图文 # 014、第2周答疑：本周问题答疑，上周作业点评

3732 人次阅读　2019-07-14 07:00:00

详情　评论

第2周问题答疑

问题一：

既然栈帧存放了方法对应的局部变量的数据也包括了方法执行的其它相关信息，那为什么不把程序计算器那块记录执行的情况，也放在各个方法自己的栈帧里，而是要单独列一个程序计数区去存储呢？请教，谢谢

答：

这就是JVM设计者的设计思想了，因为程序计数器针对的是代码指令的执行，Java虚拟栈针对的是放方法的数据，一个是指令，一个是数据，分开设计

问题二

对于第二点问题，我理解是在初始化的阶段执行该段代码的，对么？那在执行的时候也是按照jvm执行代码的这套流程来执行的么？ || 老师 1.静态变量是放在什么内存区域呢？2.然后静态代码块也是会生成一个栈帧然后压栈到当前虚拟机栈中吗,是不是只压栈一次？

答：

是在类初始化的时候来执行，但是不是JVM执行代码的流程来执行，他是类初始化，自成一套体系

问题三

回答问题：首先该类的所有实例（堆中）都已经被回收；其次该类的ClassLoader已经被回收；最后，对该类对应的Class对象没有任何引用。满足上面三个条件就可以回收该类了。

答：

正解

问题四

怎么判断离代码的远近？如果是一个mavne聚合工程，两个子工程都定义了这个类，这个时候会加载哪一个？按照classpath的声明顺序吗，先声明先加载？ || 我在使用mybatis时发现有个类不满足要求，于是在项目中新建同包同名类，复制源码然后加上我自己的逻辑，这个时候就会加载我自己写的那个类，而不是mybatis中的类，这是为什么呢？

答：

没错，他有一个顺序，你自己的那个代码是最近的，他会优先加载你代码里的那个类

问题五

文章很详细，但是对于新手而言还是很抽象。我在学这块的时候都是画了很多内存开辟的流程图。希望老师抽空写一些简单的代码，把内存开辟和初始化，赋值的步骤一步一步画出来。

答：

思路很好，同学，但是你是已经有一定了解的，都提到了内存开辟和赋值流程了，其实对新手而言，了解到文章目前的程度就可以了，不能一口气太深，要慢慢来，刚开始浅显易懂一些，接下来逐步深入。

问题六

感觉也可以按照回收堆内存对象的思路来思考。方法区的class可以回收，只需要满足任何地方都没有用到该class，即：没有任何实例，没有调用class的静态变量或静态方法，没有利用反射访问class。暂时想到的只有这些，不知道对不对

答：

推测基本正确

问题七

方法执行完后, 栈帧立马被出栈, 那该栈帧中的变量等数据是立马就被回收掉吗?还是需要等垃圾回收线程扫描到再回收掉?

答：

出栈就没了

问题八

那如果把public static int flushInterval = Configuration.getInt("xxx");中的static去掉, 那后面的getInt是在什么时候执行的呢 ,我自己测试了一下,好像是在构造方法之前执行的, 不明白这个到底属于什么阶段?

答：

这是属于类的对象实例初始化的阶段

问题九

双亲委派模型设计的出发点很重要，文章漏了 对于任意一个类，都需要由加载它的类加载器和这个类本身一同确立其在Java虚拟机中的唯一性，每一个类加载器，都拥有一个独立的类名称空间。

也就是说，，判断2个类是否“相等”，只有在这2个类是由同一个类加载器加载的前提下才有意义，否则即使这2个类来源于同一个Class文件，被同一个虚拟机加载，只要加载它们的类加载器不同，这2个类必定不相等。

基于双亲委派模型设计，那么Java中基础的类，Object类似Object类重复多次的问题就不会存在了，因为经过层层传递，加载请求最终都会被Bootstrap ClassLoader所响应。加载的Object类也会只有一个，否则如果用户自己编写了一个java.lang.Object类，并把它放到了ClassPath中，会出现很多个Object类，这样Java类型体系中最最基础的行为都无法保证，应用程序也将一片混乱

答:

这位同学非常不错，对jvm有一定的研究，不过我们第一周的文章，并不是说漏掉你说的这些点，而是我们的写作思路，是循序渐进，这点很重要。

如果在刚开始就给出大段这种说明，那么只有少数人会看懂，回到普通的那种学院派纯理论的知识传递方法了。你说的很好，不过希望耐心跟着继续看，我们会有意把很多细节放在后面讲，循序渐进，保证很多小白同学都轻松学习，这点很重要。

问题十

tomcat需要破坏双亲委派模型的原因:

(1)tomcat中的需要支持不同web应用依赖同一个第三方类库的不同版本，jar类库需要保证相互隔离;

