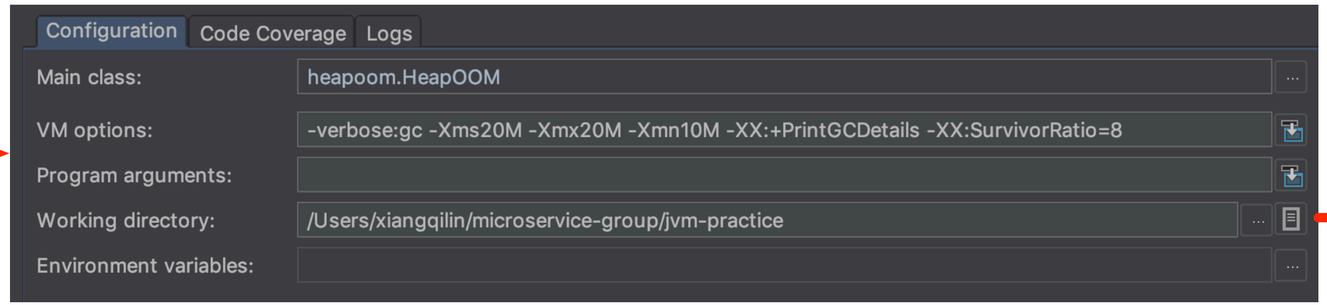
模拟Heap上的OOM，不断创建对象，并且GC Root到对象可达（避免垃圾回收），这样堆内存达到最大之后抛出OutOfMemoryError。

```java
import java.util.ArrayList;
import java.util.List;

public class HeapOOM {
    static class OOMObject {

    }

    public static void main(String[] args) {
        List<OOMObject> list = new ArrayList<>();
        while (true) {
            list.add(new OOMObject());
        }
    }
}
```

Configure

-verbose:gc 打印每次GC的情况
–Xms20m JVM初始堆大小为20m
–Xmx20m JVM最大堆大小为20m
–Xmn10m 年轻代为10m（Eden+2 survivor）
–XX:+PrintGCDetails 打印每次GC的详细消息
–XX:SurvivorRatio=8 Eden和Survivor的比例为8:1。

–Xms与Xmx相同，以避免自动扩展。
–Xmn与jmap –heap不同（Eden+1 survivor），



Run

[Times: user=0.00 sys=0.00, real=0.00 secs]
user代表进程在用户态消耗的CPU时间
sys代表代表进程在内核态消耗的CPU时间
real代表程序从开始到结束所用的时钟时间。这个时间包括其他进程使用的时间片和进程阻塞的时间（比如等待 I/O 完成）。

[PSYoungGen”和”[ParOldGen”是指GC发生的区域，分别代表使用**Parallel Scavenge**垃圾收集器的新生代和使用**Parallel old**垃圾收集器的老生代。

[GC”和”[Full GC”说明了这次垃圾收集的停顿类型，如果有”Full”，说明这次GC发生了”Stop-The-World”。
Allocation Failure说明失败类型。

8192k->1008k（9216k）：GC前该内存区域已使用的容量->GC后该内存区域已使用的容量(该内存区域总容量)
8192k->4833k（19456k）：GC前Java堆已使用容量->GC后Java堆已使用容量(Java堆总容量)
0.0052850表示GC总用时

最后堆内存的使用情况

```
[GC (Allocation Failure) [PSYoungGen: 8192K->1008K(9216K)] 8192K->4833K(19456K), 0.0052850 secs] [Times: user=0.04 sys=0.00, real=0.00 secs]
[GC (Allocation Failure) --[PSYoungGen: 9200K->9200K(9216K)] 13025K->19432K(19456K), 0.0135017 secs] [Times: user=0.07 sys=0.01, real=0.02 secs]
[Full GC (Ergonomics) [PSYoungGen: 9200K->0K(9216K)] [ParOldGen: 10232K->10072K(10240K)] 19432K->10072K(19456K), [Metaspace: 3289K->3289K(1056768K)], 0.1391476 secs] [Times: user=0.30 sys=0.00, real=0.14 secs]
[Full GC (Ergonomics) [PSYoungGen: 7964K->7784K(9216K)] [ParOldGen: 10072K->7961K(10240K)] 18037K->15745K(19456K), [Metaspace: 3289K->3289K(1056768K)], 0.1316681 secs] [Times: user=0.33 sys=0.01, real=0.13 secs]
[Full GC (Allocation Failure) [PSYoungGen: 7784K->7784K(9216K)] [ParOldGen: 7961K->7943K(10240K)] 15745K->15727K(19456K), [Metaspace: 3289K->3289K(1056768K)], 0.0810004 secs] [Times: user=0.40 sys=0.01, real=0.08 secs]
Heap
 PSYoungGen      total 9216K, used 8025K [0x00000007bf600000, 0x00000007c0000000, 0x00000007c0000000)
  eden space 8192K, 97% used [0x00000007bf600000,0x00000007bfdd6518,0x00000007bfe00000)
  from space 1024K, 0% used [0x00000007bfe00000,0x00000007bfe00000,0x00000007bff00000)
  to   space 1024K, 48% used [0x00000007bff00000,0x00000007bff7bef8,0x00000007c0000000)
 ParOldGen       total 10240K, used 7943K [0x00000007bec00000, 0x00000007bf600000, 0x00000007bf600000)
  object space 10240K, 77% used [0x00000007bec00000,0x00000007bf3c1f78,0x00000007bf600000)
 Metaspace       used 3320K, capacity 4500K, committed 4864K, reserved 1056768K
  class space    used 367K, capacity 388K, committed 512K, reserved 1048576K
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 heapoom.HeapOOM.main(HeapOOM.java:14)

Process finished with exit code 1
```