1.jvm那些内存会oom?

从上面分析的Java数据区来看，除了程序计数器不会发生OOM外，哪些区域会发生OOM的情况呢？

堆内存，堆内存不足是最常见的发送OOM的原因之一。

如果在堆中没有内存完成对象实例的分配，并且堆无法再扩展时，将抛出OutOfMemoryError异常，抛出的错误信息是“java.lang.OutOfMemoryError:Java heap space”。

当前主流的JVM可以通过-Xmx和-Xms来控制堆内存的大小，发生堆上OOM的可能是存在内存泄露，也可能是堆大小分配不合理。

Java虚拟机栈和本地方法栈，这两个区域的区别不过是虚拟机栈为虚拟机执行Java方法服务，而本地方法栈则为虚拟机使用到的Native方法服务，在内存分配异常上是相同的。

在JVM规范中，对Java虚拟机栈规定了两种异常：

a. 如果线程请求的栈大于所分配的栈大小，则抛出StackOverFlowError错误，比如进行了一个不会停止的递归调用；

b. 如果虚拟机栈是可以动态拓展的，拓展时无法申请到足够的内存，则抛出OutOfMemoryError错误。

直接内存：直接内存虽然不是虚拟机运行时数据区的一部分，但既然是内存，就会受到物理内存的限制。在JDK1.4中引入的NIO使用Native函数库在堆外内存上直接分配内存，但直接内存不足时，也会导致OOM。

方法区：随着Metaspace元数据区的引入，方法区的OOM错误信息也变成了“java.lang.OutOfMemoryError:Metaspace”

对于旧版本的Oracle JDK，由于永久代的大小有限，而JVM对永久代的垃圾回收并不积极，如果往永久代不断写入数据，例如String.Intern()的调用，在永久代占用太多空间导致内存不足，也会出现OOM的问题，对应的错误信息为“java.lang.OutOfMemoryError:PermGen space”

2.java heap oom之后能捕获，stackoverflow,oom？

1. oom异常一般是java程序在做内存分配时，发现没有足够的剩余空间可用而抛出的异常；

2. 此时的分配空间可能是出于代码的new操作（用户主动），可能是出于内存的复制操作（语言自动），也可能是出于内存数据的重振操作（语言自动），可能是出jvm检测到外部信号（jvm自动）；

3. oom只是被建议为不要捕获的异常，因为通常你对这种情况是无能为力的！但你如果实在要捕获，why not ？

4. oom一般只会影响当前线程，而jvm中只要存在一个非daemon线程在运行，jvm就不会退出；

5. 如果是线程池运行环境，一般需要一个统一管理oom的程序，否则不能及时统一处理oom；

stackoverflow：

每当java程序启动一个新的线程时，java虚拟机会为他分配一个栈，java栈以帧为单位保持线程运行状态；当线程调用一个方法是，jvm压入一个新的栈帧到这个线程的栈中，只要这个方法还没返回，这个栈帧就存在。

如果方法的嵌套调用层次太多(如递归调用),随着java栈中的帧的增多，最终导致这个线程的栈中的所有栈帧的大小的总和大于-Xss设置的值，而产生生StackOverflowError溢出异常。

2.mybatis #{} 会预防sql注入?

${}只是变量替换，不会预防sql注入

3类的动态代理在那个阶段初始化的

类加载阶段，所谓的动态代理就是想办法根据接口或者目标对象计算出代理类的字节码然后加载进 JVM 中。

实际计算的情况会很复杂，我们借助一些诸如 JDK 动态代理实现、CGLIB 第三方库来完成的

4子类隐藏父类的静态方法，父类引用调用执行

调用的父类的方法

5.调用栈存放了那些信息？

栈帧：前面说了，当线程执行到某个方法时就会往方线程栈中压入一个帧，称为栈帧，栈帧中包含了方法的局部变量表、操作数栈、返回地址、动态连接等信息

6.FileNotFoundException是敏感异常？

是

7.try-with-resources语句，catch，finally，是否按照顺序关闭资源？

try定义的变量，catch和finally不可见。

try( FileInputStream input = new FileInputStream("file.txt");

BufferedInputStream bufferedInput = new BufferedInputStream(input)

) {

int data = bufferedInput.read();

while(data != -1){

System.out.print((char) data);

data = bufferedInput.read();

}

}

这个例子在括号中创建了两个资源，一个FileInputStream和一个BufferedInputStream。

当执行完 try 块时，这两个块都将被关闭。

这些资源将按照他们在括号内创建/列出顺序相反的顺序关闭，也就是先关闭BufferedInputStream，然后关闭FileInputStream。

8.jstat,jinfo,jmap,jstack那个能查看gc?

jstat

9. char a = '\n';?

10.

List<String> aaa = new ArrayList<>();

aaa.add("1111");

aaa.add(2,"2222");

System.out.println("aaa.get(1)"+aaa.get(1));

Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 2, Size: 1

at java.util.ArrayList.rangeCheckForAdd(ArrayList.java:661)

at java.util.ArrayList.add(ArrayList.java:473)

at test.Main1.main(Main1.java:28)

11. java stream api的使用

int [] aaa = {1,2,3};

Set set = new HashSet();

set.add(Arrays.asList(aaa));

System.out.println(aaa);

https://blog.csdn.net/qq\_28410283/article/details/80633710

12.类加载反射的题

13.URL和url的题，我选的是去除.. .

14. intterupt()使用

MyThread myThread = new MyThread();

myThread.start();

Thread.sleep(2000);

myThread.interrupt();

}

static class MyThread extends Thread {

@Override

public void run() {

while (true) {

if (Thread.interrupted()) {

System.out.println("thread is interrupted");

}

try {

Thread.sleep(3000);

} catch (InterruptedException e) {

System.out.println("sleep is interrupted");

}

}

}

}

15 线程池抛异常，除零操作，问执行;