JDK8的JVM内存结构，元空间替代永久代成为方法区及常量池的变化

JVM的知识这里总结的很详细：https://github.com/doocs/jvm/blob/master/README.md，因此在本博客也不会再对其中的东西重复总结了。

现在很多文章关于JVM内存结构的说法模糊不清，这里记录一下以前的一些比较模糊的JVM相关概念的重新认识。都是经过多处考证对比的。

MetaSpace代替Perm Gen

即元空间代替了永久代,所以JVM关于永久代的参数也都作废了，取而代之的是关于MetaSpace空间的参数。而且Mete Space是属于直接内存。示意图：

（图片来源于网络）

为什么要在直接内存里拿出来一块内存作为元空间取代永久代呢？主要的说法有以下几个：

（1）类及方法的信息等比较难确定其大小，因此对于永久代的大小指定比较困难，太小容易出现永久代溢出，太大则容易导致老年代溢出。

（2）永久代会为 GC 带来不必要的复杂度，并且回收效率偏低。

即方便分配管理，因为直接内存空间比较充足；便于回收，因为永久代本来回收垃圾的事件发生概率很低，直接从JVM中拿出可以提高回收效率。

方法区与永久代的关系

很多文章里喜欢把方法区等同与永久代，永久代既然没了，方法区也就没了。但我认为方法区只是一种逻辑上的概念，永久代指物理上的堆内存的一块空间，这块实际的空间完成了方法区存储字节码、静态变量、常量的功能等等。既然如此，现在元空间也可以认为是新的方法区的实现了。

常量池随永久代的变化

常量池主要可以分为以下几种：

（1）静态常量池：即*.class文件中的常量池，class文件中的常量池不仅仅包含字符串/数字这些字面量，还包含类、方法的信息，占用class文件绝大部分空间。这种常量池主要用于存放两大类常量：字面量和符号引用量，字面量相当于Java语言层面常量的概念，如文本字符串，声明为final的常量值等；符号引用则属于编译原理　方面的概念，包括了如下三种类型的常量：类和接口的全限定名、字段名称描述符、方法名称描述符。

类的加载过程中的链接部分的解析步骤就是把符号引用替换为直接引用，即把那些描述符（名字）替换为能直接定位到字段、方法的引用或句柄（地址）。

（2）运行时常量池：虚拟机会将各个class文件中的常量池载入到运行时常量池中，即编译期间生成的字面量、符号引用，总之就是装载class文件。

（3）字符串常量池 ：字符串常量池可以理解为运行时常量池分出来的部分。加载时，对于class的静态常量池，如果字符串会被装到字符串常量池中。

（4）整型常量池：Integer，类似字符串常量池，可以理解为运行时常量池分出来的。加载时，对于class的静态常量池装的东西，如果是整型会被装到整型常量池中。

类似的还有Character、Long等等常量池（基本数据类型没有哦）。。。

在永久代移除后，字符串常量池也不再放在永久代了，但是也没有放到新的方法区---元空间里，而是留在了堆里（为了方便回收？）。运行时常量池当然是随着搬家到了元空间里，毕竟它是装静态变量、字节码等信息的，有它的地方才称得上方法区。