了JVM 些 典 ，分享出 ， 助， 你 不 ， 出， 会

1.什么情况下会发生栈内存溢出。

思路: 义，再 为什么会 出，再 下 关 ，OK 以 写 个 出 demo

我的答案:

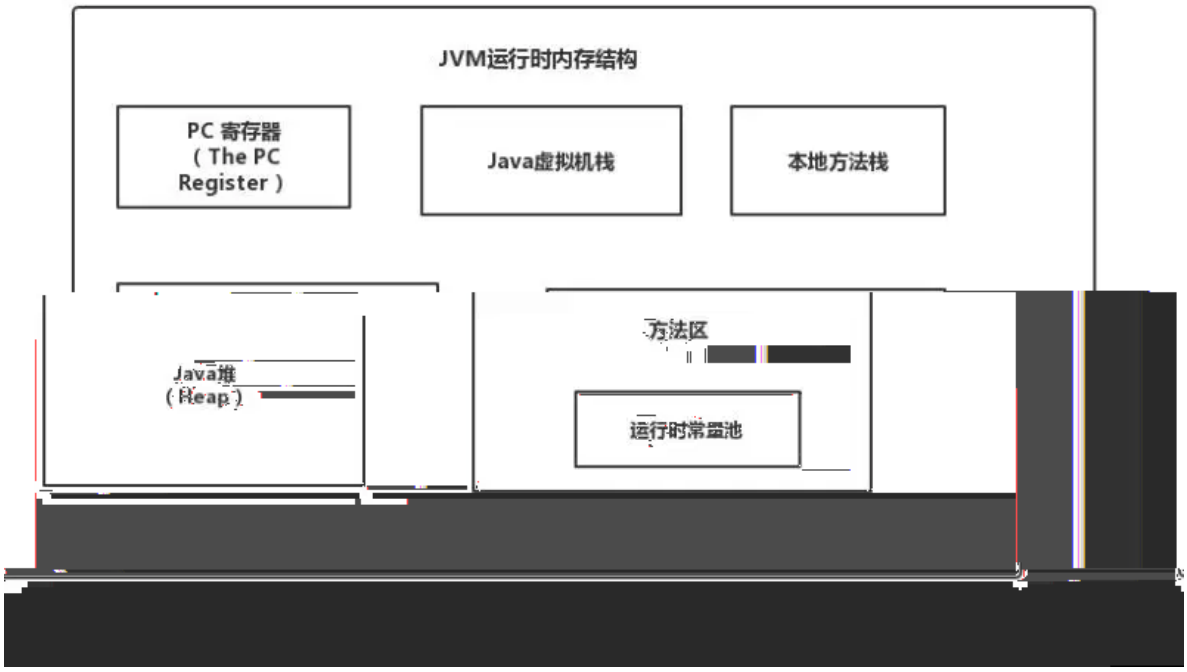
- ，他 与 ， 个 候 会创 个 ， 储 ， 作 ，动 ， 出 信 包 ，
- 于 允 ， 出StackOverflowError ， 产
- Java 以动 ， 且 动作 ，但 到 内 ， 候 内 创 ， 么Java 出 个OutOfMemory (动)
- -Xss JVM

2.详解JVM内存模型

思路: 下JVM内 ， 个 义，作 ，以 会 ， 出

我的答案:

- JVM内



: 前 , 于 令 ,

Java : 出 ,

Native : 似, 不 务于Native ,

Java : java内 , 例 java , GC , 共享

区: 加 信 即 代 (即 久),
主 , 共享

3.JVM内存为什么要分成新生代，老年代，持久代。新生代中为什么要分为Eden和Survivor。

思路: 先 下JAVA , 代 划分，再 们之 化, 互之 些 (: –XX:NewRatio，–XX:SurvivorRatio ），再 为什么 划分, 加

我的答案:

1）共享内存区划分

- 共享内 区 = 久 +
- 久 = 区 + 其他
- Java = 代 + 代
- 代 = Eden + S0 + S1

2）一些参数的配置

- , 代 (Young) 与 代 (Old) 例 值为 1:2 , 以 –XX:NewRatio
- , Edem : from : to = 8 : 1 : 1 (以 –XX:SurvivorRatio)
- Survivor区中 制 为15(-XX:+MaxTenuringThreshold)

3)为什么要分为Eden和Survivor?为什么要设置两个Survivor区?

