**内存溢出 out of memory，是指程序在申请内存时，没有足够的内存空间供其使用，出现out of memory**；

内存泄露 **memory leak**，是指程序在申请内存后，无法释放已申请的内存空间，一次内存泄露危害可以忽略，但内存泄露堆积后果很严重，无论多少内存,迟早会被占光。

内存泄漏最终会导致内存溢出。

Java内存泄漏的主要原因：**长生命周期的对象持有短生命周期对象的引用就很可能发生内存泄漏，尽管短生命周期对象已经不再需要，但是因为长生命周期对象持有它的引用导致不能被回收。**

1. 静态集合类引起的内存泄漏

像HashMap,Vector等的使用最容易出现内存泄漏，这些静态变量的生命周期和应用程序一致，他们所引用的所有对象Object也不能被释放。

1. **当集合里面对象的属性被修改后，再调用remove（）方法时不起作用。**
2. 监听器 在删除对象的时候没有删除这些监听器，从而增加了内存泄漏的机会。
3. 各种连接

**比如数据库连接，网络连接和io连接，除非其显式的调用了其close（）方法将其连接关闭，否则是不会被自动gc回收的。**

1. 单例模式

如果单例对象持有外部对象的引用，那么这个外部对象将不能被jvm正常回收，导致内存泄露。不正确使用单例模式是引起内存泄露的一个常见问题，单例对象在被初始化后将在JVM的整个生命周期中存在（以静态变量的方式），如果单例对象持有外部对象的引用，那么这个外部对象将不能被jvm正常回收，导致内存泄露，考虑下面的例子：