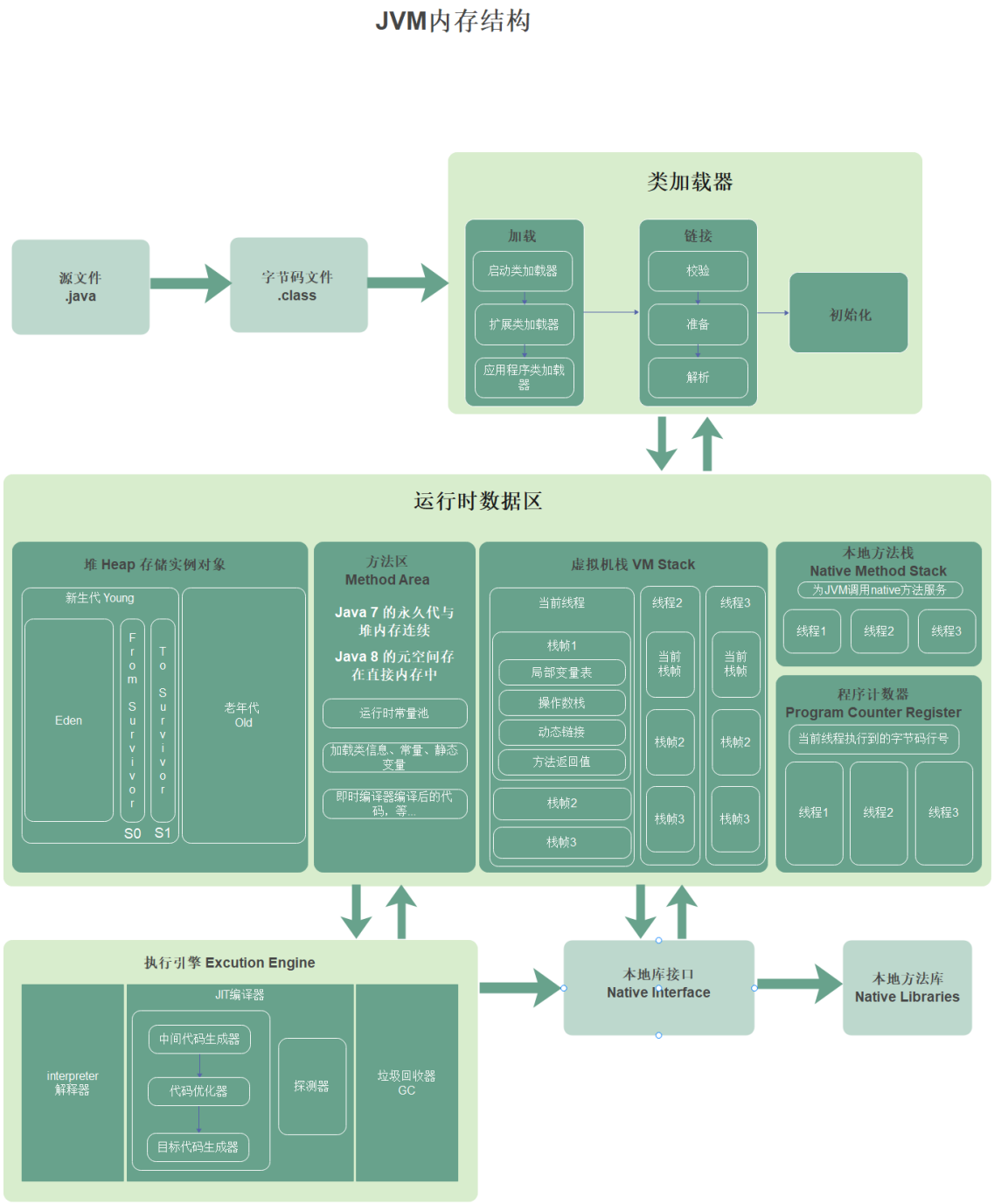
【第20话：碰到年龄大一点的面试官就容易被问到的JVM内存结构】

Hello 小伙伴们，这结课给大家讲解下JVM内存结构。这是一个出现频率特别高的面试题。尤其碰到年龄大一点的面试官，特别容易问到这个问题。

话不多说，我们先上图。



我们领小伙伴们一点一点看一下图中的每个部分。

源文件

源文件就是我们编写Java代码的文件。文件扩展名为.java

字节码文件

字节码文件是源文件编译后的文件。字节码文件是二进制文件，需要通过特定的工具才能查看。里面存放了源文件编译后的字节码指令。

类加载器 Class Loader

Java 程序运行时会由类加载器负责把.class的字节码文件装在到内存中，供虚拟机执行。

加载 Loading

1. **启动类加载器 BootStrap Class Loader**

   负责从启动类中加载类。具有最高执行优先级。即：rt.jar等。

2. **扩展类加载器 Extension Class Loader**

   负责加载扩展相关类。即：jre/lib/ext 目录

3. **应用程序加载器 Application Class Loader**

   加载应用程序类路径(classpath)中相关类

链接 Linking

1. **校验 Verify**

校验器会校验字节码文件是否正确。

2. **准备 Prepare**

所有静态变量初始化并赋予默认值

3. **解析 Resolve**

符号引用被换成直接引用。

初始化 Initialization

所有静态变量赋予初值，静态代码块执行。

执行引擎

运行时数据区的字节码会交给执行引擎执行

解释器 Interpreter

解释器负责解释字节码文件。每次方法调用都会被重新解释。

JIT编译器

JIT 即时编译器。对于多次被使用的方法会放入到本地内存中，下次就可以直接调用了。

探测器

负责探测多次被调用的代码。

垃圾回收器 GC

负责回收不在被使用的对象。GC是JVM中非常重要的一块，在后面我们会单独讲解GC。

本地库接口

在Java代码中使用native修饰的方法表示方法具体实现使用其他编程语言实现的。例如：C语言。通过本地库接口为Java程序提供调用其他语言的实现方案。

本地方法库

所有的本地方法，通过本地库接口调用。

程序计数器

程序计数器简称：PC Register

程序计数器是一块较小的内存空间。记录了当前线程执行到的字节码行号。每个线程都有自己的程序计数器，相互不影响。如果是native方法，计数器为空。

虚拟机栈

虚拟机栈跟随线程创建而创建，所以每个线程都有一个虚拟机栈。

虚拟机栈中存储的是栈帧（frames），每个栈帧对应一个方法，每个栈帧都有自己的局部变量表、操作数栈、动态链接和返回地址等。当前正在执行的方法称为当前方法，当前方法所在的帧称为当前帧。方法执行时帧就是一个入栈操作，方法执行完成之后栈帧就是一个出站操作。