- Survivor，Eden区 Minor GC， 会 到 代 代 ， Major GC. 代 内 于 代， Full GC Minor GC ，以 分为Eden Survivor
- Survivor 义， 减 到 代 ， 减 Full GC ，Survivor 保 ， 16 Minor GC 代中 ， 会 到 代
- 两个Survivor区 决了 化，刚刚 Eden中， Minor GC，Eden中 会 动到 survivor space S0，Eden ； Eden区再 了， 再 Minor GC，Eden S0中 会 制 入 二 survivor space S1（ 个 ， 为 制 保 了S1中 S0 Eden两 分 占 内 ， 免了 化 ）

4. JVM中一次完整的GC流程是怎样的，对象如何晋升到老年代

思路： 先 下Java 内 划分，再 Minor GC，Major GC，full GC， 们之 化

我的答案：

- Java = 代 + 代
- 代 = Eden + S0 + S1
- Eden 区 了，Java 会 Minor GC，以 代 ， 下 ，则会 到 Survivor区
- **大对象**（ 内 Java ， 串）**直接进入老年态**；
- Eden出 ， Minor GC 仍 ， 且 Survivor ， 为1， Minor GC， +1，**若年龄超过一定限制（15），则被晋升到老年态** 即**长期存活的对象进入老年态**
- 代 了 **无法容纳更多的对象**，Minor GC 之 会 Full GC，Full GC 个内 – **包括年轻代和年老代**
- Major GC **发生在老年代的GC**，**清理老年区**， 会伴 Minor GC，**比Minor GC慢10倍以上**

5.你知道哪几种垃圾收集器，各自的优缺点，重点讲下cms和G1，包括原理，流程，优缺点。

思路： 住典 ， 其cms G1， 们 与区别，

我的答案：

1）几种垃圾收集器：

- **Serial收集器：** 单 ， ， stop the world，使 制
- **ParNew收集器：** Serial ，也 stop the world， 制
- **Parallel Scavenge收集器：** 代 ， 制 ， ， 到 个 共 100分 ，其中 1分 ， 99%
- **Serial Old收集器：** Serial 代 ，单 ，使
- **Parallel Old收集器：** Parallel Scavenge 代 ，使 ， -
- **CMS(Concurrent Mark Sweep) 收集器：** 以 停 为 ，**标记清除算法，运作过程：初始标记，并发标记，重新标记，并发清除**， 会产

- **G1收集器：** ，**运作流程主要包括以下：初始标记，并发标记，最终标记，筛选标记** 不会产 ， 以 制停

2）CMS收集器和G1收集器的区别：

- CMS 代 ， 以 代 Serial ParNew 使 ；
- G1 代 代，不 其他 使 ；

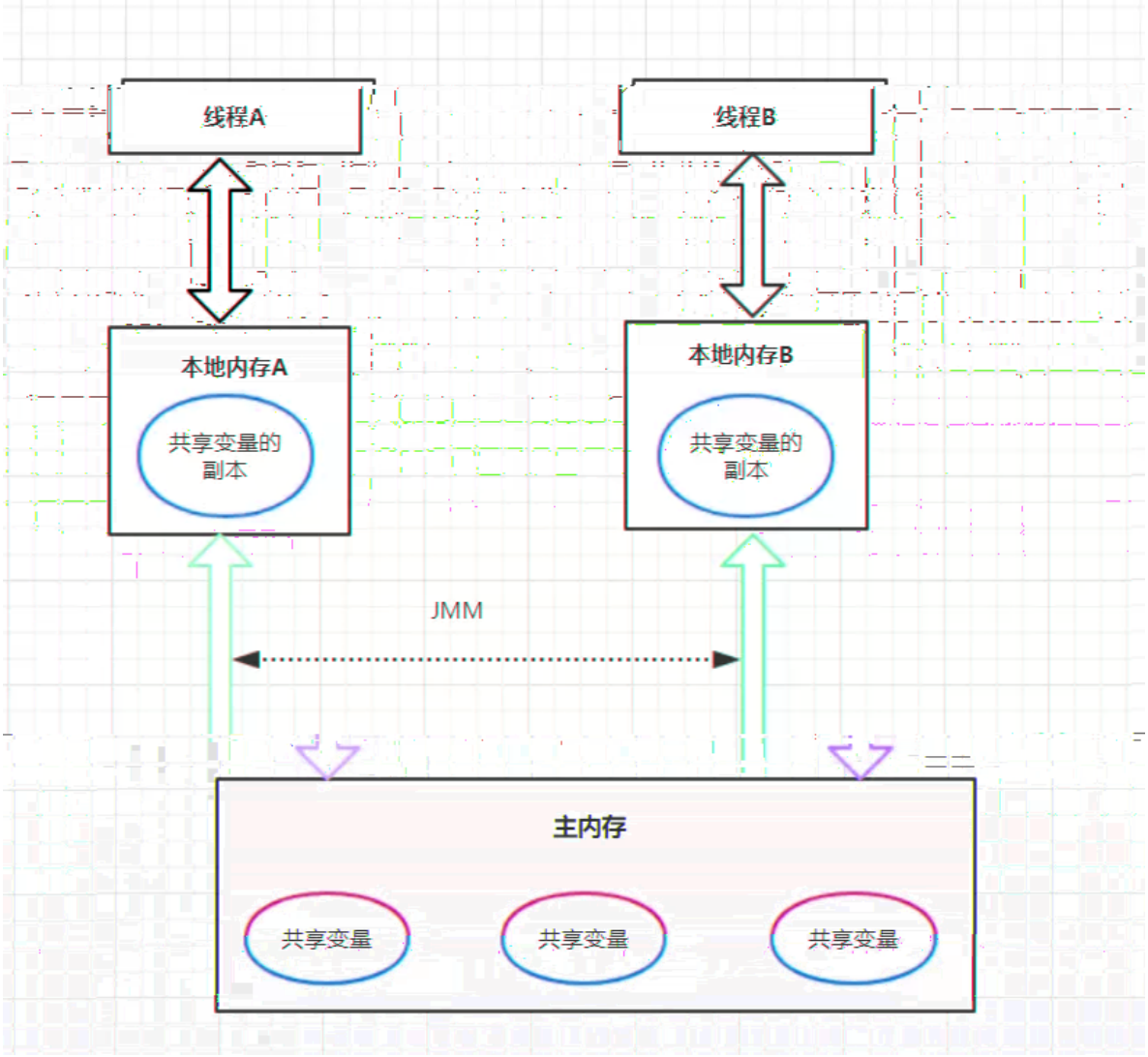
- CMS 以 停 为 ；
- G1 停
- CMS 使 “ - ” , 产 内
- G1 使 “ - ” , 了 , 低了内

