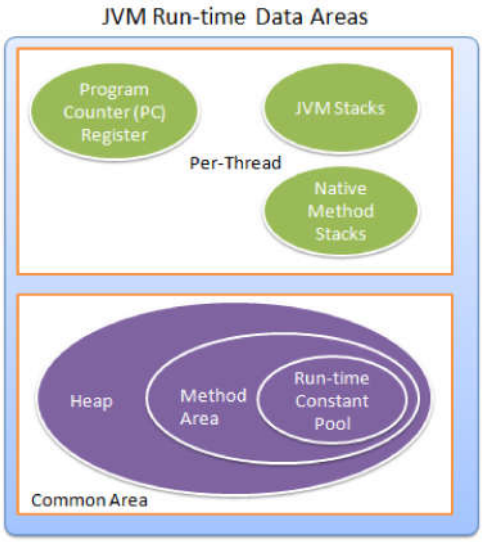

image.png

程序计数器 Java虚拟机栈 本地方法栈都是线程私有的
而
Java堆**方法区****运行时常量池**都是所有线程共享的

进一步理解：

- 对于线程私有的数据区域**程序计数器 Java虚拟机栈 本地方法栈**，他们的生存周期都是一致的，都是
  随着线程开始，而进行初始化
  随着线程结束而销毁

- 而对于线程共享的数据区域**Java堆****方法区****运行时常量池**，他们的生存周期都是一致的
  随着JVM的启动而分配内存
  随着JVM的关闭而销毁

从存储内容理解Java Runtime Area

下面我们再根据不同区域所存储的数据类型进行分类：
可以分为三类

- 方法区和常量池存储类的信息
- 堆内存存储对象信息
- 程序计数器，Java虚拟机栈，本地方法栈存储线程的信息

下图很清楚的说明

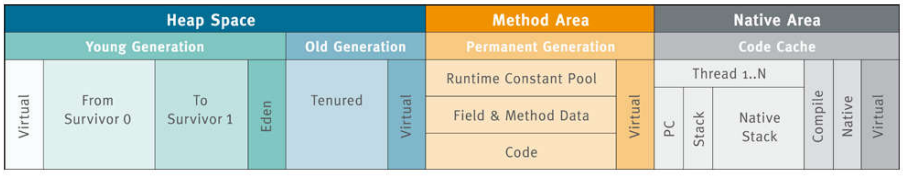

image.png

The heap space holds object data, the method area holds class code, and the native area holds references to the code and object data.
堆存储object的data，方法区存储class的信息和code，native区域存储指向class信息

和code的引用和指向对象的data的引用

下面这个图更详细的指出了三个区域存储的内容：

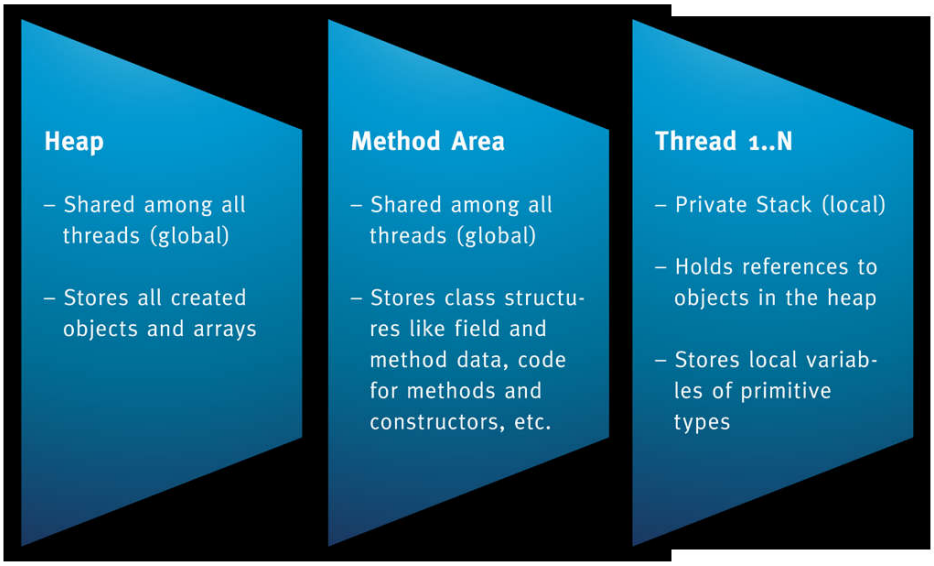

image.png

下面我们通过一个实际代码的例子，来说明;

