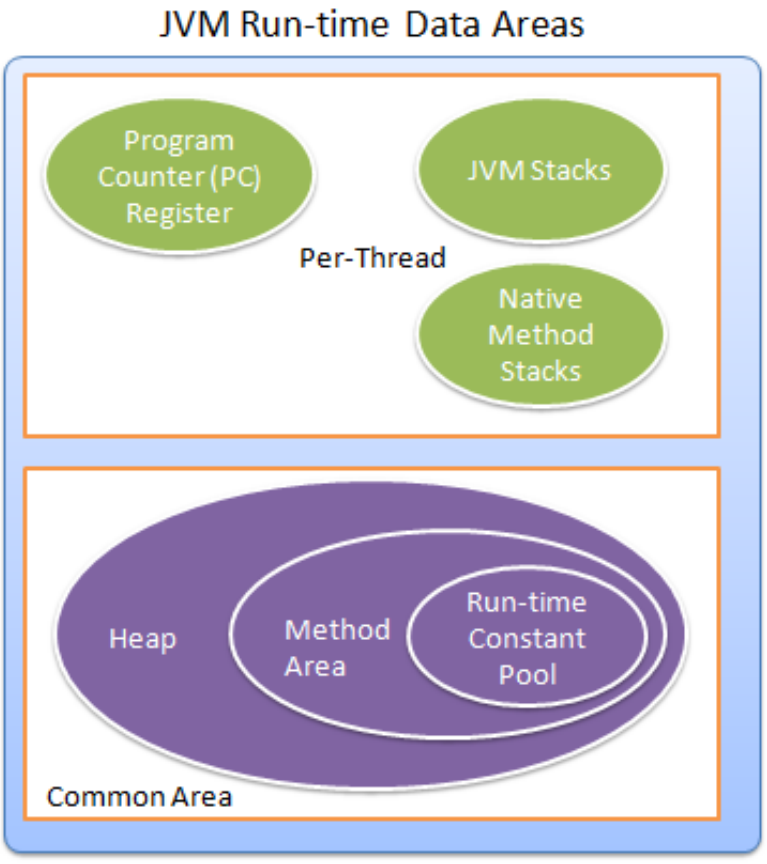
理解JVM运行时的数据区是Java编程中的进阶部分。我们在开发中都遇到过一个很头疼的问题就是OutOfMemoryError（内存溢出错误），但是如果我们了解JVM的内部实现和其运行时的数据区的工作机制，那么前面的问题就会迎刃而解。在这片文章中，我们将简单了解JVM中有哪些运行时数据区以及这些数据区的工作机制。

JVM运行时数据区分类

- 程序计数器（Program Counter (PC) Register）
- JVM栈（Java Virtual Machine Stacks）
- 堆内存（Heap Memory）
- 方法区（Method Area）
- 运行时常量池（Run-time Constant Pool）
- 本地方法栈（Native Method Stacks）

有图才能说



按线程持有划分

查看上面的图，可以得知以上六个数据区其实线程私有还是共享，可以分为如下两种。

- 单个线程私有(Managed Per-Thread) 属于这一种的数据区包含 程序计数器， JVM栈还有本地方法栈。 每个线程都私有这三个数据区，这些数据区在其所属的线程创建时初始化，并随着所属线程结束被销毁。
- 多个线程共享 属于这一种的数据区包含 堆内存，方法区和运行时常量池。这些数据区可以被每一个线程访问，他们随着JVM启动而初始化，同时伴随JVM关闭而销毁。

程序计数器

在通用的计算机体系中，程序计数器用来记录当前正在执行的指令，在JVM中也是如此。程序计数器是线程私有，所以当一个新的线程创建时，程序计数器也会创建。由于Java是支持多线程，Java中的程序计数器用来记录当前线程中正在执行的指令。如果当前正在执行的方法是本地方法，那么此刻程序计数器的值为undefined。注意这个区域是唯一一个不抛出OutOfMemoryError的运行时数据区。

JVM栈

在介绍JVM栈之前，简单介绍一个概念，栈帧

栈帧

一个栈帧随着一个方法的调用开始而创建，这个方法调用完成而销毁。栈帧内存放者方法中的局部变量，操作数栈等数据。

JVM栈只对栈帧进行存储，压栈和出栈操作。栈内存的大小可以有两种设置，固定值和根据线程需要动态增长。在JVM栈这个数据区可能会发生抛出两种错误。

- StackOverflowError 出现在栈内存设置成固定值的时候，当程序执行需要的栈内存超过设定的固定值会抛出这个错误。
- OutOfMemoryError 出现在栈内存设置成动态增长的时候，当JVM尝试申请的内存大小超过了其可用内存时会

抛出这个错误。

堆数据区

堆数据区是用来存放对象和数组（特殊的对象）。堆内存由多个线程共享。堆内存随着JVM启动而创建。众所周知，Java中有一个很好的特性就是自动垃圾回收。垃圾回收就操作这个数据区来回收对象进而释放内存。如果堆内存剩余的内存不足以满足于对象创建，JVM会抛出OutOfMemoryError错误。

方法区

在JVM规范中，方法区被视为堆内存的一个逻辑部分。这一点可能由于具体的JVM实现而不同，甚至在方法区不实现垃圾回收处理也是可以的。方法区和堆内存一样被多个线程访问，方法区中存放类的信息，比如类加载器引用，属性，方法代码和构造方法和常量等。当方法区的可用内存无法满足内存分配需求时，JVM会抛出OutOfMemoryError错误。

运行时常量池

运行时常量池创建在方法区，当一个类或者一个接口被创建的时候，JVM会创建一个运行时常量池。一个运行时常量池实际上是一个类或者接口的class文件中常量池表（constant_pool table）的运行时展示形式。一个运行时常量池包含了多种类型的常量，从诸如运行时可以确定的数值型字面量到运行时才能决定的方法和属性引用。当运行时常量池无法满足于内存分配需求时，JVM会抛出OutOfMemoryError错误。

本地方法栈

一个支持native方法调用的JVM实现，需要有这样一个数据区，就是本地方法栈，Java官方对于本地方法的定义为methods written in a language other than the Java programming language，就是使用非Java语言实现的方法，但是通常我们指的一般为C或者C++，因此这个栈也有着C栈这一称号。一个不支持本地方法执行的JVM没有必要实现这个数据区域。本地方法栈基本和JVM栈一样，其大小也是可以设置为固定值或者

动态增加，因此也会对应抛出StackOverflowError和OutOfMemoryError错误。

译文信息

- 原文出处Java JVM Run-time Data Areas 原文有些东西偏于隐晦，译文有这方面的修改。
- 翻译的不错。Chapter 2. The Structure of the Java Virtual Machine

注：本文介绍JVM运行时数据相对比较概括，以后会有更加详细的针对单个数据区的介绍。