# EventProcessor.ProcessRef 该ThreadLocal 在两种情况下存在泄露，导致shutdown回收不掉sandbox的Classloader，造成MetaSpace空间内存泄露

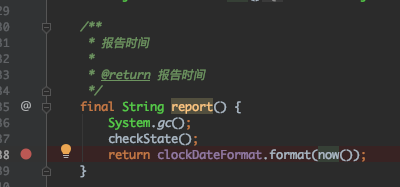
情况一：使用了ProcessControl 特性，shutdown无法释放SandboxClassLoader  
分析：  
EventListenerHandlers.handleEvent中的  
catch (ProcessControlException pce) {  
final EventProcessor.Process process = processor.processRef.get();  
....}  
此处的 processor.processRef 很可能在 EventListenerHandlers.handleOnEnd 的方法中，已经被释放掉，此处的get会导致重新初始化且后续无释放

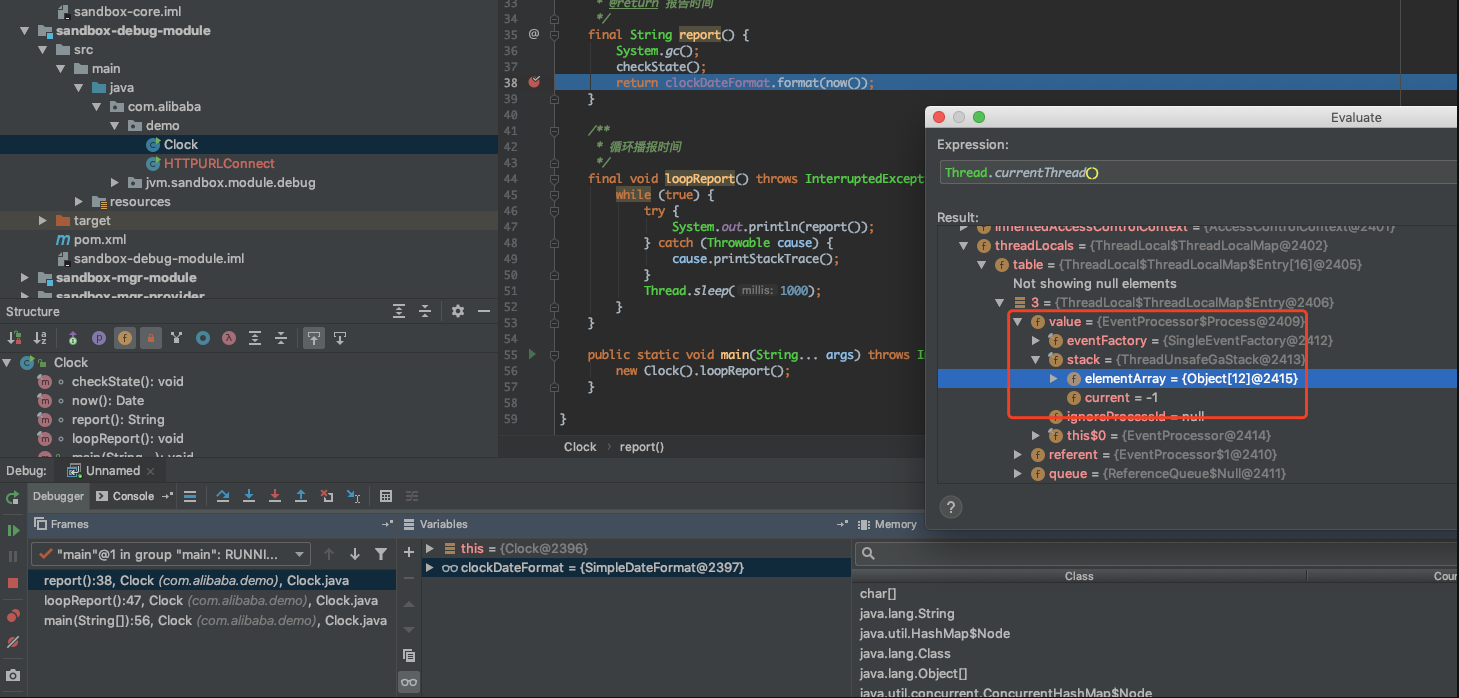
一个空Process 对象放在ThreadLocal中，导致使用了ProcessControl 特性且复用线程的的应用，ThreadLocal始终存在没有释放的Process，shutdown导致Classloader泄露