局部变量表

局部变量表存储的8大基本数据类型和返回值以及方法参数及对象的引用。 其中long和double占用2倍长度。

局部变量表就是一个数组，数组的长度在编译器确定。通过从0开始的索引调用局部变量表的内容。

对于类方法，从索引0开始连续N个作为方法传递。对于实例方法索引0存储的都是实例化方法的实例对象的引用。

操作数栈

操作数栈存在于栈帧中，其大小在编译期确定。

操作数栈中存储了class文件中虚拟机指令以及准备要传递的参数和接收对方的返回结果。

运行时常量池中数据以及局部变量表中得值都可以由操作数栈进行获取。

动态链接

方法·把符号转换为直接引用。

在JVM加载或第一次使用转换时称为静态链接或静态解析。而在运行期间把符号转换为直接引用时就称为动态链接。

方法返回地址

方法返回地址分为两种情况：

1. 正常结束执行。例如碰见return关键字。调用程序计数器的值后当前栈帧直接出栈就可以了。

2. 异常结束。可能需要恢复上层方法的局部变量表和操作数栈，然后把返回值压如到栈帧的操作数栈中，之后调用程序计数器的值后获取到下条指令。

堆

堆是所有线程共享的，存储类的实例和数组。

堆是在虚拟机启动时创建的，由GC负责回收。

堆可以是一块不连续的内存空间。

在Java 8 中，String是存在于堆中的。

堆被分为二大部分：

在Java 7时分为：新生代（Young Generation）、老年代（Old Generation）。在HotSpot中使用永久代来实现方法区的规范。且新生代、老年代和永久代是连续的。

新生代又被分为Eden区、From Survivor区、To Survivor区。官方说明默认分配比例为8：1：1。但是使用jmap工具进行测试时发现比例为6：1：1。

在Java 8时把永久代替换为元空间（MetaSpace），也就是说在Java8中使用元空间来实现方法区。且在Java8中把元空间移植到本地内存上（Native Memory），其实在Java 7 时，永久代在堆空間中，但是部分数据已经移植到本地内存上了。

方法区

方法区是线程共享的。

在虚拟机启动时自动创建方法区，方法区可以是一块不连续的内存空间。
方法区可以理解为编译代码存储区。在方法区中存储每个类的结构、运行时常量池、字段、方法、构造方法。

在JVM规范上方法区是一个独立的区域，但是在Java SE7 的HotSpot 上方法区使用永久代作为实现，永久代和堆是一块连续空间。在Java SE8的JVM规范实现上，HotSpot使用元空间实现方法区。

运行时常量池

运行时常量池存在于方法区。存储每个类或每个接口从编译时已知的数字文字到必须在运行时解析的方法和字段引用。

运行时常量池存在于方法区。

存储每个类或每个接口从编译时已知的数字文字到必须在运行时解析的方法和字段引用。