(2)同一个第三方类库的相同版本在不同web应用可以共享

(3)tomcat自身依赖的类库需要与应用依赖的类库隔离 (3)jsp需要支持修改后不用重启tomcat即可生效 为了上面类加载隔离和类更新不用重启，定制开发各种的类加载器

答:

解答很详细，我们给的一些思考题，其实第二天只会给一些思路，鼓励大家自己查阅资料去思考，给出答案，给一些指引

问题十一

老师您好，我想问一下，我们的应用如果关掉，创建在堆中的对象，还有方法区的数据都还在吗

答:

应用关了，那么系统对应的JVM进程就没了，那JVM内存区域的数据就全没了

问题十二

请问老师:

1、“实例对象都已经从Java堆内存里被回收”和“Class对象没有任何引用”是一个概念么?

2、 “ClassLoader已经被回收”，什么时候会回收?

答:

1、不是，Class对象代表类，如果你有变量引用了类的Class对象，那么就是有引用 2、比如你自定义的ClassLoader，本身就是个对象，一旦他没人再使用了，就会被垃圾回收

问题十三

引用Class对象的是该类的实例对象？还是其他什么?

答:

比如用反射，可以获取一个对象的类的Class对象实例，比如Class clazz = replicaManager.getClass()，就可以通过replicaManager引用的对象，获取到ReplicaManager类的Class对象，那个clazz变量，就可以引用这个Class对象

问题十四

栈帧里的数据出栈就没有了，怎么理解？不是垃圾回收线程来回收，那是main线程执行完这个方法就直接将其栈帧里面的数据销毁了吗?

答:

没错

问题十五

第二周打卡，跟上节奏。今日思考题：项目中托管给Spring管理的对象，带@Configration的配置对象，都是长期存在老年代。自己定义那些pojo对象，如果不被定义为类对象就是朝生夕灭的，所以分配在年轻代里面。

答:

非常好，同学

问题十六

大多数情况下，方法中的对象都是短生存周期的对象，而实例对象和类对象都是长生存周期的对象。

猜测一下：spring容器中管理的singtone的bean是相对长生存周期的对象，而prototype是短生存周期对象

答:

可以多看看你手头负责的系统代码，去分析一下

问题十八

public void load(){ A a = new A(); a这个保存地址的变量是存在虚拟机栈的,这个方法执行完成后就销毁了, 那new A()这个对象是需要等待垃圾回收线程扫描后才回收吗?还是和a这个变量同时回收?

答:

对象要等待垃圾回收才销毁

问题十九

感觉每天篇幅内容太少了，每天都迫不及待期待下一节内容，有点像小时候追看连续剧的感觉

答:

要一步一步来，太快内容太多了，不利于消化总结，每周的文章，建议反复看2~3遍，自己配合作业，总结和梳理，学习+复习+作业，才能让你真正消化掉这个知识

问题二十

可以自己通过参数设置多大的对象直接放到老年代。 || 对于超大对象直接进入老年代这句话、老师可以给举个栗子吗

答:

是的

问题二十一

内存不够才会回收软引用对象，内存空间足够的话，不会回收软引用对象。弱引用不管内存空间够不够，只能撑到下次垃圾回收之前，就被会回收。 || gc回收的是软引用，弱应用和虚引用，关于软引用和弱引用傻傻分不清，这两者有何异同，请指教

答:

没错

问题二十二

大部分是spring容器的对象，spring默认单实例方式加载，这些对象可能会存在老年代中。但是方法内部new出来 对象不会存货太长时间，方法结束，引用消息，对象也会在下一次gc被回收。

答:

没错

问题二十三

类初始化时，变量引用的是new出来的对象，此时变量引用的对象会被实例化到堆内存吗？

答：

会实例化放到堆内存

问题二十四

老师好。我今天使用Java VisualVm看了一下，发现了一个问题，我配置的是-Xms4M -Xmx4M -Xmn2M。应该是年轻代2M老年代2M。

写了一个while循环不断的在方法里创建临时变量对象，但是我发现当内存堆达到3m左右的时候，就发生了Minor GC，堆内存回到了2M，而不是4M的时候，理论上不应该是堆内存满了再Minor GC吗？

然后我老年代里的对象是一个静态的简单对象，这个对象会直接把2M占满嘛？因为每次堆内存都回到2M左右。是我理解的有问题吗？谢谢老师

答：

这个很正常的，因为后续第三周会讲新生代的内存结构，就是其实不是新生代全部占满才minor gc，是里面一块主要的内存区域满了，就minor gc

问题二十五

堆内存3G,给新生代2G,剩下老年代1G,一般是不是新生代设置得比老年代大？一般比例是多少？如果机器16G,还是这个支付环境，那么老年代还是1G，新生代7G好。还是老年代2G,新生代5G更好？

答：

明天会说这个老年代的事儿，别着急

问题二十六

打卡。做项目时候没有关注系统压力，主要是考虑功能怎么现实，然后按时交付测试。以后可以按老师今天这个思路去估算一下系统压力了。

答：

是的，如果没合理估算内存压力，没合理设置jvm内存大小，那么上线之后，可能会发现频繁gc问题，导致系统卡顿，这是jvm优化的第一步，合理估算业务压力，合理设置内存大小

问题二十七

总结:

1. 分析系统压力点在哪里？

2. 压力点的每秒请求数?