情况二：  
被切的方法存在阻塞方法或方法执行时间较长，在阻塞或方法自身执行的的过程中，shutdown会导致processor.processRef 无法释放，并发分析图如下

情况一：使用了ProcessControl 特性，shutdown无法释放SandboxClassLoader

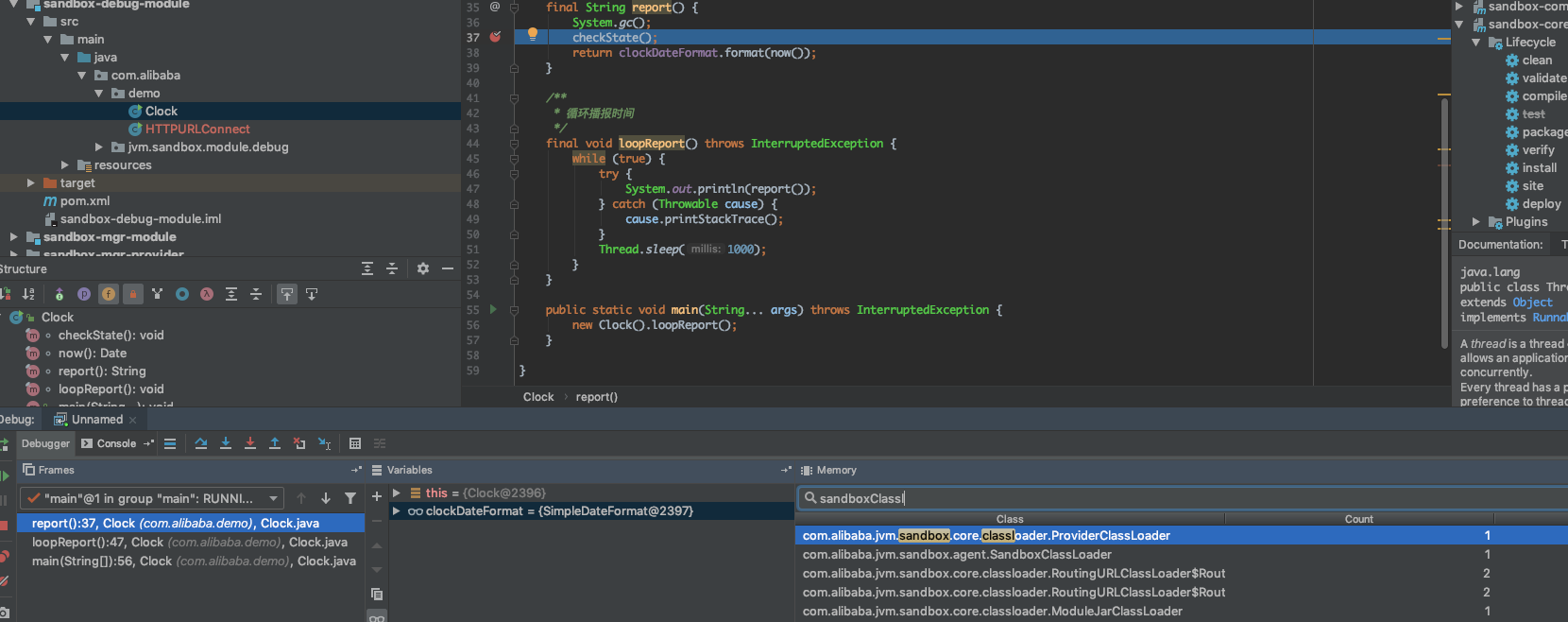
1. 启动官方Clock例子
2. 挂载sandbox
3. 执行表修复脚本

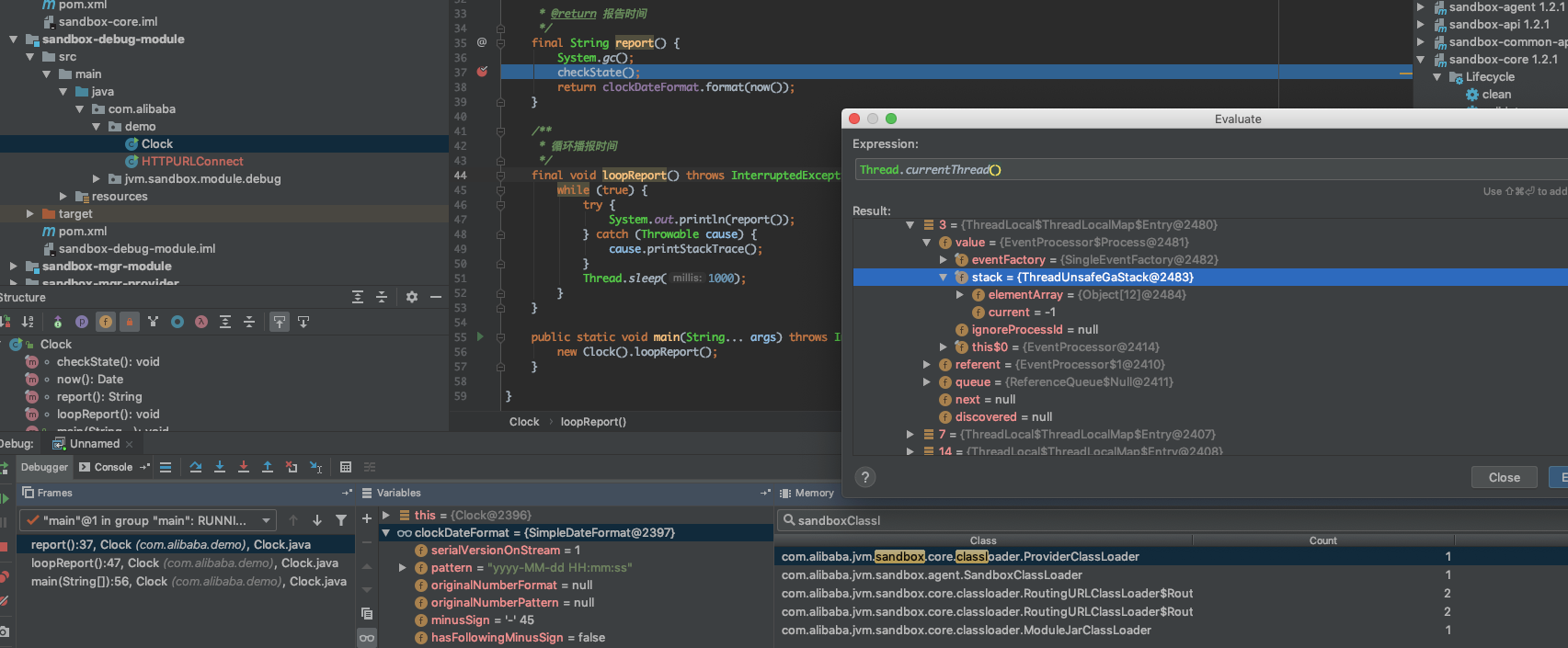
在Clock例子中打上断点,为了加快gc，在代码里主动加了System.gc  


检查当前线程本地变量，发现有Sandbox的类  
  
此时，代码已经走到sandbox未增强的指令中（断点处非checkstate方法），

大胆猜测此时发生shutdown，不会释放用户线程中的EventProcessor$Prcocess

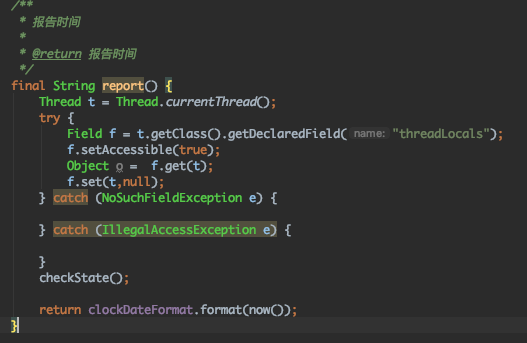
1. 恢复执行，执行shutdown

断点后检查shutdown遗留的类  


检查当前线程本地变量,发现有Sandbox的类  


基本认为是线程本地变量发生了泄露

1. 验证思想

shutdown后在用户代码中暴力清理线程本地变量  


重复1-4的流程，检查遗留的类,可以看到基本清理干净  