看下面这段代码：

```
import java.text.SimpleDateFormat;
import java.util.Date;

import java.util.logging.Logger;;

/**
 *
 * @author Tai Truong
 *
 */
public class HelloWorld {
    private static Logger LOGGER = Logger.getLogger(HelloWorld.class.getName());

    public void sayHello(String message) {
        SimpleDateFormat formatter = new SimpleDateFormat("dd.MM.YYYY");
        String today = formatter.format(new Date());
        LOGGER.info(today + ": " + message);
    }
}
```

image.png

这段代码编译之后，就存储成如下这个样子：

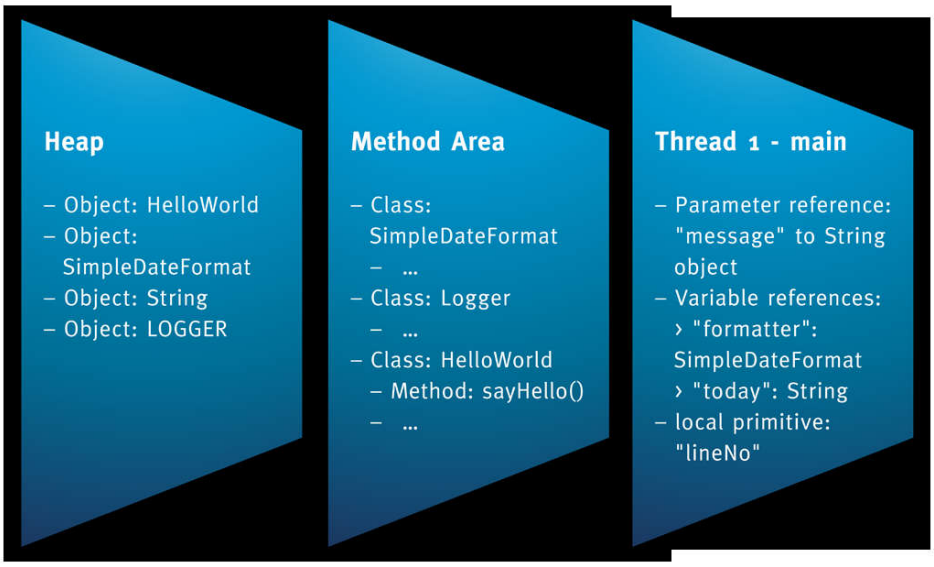

image.png

易混淆的Java Runtime Area 的问题

下面我们会对关于Java 运行时数据区易混淆的问题进行释疑

方法区中究竟存储了哪些信息?

栈中存放了局部变量表等与方法有关的信息，但方法中还有指令代码这一重要内容，它既没有放在栈(Stack)中也没放在堆(Heap)中，那它放在哪呢?
其实，方法区中除了包括你所说的“已加载的类的基本信息、常量、静态变量等”外，还包括编译器编译后的代码，而且这应该是方法区中主要的一部分，毕竟类中主要是方法和属性，而类中的属性，如果是实例域的话则新建对象后存储在堆(Heap)中，静态的话就如你所说存储在方法区中，因此该区域中方法占主要部分，这应该是此运行时数据区称为方法区的原因吧。

基本数据类型的成员变量放在jvm的哪块内存区域里?

比如

```
class{
private int i;
}
```

有的朋友可能因为基本数据类型，就认为存储在栈中。但其实是存储在堆中的，因为这是属于对象的信息，每个对象都拥有不同的实例变量，这些实例变量都存储在堆中，不管是基本数据类型还是引用数据类型
ava虚拟机栈是线程私有的，生命周期跟线程相同，每个方法调用的时候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，方法出口等信息。每个方法调用的过程，就代表了一个栈帧在虚拟机栈中入栈到出栈的过程，当进入一个方法时，这个方法在栈中需要分配多大的内存都是完全确定的，方法运行时不会改变局部变量表的大小
——《深入理解java虚拟机第二版》
很多java程序员一开始就被网上的一些教程所误导：基本数据类型放在栈中，数组和类的实例放在堆中。 这个说法不准确，事实上，如上面的实例变量i，他是存放在java堆中。因为它不是静态的变量，不会独立于类的实例而存在，而该类实例化之后，放在堆中，当然也包含了它的属性i。
如果在方法中定义了int i = 0;则在局部变量表创建了两个对象：引用i和0。 这两个对象都是线程私有（安全）的。 比如定义了int[] is = new int[10]. 定义了两个对象，一个是is引用，放在局部变量表中，一个是长度为10的数组，放在堆中，这个数组，只能通过is来访问，方法结束后出栈，is被销毁，根据java的根搜索算法，判断数组不可达，就将它销毁了。

JVM (/nb/15069334)

六尺帐篷 (/u/f8e9b1c246f1)
写了 248002 字，被 15247 人关注，获得了 1422 个喜欢
(/u/f8e9b1c246f1)

喜欢 4

更多分享

(http://cwb.assets.jianshu.io/notes/images/154070

被以下专题收入，发现更多相似内容

投稿管理

+ 收入我的专题

 Android... (/c/5139d555c94d?utm_source=desktop&utm_medium=notes-included-collection)

 Android开发 (/c/d1591c322c89?utm_source=desktop&utm_medium=notes-included-collection)

 Android知识 (/c/3fde3b545a35?utm_source=desktop&utm_medium=notes-included-collection)

 程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)

 编程设计之Java (/c/0ffed0f3a386?utm_source=desktop&utm_medium=notes-included-collection)