6.JVM内存模型的相关知识了解多少，比如重排序，内存屏障，happen-before，主内存，工作内存。

思路： 先 出Java内 , 例 volatile , 什么 , 内 , 写 以下demo

我的答案：

1）Java内存模型图：



Java内 了 **变量都存储在主内存**中，**线程还有自己的工作内存**，作内 中保 了 中 到 主内 副 ，**线程对变量的所有操作都必须在工作内存中** ，**而不能直接读写主内存** 不 之 也**无法直接访问对方工作内存中的变量**，传 作内 主 之

2）指令重排序。

，先 代

```
public class PossibleReordering {
static int x = 0, y = 0;
static int a = 0, b = 0;

public static void main(String[] args) throws InterruptedException {
 Thread one = new Thread(new Runnable() { public void run() { a = 1; x = b; }
});
 Thread other = new Thread(new Runnable() { public void run() { b = 1; y = a; }
}); one.start();other.start(); one.join();other.join(); System.out.println("(" +
x + "," + y + ")");}
```

为(1,0) (0,1) (1,1)，也 (0,0) 为， ，代 令 不 严 代 代 会 令乱 (out-of-order execution，OoOE OOE) ， 件允 况下， 前 力 即 令，下 令 3 乱 ， 以

指令重排

3）内存屏障

内存屏障，也 内 ， CPU 令， 于 制 件下 内

- **LoadLoad屏障**： 于 Load1; LoadLoad; Load2， Load2 作 前，保 Load1
- **StoreStore屏障**： 于 Store1; StoreStore; Store2， Store2 写入 作 前，保 Store1 写入 作 其
- **LoadStore屏障**： 于 Load1; LoadStore; Store2， Store2 写入 作 刷出 前，保 Load1
- **StoreLoad屏障**： 于 Store1; StoreLoad; Load2， Load2 作 前，保 Store1 写入 中 中， 个 个万 ，兼具其 三 内 功

4）happen-before原则

- **单线程happen-before原则**： 个 中，书写 前 作happen-before 作 happen-before 则： 个 unlock 作happen-before lock 作
- **volatile的happen-before原则**： 个volatile 写 作happen-before 任作(也包 写 作了)
- **happen-before的传递性原则**： A 作 happen-before B 作，B 作happen-before C 作， 么A 作happen-before C 作
- **线程启动的happen-before原则**： 个 start happen-before 其
- **线程中断的happen-before原则**： interrupt happen-before 中到中 代
- **线程终结的happen-before原则**： 中 作 happen-before
- **对象创建的happen-before原则**： 个 初 化 先于他 finalize

7.简单说说你了解的类加载器，可以打破双亲委派么，怎么打破。

思路： 先　　下什么　　加　　，　以　　个　，再　　下　加　　义，　　下　亲　　，　　么　　亲

我的答案：

1) 什么是类加载器?

