**import** java.io.File;

**import** java.util.Iterator;

**import** soot.Body;

**import** soot.Scene;

**import** soot.SootClass;

**import** soot.SootMethod;

**import** soot.Unit;

**import** soot.UnitPatchingChain;

**import** soot.Value;

**import** soot.jimple.ArrayRef;

**import** soot.jimple.InstanceFieldRef;

**import** soot.jimple.StaticFieldRef;

**import** soot.jimple.Stmt;

**import** soot.jimple.internal.JAssignStmt;

**import** soot.options.Options;

**public** **class** HelloWorld {

**int** f; // Field.

**static** **int** *g*; // Static field.

**int**[] a = { 1, 2, 3 }; // Array.

**public** **static** **void** main(String[] args) {

Options.*v*()

.set\_soot\_classpath(System.*getProperty*("user.dir") + File.***separator*** + "bin" + File.***pathSeparator***

+ System.*getenv*("JAVA\_HOME") + File.***separator*** + "jre/lib/rt.jar" + File.***pathSeparator***

+ System.*getProperty*("user.dir") + File.***separator*** + "soot-4.0.0-jar-with-dependencies.jar"

+ File.***pathSeparator*** + System.*getenv*("JAVA\_HOME") + File.***separator*** + "jre/lib/jce.jar");

Options.*v*().set\_whole\_program(**true**);

Options.*v*().set\_ignore\_resolving\_levels(**true**);

Scene.*v*().loadNecessaryClasses();

SootClass cl = Scene.*v*().loadClassAndSupport("HelloWorld");

cl.setApplicationClass();

SootMethod method = cl.getMethodByName("test");

Body body = method.retrieveActiveBody();

UnitPatchingChain chain = body.getUnits();

Iterator<Unit> it = chain.iterator();

**while** (it.hasNext()) {

Unit unit = it.next();

Stmt stmt = (Stmt) unit;

**if** (stmt **instanceof** JAssignStmt) {

Value lhs = ((JAssignStmt) stmt).getLeftOp();

Value rhs = ((JAssignStmt) stmt).getRightOp();

**if** (lhs **instanceof** InstanceFieldRef) {

System.***out***.println(

"Statement a." + stmt.getFieldRef().getField().getName() + " = " + rhs + ", heap write;");

} **else** **if** (lhs **instanceof** StaticFieldRef) {

System.***out***.println(

"Statement A." + stmt.getFieldRef().getField().getName() + " = " + rhs + ", heap write;");

} **else** **if** (lhs **instanceof** ArrayRef) {

System.***out***.println("Statement " + stmt.getArrayRef().getBase() + " = " + rhs + ", heap write;");

}

**if** (rhs **instanceof** InstanceFieldRef) {

System.***out***.println(

"Statement " + lhs + " = a." + stmt.getFieldRef().getField().getName() + ", heap read;");

} **else** **if** (rhs **instanceof** StaticFieldRef) {

System.***out***.println(

"Statement " + lhs + " = A." + stmt.getFieldRef().getField().getName() + ", heap read;");

} **else** **if** (rhs **instanceof** ArrayRef) {

System.***out***.println("Statement " + lhs + " = " + stmt.getArrayRef().getBase() + ", heap read;");

}

}

}

}

**public** **void** test() {

**int** b = 5;

**int** i = 0;

f = b;

b = f;

*g* = b;

b = *g*;

a[i] = b;

b = a[i];

}

}

**Output**

Statement a.f = b1, heap write;

Statement i2 = a.f, heap read;

Statement A.g = i2, heap write;

Statement i3 = A.g, heap read;

Statement $r1 = a.a, heap read;

Statement $r1 = i3, heap write;

Statement $r2 = a.a, heap read;

Statement i4 = $r2, heap read;