3. 每个请求耗时?

4. 每个请求消耗的内存?

5. 整个系统的所有请求重复1-4。

6. 算出部署多少台机器? 每个机器多少内存?

思考题: 平时工作中很少这样预估系统压力，一般我的做法都是部署上去后分配一个堆内存，然后测试时再去监控GC的频率做适当调整。这样做确实很被动，很多时候上线后发现和测试的GC频率差太多，以后试试老师这种估算方法。

这个案例分析的方法很好，虽然很浅，但也能学到东西。希望以后能多多发一些更加全面的案例分析。

答:

案例是一步一步来的，每个案例都针对不同的方面和问题，慢慢来，学完几十个案例，你对jvm的内部原理，参数优化，故障解决，就有全面的掌握了

问题二十八

上次发生内存溢出，我们搞到凌晨5点多，最后我们老大调大了堆内存解决的，说是由于使用过多的静态内部类，有地方引用到无法释放导致的，不过我现在还没有明白为啥??

答:

以后慢慢学习，你也能掌握这种能力

问题二十九

这篇文字最重要的收获是分析处理问题的思路，分解然后一步一步分析处理。赞。

答:

是的，思路非常的重要，按照这个思路来，你们自己也能做jvm内存压力预估，系统上线前，合理设置一个内存大小

问题三十

是不是不应该在高峰期的时候让系统进行垃圾回收，这样会造成STW。老师你们线上系统会考虑在低峰期手动触发垃圾回收么?

答:

建议不要手动触发，依托合理的内存设置以及参数优化，让系统自行运转

问题三十一

是不是应该通过老师说的估算方式，尽量设大新生代 ，让系统在高峰期不产生gc?

答:

是的，尽量是这样

问题三十二

老师，那不管三七二十一，在内存大的条件下，多分配给新生代就好了，如果不行就加内存??

答:

那你就浪费机器资源，要合理评估，不需要大内存，就用小内存就可以了

问题三十三

1、支付系统高分期需要处理的请求是是不是应该这么算：100万 / （24 * 3600） ≈ 12，根据28法则，大部分请求发生在中午12点到13点以及晚上的18点到19点，所以 80万请求 / （2 * 3600） ≈ 111，即算出如果单台每秒大概是100多个请求

2、还有就是在完整的支付系统内存占用需要进行预估中，你提到“可以把之前的计算结果扩大10倍~20倍。也就是说每秒除了内存里创建的支付订单对象还创建数十种对象” 这里如果要计算的话 之前的计算结果是 30 * 500字节 * 20倍 = 300000字节=300KB 是这么算吗?

答:

没错的，这是大致估算的方法

问题三十四

老师 您这儿的案例中提到，一个支付请求需要1s中，30个请求也是1s钟，那是不是可以理解为开了30个线程同时并发处理支付请求入库?

答:

没错，就是这个意思

问题三十五

付订单对象，还创建其他数十种对象。那么计算的方式是不是：30 * 500字节/个 * 放大20倍≈300000字节≈300KB ？ ||

1、支付系统高分期需要处理的请求是是不是应该这么算：100万 / （24 * 3600） ≈ 12，根据28法则，大部分请求发生在中午12点到13点以及晚上的18点到19点，所以 80万请求 / （2 * 3600） ≈ 111，即算出如果单台每秒大概是100多个请求

2、还有就是在完整的支付系统内存占用需要进行预估中，你提到“可以把之前的计算结果扩大10倍~20倍。也就是说每秒除了内存里创建的支付订单对象还创建数十种对象” 这里如果要计算的话 之前的计算结果是 30 * 500字节 * 20倍 = 300000字节=300KB 是这么算吗?