类加载器　　全　　class　件加　到JVM内　，　为Class

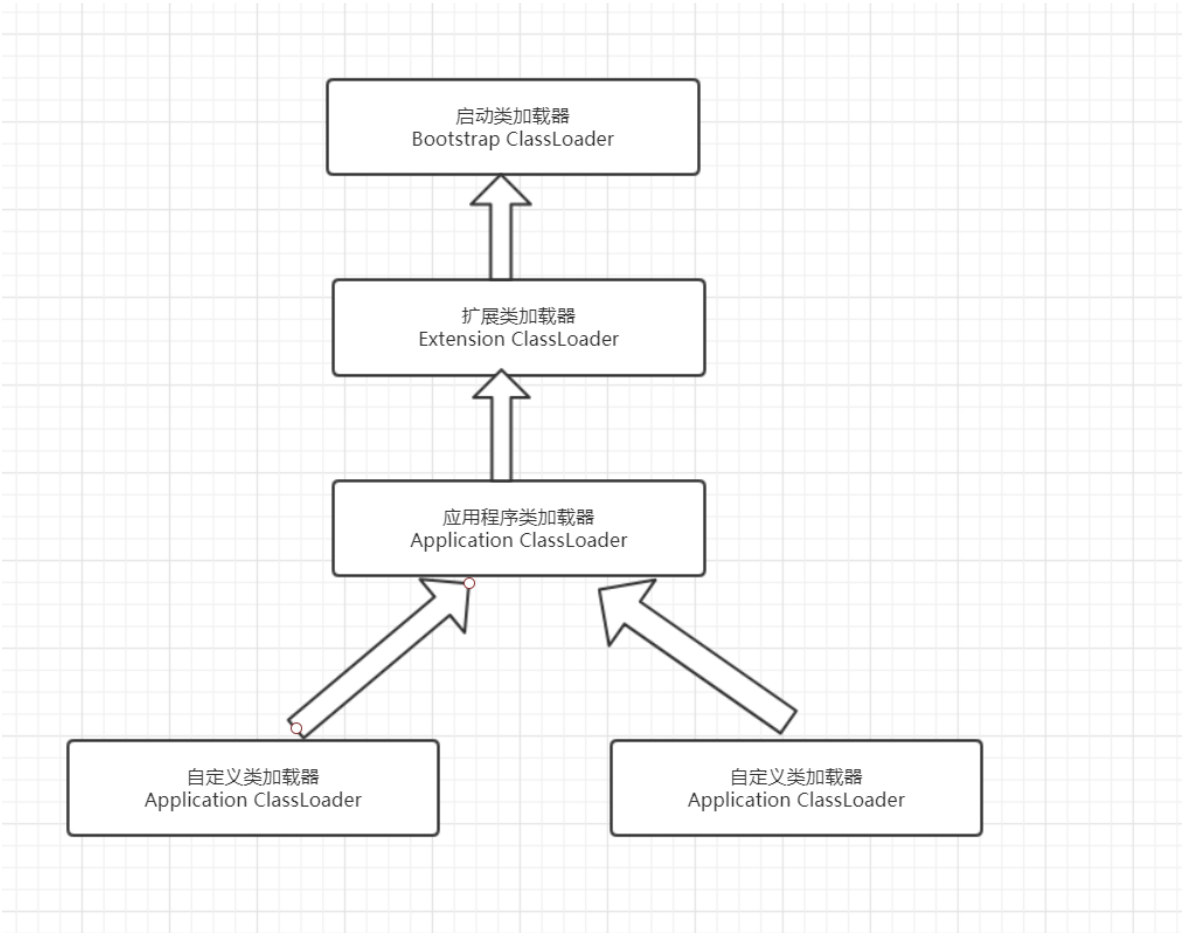
> - 　动　加　　（Bootstrap ClassLoader）：　C++　　（　　HotSpot），<JAVA_HOME>\lib　　-Xbootclasspath　　中　　加　到内　中
> - 其他　加　　：　Java　　，　　ClassLoader　　：
>
>> - 　　加　　（Extension ClassLoader）：　　加　<JAVA_HOME>\lib\ext　java.ext.dirs　　中
>> - 　　加　　（Application ClassLoader）　　加　　（classpath）上　，　们　以　　使　　个　加　　况，　　们　　义　加　　个加

2）双亲委派模型

双亲委派模型工作过程是：

> 个　加　　到　加　　，　　先不会　　加　　个　，　　个　加　　个　加　　，　　加　　内　不到　(即ClassNotFoundException），　加　　会　　加

亲　　：



3）为什么需要双亲委派模型?

