主题： Jmtrace实验报告

姓名： 黄彬寓

学号： MF20330030

1 设计总述

访问内存的指令只有 getstatic/putstatic/getfield/putfield/*aload/*astore，因此我们只要对上述访问内存的指令进行捕获，就能够收集到所有访问共享内存的信息。

本实验使用到的技术主要有ASM和java.lang.Instrument这两个模块，Instrument提供了加载class文件后对文件中的字节码进行修改的功能，而ASM用来具体完成对字节码的重写。

Instrument有两种加载Agent模式，一种是on load加载，也就是在VM启动时加载Agent；第二种是on attach加载，也就是在VM运行时加载Agent，这里我们使用的是启动时加载，所以命令参数为含-javaagent的形式，且需要对premain函数进行重载。

ASM是功能比较齐全的java字节码操作与分析框架，通过ASM框架，我们可以利用访问者设计模式进行遍历，在遍历过程中对字节码进行修改，主要是对visitInsn和visitFieldInsn两个ASM内定义函数进行重写，使得我们希望的内容能够被打印出来。

2 实现步骤

1.在Instrument模块，通过premain在启动时加载Agent，JVM在每次在装载class的时候调用transform函数，通过transform我们获得了进行改写的接口。

2.随后到达ClassVisitor模块，我们创建一个它的继承类ASMClassChange，来对它进行重写，通过在其内部重写visitMethod使我们能够进一步地对MethodVisitor模块进行改写。

3.随后进入MethodVisitor模块，我们创建一个它的继承类ASMMethodChange，在这个模块中，我们把需要修改的指令分为两部分，一类是*aload和*astore，即数组类型的访问，在重写的visitInsn中进行打印；另一类是get/put static/field，即对field访问，在重写的visitFieldInsn中进行打印。

3 实现细节

1.打印条目：在本实验中，打印函数统一使用printLog(int index,int rw,String name,Object owner,String arrayType)，这5个变量分别表示为index数组下标，rw读或写，name对象的类名和变量名，owner为对象标识用于生成哈希值，arrayType为数组类型。

2.打印前的保存现场：类似于栈中的函数调用，我们打印前需要往栈顶存入希望的变量，以使打印函数printLog能成功访问到，于是要依次往栈中存放index、rw、name、owner、arrayType这些信息。index和arrayType仅对数组有效，name仅对class field有效。

3.*astore类型提取index：在调用*astore指令，栈信息为..., arrayref, index, value，显然index并不在栈顶，我们通过先执行DUP2使得栈变为..., arrayref, index, value, index, value，再通过一个POP指令，将value pop掉使得栈顶为index。

4.使用类全局变量简化提取信息难度：在*aload和*astore中，我们显然需要arrayType和owner，但这些信息仅通过visitInsn传进来的opcode是无法得到的，但是整个执行过程我们可以看作是串行的，所以我们所需要的信息一定恰好在不久之前被访问过，于是通过类全局变量STATIC_STATE,current_owner和arrayTyp确定，我们就可以轻松的在任何地方知道当前的这些关键信息。