1. Java虚拟机运行时数据区域

Java虚拟机在执行Java程序的过程中会把它所管理的内存划分为多个不同的数据区域，这些区域有各自的用途以及创建和销毁时间。

（一）程序计数器

程序计数器占用内存较少，用作当前线程所执行的字节码的行号指示器，字节码解释器工作时就是通过改变程序计数器的值来选取下一条要执行的字节码指令。Java程序的多线程是通过线程的切换实现的，任一时刻，任一个处理器都只会执行一条线程中的指令，因此，为了线程切换后能够恢复到正确的执行位置，每个线程的程序计数器都是私有的。

如果线程执行的是一个Java方法，则程序计数器记录的是正在执行的虚拟机字节码指令的地址；如果线程执行的是本地方法，则计数器值为空。且程序计数器是唯一一个没有规定任何OutOfMemory情况的区域。

（二）Java虚拟机栈

Java虚拟机栈也是线程私有的，生命周期和线程相同。该栈描述的是Java方法执行的线程内存模型，每个方法执行时，虚拟机会同步创建一个栈帧，存储局部变量表，操作数栈，动态连接，出口信息。方法调用到执行完毕，对应着该栈帧在虚拟机栈中入栈到出栈，

局部变量表存放编译器已知的Java虚拟机基本数据类型，如boolean,byte,char…；对象引用类型，如指向对象起止地址的引用指针，执行对象的句柄，returnAddress类型，如指向一条字节码指令的地址。数据类型在局部变量表中的存储空间用局部变量槽表示，64位的long和double类型会占用两个变量槽，其余的数据类型只占一个。当进入一个方法时，该方法需要在栈帧中分配的内存空间的大小（变量槽的数量）在编译期间已经确定好，运行期间不会改变，而不同虚拟机的变量槽具体大小（比特数）由虚拟机具体实现。

Java虚拟机栈有两种异常：

1. 如果线程请求的栈深度大于虚拟机所允许的深度，将抛出StackOverflowError异常；
2. 如果Java虚拟机栈容量可扩展，扩展时却无法申请到足够的内存，将抛出OutOfMemoryError异常。

（三）本地方法栈

与虚拟机栈作用类似，不过虚拟机栈为虚拟机执行Java方法（字节码）服务，而本地方法栈为虚拟机使用到的本地方法服务。

（四）Java堆

虚拟机管理的内存中最大的一块，所有线程共享，在虚拟机启动时创建。堆用于存放所有的对象实例，同时也是垃圾收集器管理的内存区域。从分配内存和内存回收的角度来看，Java堆可以划分为多个线程私有的分配缓冲区，以提升对象分配时的效率，以及回收内存的效率。

Java堆可以处于物理上不连续的内存空间中，但是在逻辑上它是连续的。对于大文件，如数组对象，多数虚拟机处于实现简单的考虑，会要求连续的内存空间。

Java堆是可扩展的，如果在Java堆中没有内存完成实例分配，并且堆也无法再扩展时，Java虚拟机会抛出OutOfMemoryError异常。

（五） 方法区

方法区是所有线程共享的内存区域，用于存储已被虚拟机加载的类型信息，常量，静态变量，即时编译器编译后的代码缓存等数据。方法区常常被描述为Java堆的一个逻辑部分。JDK6以前，HotSpot虚拟机设计团队选择把垃圾收集器的分代设计扩展至方法区，即使用永久代来实现方法区，这样HotSpot的垃圾收集器就能像管理Java堆一样管理方法区了。这种设计导致了Java应用容易遇到内存溢出的风险，因为永久代有-XX:MaxPermSize的内存上限。JDK7以后，HotSpot将原本放在永久代中的字符串常量池，静态变量溢出；JDK8，废弃了永久代概念，采用本地内存中实现的元空间，将永久代中的内容都移到元空间中。

方法区的约束比较少，不需要连续物理内存、可选择固定内存或扩展、可选择不实现垃圾收集。方法区的内存回收主要是针对常量池的回收以及对类型的卸载，可是回收效果很不好，可又需要回收这些内存，否则可能会出现内存泄露。

如果方法区无法满足新的内存分配需求，将抛出OutOfMemoryError异常。

（六）运行时常量池

运行时常量池是方法区的一部分，Class文件中除了有类的版本，子段，方法，接口等描述信息外，还有常量池表，存放编译器生成的各种字面量和符号引用，这些内容将在类加载后放到方法区的运行时常量池中。

运行常量池相对于Class文件常量池表的重要特征是动态性，Java语言未要求常量只有在编译期间产生，运行期间也可将新的常量放入池中，这种特性对应着String类的intern()方法。