，先 下， 亲 ， 么 不 以**自己定义一个java.lang.Object的同名类，java.lang.String的同名类，** 到ClassPath中， 么**类之间的比较结果及类的唯一性将无法保证，** ，为什么 亲 ？**防止内存中出现多份同样的字节码**

4）怎么打破双亲委派模型?

亲 制则不仅**要继承ClassLoader** ， **重写loadClass和findClass**

8.说说你知道的几种主要的JVM参数

思路: 以 下 关 ， 关 ， 下 助信 关

我的答案:

1）堆栈配置相关

```
java -Xmx3550m -Xms3550m -Xmn2g -Xss128k
-XX:MaxPermSize=16m -XX:NewRatio=4 -XX:SurvivorRatio=4 -
XX:MaxTenuringThreshold=0
```

-Xmx3550m: 为3550m

-Xms3550m: 初 为3550m

-Xmn2g: 代 为2g

-Xss128k: 个 为128k

-XX:MaxPermSize: 久代 为16m

-XX:NewRatio=4: 代（包 Eden 两个Survivor区）与 代 值（ 久代）

-XX:SurvivorRatio=4: 代中Eden区与Survivor区 值 为4，则两个Survivor区与 个Eden区 值为2:4， 个Survivor区占 个 代 1/6

-XX:MaxTenuringThreshold=0: 为0 ，则 代 不 Survivor区， 入 代

2）垃圾收集器相关

```
-XX:+UseParallelGC
-XX:ParallelGCThreads=20
-XX:+UseConcMarkSweepGC
-XX:CMSFullGCsBeforeCompaction=5
-XX:+UseCMSCompactAtFullCollection:
```

-XX:+UseParallelGC: 为

-XX:ParallelGCThreads=20:

-XX:+UseConcMarkSweepGC: 代为

-XX:CMSFullGCsBeforeCompaction: 于 不 内 ， 以 以 会产 “ ”，使 低 值 GC以 内

-XX:+UseCMSCompactAtFullCollection: 代 会 ，但 以

3）辅助信息相关

```
-XX:+PrintGC
-XX:+PrintGCDetails
```

-XX:+PrintGC 输出形式:

[GC 118250K->113543K(130112K), 0.0094143 secs] [Full GC 121376K->10414K(130112K), 0.0650971 secs]

-XX:+PrintGCDetails 输出形式:

[GC [DefNew: 8614K->781K(9088K), 0.0123035 secs] 118250K->113543K(130112K), 0.0124633 secs] [GC [DefNew: 8614K->8614K(9088K), 0.0000665 secs][Tenured: 112761K->10414K(121024K), 0.0433488 secs] 121376K->10414K(130112K), 0.0436268 secs

9.怎么打出线程栈信息。

思路: 以 下jps，top ，jstack 几个 令，再 上

我的答案:

- 入jps，
- top -Hp pid 中 CPU
- jstack pid 令 前java
- jstack -l > /tmp/output.txt 信 到 个txt 件
- 以使 fastthread 位，fastthread.io/

10.强引用、软引用、弱引用、虚引用的区别?

思路: 先 下 义, 以 代 下,也 以 到ThreadLocalMap

我的答案:

1）强引用

们 new了 个 ,例 Object obj = new Object();即使 内 不 况下,JVM 出OutOfMemory 也不会

2）软引用

个 具 ,则内 , 不会 ; 内 不 了, 会 些 内

```
SoftReference<String> softRef=new SoftReference<String>(str);     // 软引用
```

用处: 中 ,例 , 个 内 从 中 出 ? 具体 了

(1) 个 内 ,则 前 ,

(2) 储到内 中会 内 , 会 内 出

下代 :

```
Browser prev = new Browser();                 // 获取页面进行浏览
SoftReference sr = new SoftReference(prev); // 浏览完毕后置为软引用
if(sr.get()!=null){
rev = (Browser) sr.get();           // 还没有被回收器回收，直接获取
}else{
 prev = new Browser();                // 由于内存吃紧，所以对软引用的对象回收了
    sr = new SoftReference(prev);        // 重新构建
}
```

3）弱引用

具 内 区 中,
了 具 ,不 前内 与 , 会 内

```
String str=new String("abc");
WeakReference<String> abcWeakRef = new WeakReference<String>(str);
str=null;
等价于
str = null;
System.gc();
```

4）虚引用

个 仅 ， 么 任何 ， 任何 候
主 动