在Java虚拟机规范中，定义了这么两种异常：StackOverflowError与OutOfMemoryError。

那么它们到底直接有啥区别呢？

在《The Java ® Virtual Machine Specification Java SE 8 Edition》中是这么说的：

The following exceptional conditions are associated with native method stacks:

If the computation in a thread requires a larger native method stack than is permitted, the Java Virtual Machine throws a StackOverflowError .

If native method stacks can be dynamically expanded and native method stack expansion is attempted but insufficient memory can be made available, or if insufficient memory can be made available to create the initial native method stack for a new thread, the Java Virtual Machine throws an OutOfMemoryError .

如果在一个线程计算过程中不允许有更大的本地方法栈，那么JVM就抛出StackOverflowError

如果本地方法栈可以动态地扩展，并且本地方法栈尝试过扩展了，但是没有足够的内容分配给它，再或者没有足够的内存为线程建立初始化本地方法栈，那么JVM抛出的就是OutOfMemoryError。

先看一下StackOverflowError的情况：

public class StackOverFlow {

public int stackSize = 0;

public void stackIncre() {

stackSize++;

stackIncre();

}

public static void main(String[] args) throws Throwable{

StackOverFlow sof = new StackOverFlow();

try {

sof.stackIncre();

} catch (Throwable e) {

System.out.println(sof.stackSize);

throw e;

}

}

}

以上代码逻辑很简单，就是无限制递归，直到堆栈溢出，运行输出如下：

16896

Exception in thread "main" java.lang.StackOverflowError

at StackOverFlow.stackIncre(StackOverFlow.java:9)

at StackOverFlow.stackIncre(StackOverFlow.java:9)

at StackOverFlow.stackIncre(StackOverFlow.java:9)

at StackOverFlow.stackIncre(StackOverFlow.java:9)

at StackOverFlow.stackIncre(StackOverFlow.java:9)

..........

可以看出运行16896次的时候，堆栈就扛不住了，抛出StackOverflowError异常。

再比如有这么一段代码：

public class HeapOverFlow {

public static void main(String[] args) {

ArrayList<HeapOverFlow> list = new ArrayList<HeapOverFlow>();

while (true) {

list.add(new HeapOverFlow());

}

}

}

代码逻辑很简单，就是无限制地往list里面塞对象，直到它扛不住：

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space

at java.util.Arrays.copyOf(Arrays.java:3210)

at java.util.Arrays.copyOf(Arrays.java:3181)

at java.util.ArrayList.grow(ArrayList.java:261)

at java.util.ArrayList.ensureExplicitCapacity(ArrayList.java:235)

at java.util.ArrayList.ensureCapacityInternal(ArrayList.java:227)

at java.util.ArrayList.add(ArrayList.java:458)

at HeapOverFlow.main(HeapOverFlow.java:11)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at com.intellij.rt.execution.application.AppMain.main(AppMain.java:144)

这就是上面所说的，我们都知道List是动态增长的么，它不够了就要扩展，不够了就要扩展，但是机器本地内存还是优先的，要是扩展不了了，就抛出OutOfMemoryError。