内存泄露与内存溢出的区别和联系：

1.内存泄露memory leak：是指程序在申请内存后，无法释放已经申请的内存空间，一次内存泄露似乎不会有很大影响，但内存泄露堆积后的结果就是内存溢出。

2.内存溢出out of memory ：是指程序申请内存时，没有足够的内存供申请者使用，或者说，给了你一块存储int类型数据的存储空间，但是你却存储了long类型的数据，那么结果就是内存不够用，此时就会出现内存溢出。

二者之间的关系：

1.内存泄露的堆积会导致内存溢出。比如申请分配内存进行使用，可是使用完了以后却不归还，结果你申请的那块内存自己也不能访问，而系统也不能再次将它分配给需要的程序。

2.内存溢出就是你要的内存空间超过了系统实际分配你的空间，此时系统相当于没有满足你的要求，就会报内存溢出的错误。

内存泄露的分类：

1.常发性内存泄露：发生泄露的代码会被多次执行到，每次被执行的时候都会导致一块内存泄露。

2.偶发性内存泄露：发生内存泄露的代码只有在某些特定环境或者操作过程下才会发生。常发性和偶发性是相对的。对于特定的环境，偶发性的也许就变成了常发性的。

3.一次性内存泄露：发生内存泄露的代码只会被执行一次，或者是由于算法上的缺陷，导致总会有一块仅且一块内存发生泄露。

4.隐式内存泄露：程序在运行过程中不停的分配内存，但是直到结束的时候才释放内存。严格说这里没有发生内存泄露，因为最终程序释放了所有申请的内存，但是有些程序需要执行很久，不及时释放内存很有可能最终耗尽系统的所有内存。

内存溢出的原因：

1.内存中加载的数据量过大，如一次从数据库中取出数据过多。

2.集合类中有对对象的引用，使用完后未清空，使得jvm不能回收。

3.代码中存在死循环或者循环产生过多的重复对象实体。

4.启动参数内存值设定得过小。

检查对数据库查询中，是否有一次获得全部数据的查询。一般来说，如果一次取十万条记录到内存，就可能引起内存溢出。

栈内存溢出：

stackOverFlowError错误，出现此种情况是因为方法运行的时候栈的深度超过了虚拟机容许的最大深度所致。出现这种情况，一般情况下是程序错误所致，比如写了一个死递归，就有可能造成此种情况。根据<java 虚拟机规范>中文版，如果线程请求的栈数量超过栈允许的最大容量的话，java虚拟机将抛出一个StackOverflow异常，如果java虚拟机栈可以动态扩展，并且

内存溢出的排查：
1.是否类中或引用变量过多的使用了static修饰，如public static student s，在类中的属性中使用static修饰最好只用基本类型或字符串。如public static int i = 0; //public static String str; 2.是否app中使用了大量的递归或无限递归(递归中用到了大量的新建的对象)。3.是否app中使用了大量循环或者死循环