答

是的，你学会大致估算系统内存压力的办法了

问题三十六

“可能你每秒过来的1000笔交易，不再是1秒就可以处理完毕了，因为压力骤增，会导致你的系统性能下降，可能偶尔会出现每个请求处理完毕需要几秒钟”:老师，这里说的压力骤增是磁盘读写压力吗还是内存CPU压力，出行每个请求处理完毕需要几秒这里是写入压力吗?与网络有关吗?谢谢

答

都有可能，主要是CPU负载过高，会导致高并发下每个请求的处理性能直线下降，还有网络问题也会有

问题三十七

我们订单一天二百多万，线上正常每秒产生也应该在1M以上，xmn2048，xmx8192,本来半个多小时一次minor gc,扩大一百倍，不到一分钟一次，应该会出现案例中的问题，老年代会频繁gc,不过我们有6g老年代，达到full gc应该时间会稍微长点

答

对的，自己分析的非常好，掌握这个方法了

问题三十八

我现在的业务遇到的瓶颈是带宽，压测的时候，请求量暴增百倍，上下行带宽瞬间被打满，导致部分业务直接不能用了。。直到慢慢恢复过来

答

是的，内存、网络带宽、磁盘IO、数据库，都是系统的瓶颈

问题三十九

老师, 可以说下, 为什么并发上来了, 压力就会剧增嘛? 哪些方面的压力.

答

并发上来之后，内存、网络带宽、磁盘IO、数据库，都是系统的瓶颈，比如网络带宽打满，你的请求就会排队卡住，磁盘IO变满，数据库性能下降，都会导致请求处理慢几十倍

问题四十

您好，我有一个问题，就是main函数中创建了对象，这个对象在堆中开启空间，那么如果这个对象中有成员变量，这个成员变量是存在哪里？成员变量的引用存在哪里

答

成员变量也是在堆内存里的

问题四十一

看了老师的专栏我觉得讲的非常清晰易懂，学到很多。但是，我这也有一个建议。学习的过程离不开实践，我在想老师能不能提供一个让我们能够模拟出一些问题的方式，然后通过设置jvm的内存分布来解决这个问题，这样会不会更好一些?

答

放心，现在你们还学的不够深入，后续大量的案例环节，会给很多示范代码，模拟故障，让你们动手去操作的

问题四十二

老师我上网查了一下资料，把问题弄明白了。Test.class是被加载了，但是并没有 执行初始化步骤。

课程中提到了类加载的时机，但是没有提到类初始化的时机，我把一直理解类的 加载->验证->准备->解析->初始化是一个连续的动作，以为类一旦加载必定 会立即初始化。

补充类初始化的时机如下：

1.当创建某个类的新实例时（如通过new或者反射，克隆，反序列化等）

2.当调用某个类的静态方法时

3.当使用某个类或接口的静态字段时

4.调用Java API中的某些反射方法时，比如类Class中的方法，或者java.lang.reflect中的类的方法时

5.当初始化某个子类时

6.当虚拟机启动某个被标明为启动类的类（即包含main方法的那个类） 所以System.ou.println(Test.class)不满足上面6种情况，也就没有做初始化

答

对的，就是这样

问题四十三

老师 假如Kafka类里面 声明一个实例变量 private ReplicaFetcher = new ... 这个实例变量放在哪个区

答

实例变量就在堆内存里

问题四十四

很感谢这个平台，这次出去面了3家offer都拿到了，换作以前，是不可能的事情

答

加油，坚持学习，每天学习

问题四十五

老师 根据示例代码， 我做了以下jvm参数配置：-Xms10m -Xmx10m， 然后在visualVM里跟踪堆栈使用情况。 十分诧异的现象是： 在while true循环中，也就是执行fetchFromRemote的时候， 新生代大小一直在有规律的增长，然后不停的minor GC， 每次GC(而不是等到15次以后)，老年代都会相应的增长一点。

我的问题是，使新生代增长的到底是什么对象？GC时又是什么对象跑到老年代里去了？ 按我的理解，fetcher对象应该有且只有一份实例，而且while循环中，不会生成新的对象， 最初，新生代里有一份fetcher，然后第16次minor GC的时候，fetcher被转移到老年代， 无论如何，新生代和老年代都不会不断增长。 所以，是不是有什么我不知道的对象混了进去，导致新生代不断增长？

答

新生代到老年代转移的机制不只是年龄一种，还有别的，下周会详细说的

问题四十六

老师，多谢回复，每个订单处理时间是1秒和10秒，10秒的就要比1秒的要多加内存吗？请问是怎样的逻辑？能否量化？

答

那你得计算一下，你的内存每秒被使用的速度，根据这个来规划内存大小，还有你要是10秒一个请求，可能内存里累计起来会有大量对象没法释放，会导致瞬间新生代被打满，而且大量对象没法回收，然后全部去老年代，然后老年代也很快就满了，最后内存不够，很快就内存溢出了

问题四十七

生活所迫，比较功利，我这套课程如果能达到老师的预期，我出去找工作要多少合适，我现在20K左右

答

跳槽不是一套课程就能搞定的，毕竟这就是一个88的专栏，能帮你搞定面试中的一个技术点而已，如果你要跳槽找好工作，需要系统训练，后续我们会出更多的课程的