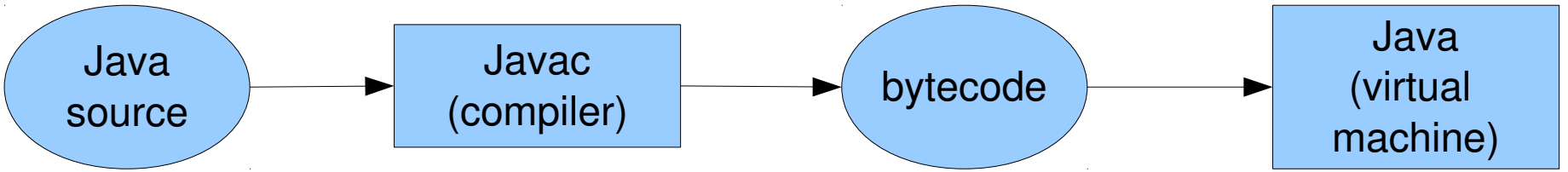
Sobreviviendo SOOT

La máquina virtual Java

- El compilador Java traduce un programa Java en el lenguaje de entrada de la Java Virtual Machine (JVM)
- El bytecode Java es una suerte de lenguaje "assembler" para la JVM



El framework Soot

- Conjunto de APIs de Java para operar con bytecode Java
  - optimización
  - anotación
- Creado por el Sable Research Group (http://www.sable.mcgill.ca/)
- Web: http://www.sable.mcgill.ca/soot/

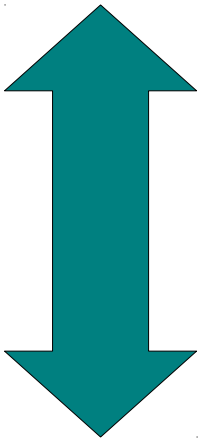
Representaciones Intermedias

- Jimple: ppal representación de Soot
- Grimp: Jimple con expresiones
- Shimple: Jimple con SSA
- Baf: bytecode "humanizado"

Representaciones Intermedias

Java

- Grimple (alto)
- Jimple/Shimple (medio)
- Baf (bajo)



Bytecode

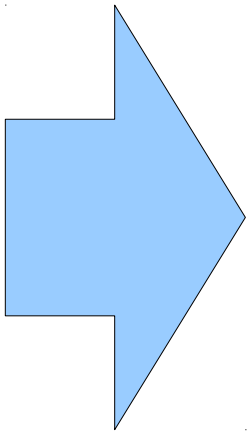
Jimple

- Puede crearse usando:
  - programa Java
  - bytecode Java
- Características :
  - 3 referencias: Todas las expresiones usan a lo sumo 3 direcciones
  - No-estructurado: while's, if's, for's son reemplazados por GOTO's
  - Tipado: las variables locales son tipadas

Jimple: Ejemplo

- Programa Java original
- Representación Jimple

```
if (x+y!=z)
  return;
else
  System.out.println(“foo”);
```



```
  t = x+y;
  if (t==z) goto label10;
  return;

label10:
  ref = System.out;
  ref.println(“foo”);
```

Soot: Análisis dataflow

- Eclipse plugin:

  1. seleccionar Resource Perspective

  2. botón derecho sobre el archivo Java a analizar

- Soot -> Process Source File -> Run Soot ...
  - Output Options -> Output Format -> Jimple File
  - Phase Options -> Jimple Annotation Pack ->
    - Live Variables Tagger
    - Reaching Defs Tagger
    - Available Expressions Tagger (etc...)

Soot: Análisis dataflow

(demo)

Soot: Análisis de dataflow

- Modo Interactivo:
  - Run... -> General Options -> Interactive Mode
- Ejecutando un Análisis creado
  - Run... -> Soot Main Class

Desarrollando un análisis Soot

1. Crear nuevo proyecto
2. Agregar al Build Path del proyecto las siguientes librerías externas:
   - jasminclasses.jar
   - polyglot.jar
   - sootclasses.jar
   - O solamente soot-trunk.jar
- Estas librerías se encuentran en la carpeta lib/ del plugin
- Javadoc:
  - http://www.sable.mcgill.ca/soot/doc/index.html

Dataflow framework

1. Decidir dirección del flujo: ¿backward o forward?
2. Decidir aproximación: ¿may o must?
3. Realizar la transformación del flujo: ej: ¿cómo deben tratarse las asignaciones?
4. Decidir estados iniciales y el flujo inicial del entry/exit node (dependiendo del tipo de análisis)

1.Dirección del flujo

- Soot posee 3 implementaciones de análisis :
  - ForwardFlowAnalysis
  - BackwardFlowAnalysis
  - ForwardBranchedFlowAnalysis
- La salida es un Map<Nodo,<IN set, OUT set>>

```
public class MyFwdAnalysis extends ForwardFlowAnalysis<Unit,FlowSet> {
  public MyFwdAnalysis(DirectedGraph<Unit> graph) {
    super(graph);
    doAnalysis();
  }}
```

2. Aproximación

- Implementar los métodos “merge” y “copy”

```
protected void merge(FlowSet intSet1,
                     FlowSet inSet2,
                     FlowSet outSet) {
   inSet1.intersection(inSet2, outSet);
}

protected void copy(FlowSet srcSet,
                    FlowSet destSet) {
   srcSet.copy(destSet);
}
```

3. Transformación del flujo

- Implementar el método flowTrough

```
protected void flowThrough(FlowSet in,
                           Unit node,
                           FlowSet out) {
  kill(inSet, u, outSet);
  gen(outset, u);
}
```

- Los métodos kill y gen son definidos por el usuario

4. Flujos iniciales

- Hay que definir el contenido inicial del entry/exit point, y de los otros nodos del CFG

```
protected FlowSet entryInitialFlow() {
  return new FlowSet();
}
protected FlowSet newInitialFlow() {
  return new FlowSet();
}
```

Hay distintos tipos de FlowSet

- ArraySparseSet
- ArrayPackedSet
- ToppedSet

Ejecutando el análisis

```
Scene.v().setSootClassPath(“...”)
SootClass c =
Scene.v().loadClassAndSupport("ar.edu.soot.MyClass");
c.setApplicationClass();
SootMethod m = c.getMethodByName("myMethod");
Scene.v().loadNecessaryClasses();
Body b = m.retrieveActiveBody();
UnitGraph g = new ExceptionalUnitGraph(b);
MyFwdAnalysis an = new MyFwdAnalysis(g);
for (Unit unit : g) {
  FlowSet in = an.getFlowBefore(unit);
  FlowSet out = an.getFlowAfter(unit);
}
```

Transformación del flujo

- Se puede utilizar el design-pattern Visitor para recorrer el AST Jimple :
  - soot.jimple.AbstractStmtSwitch
  - soot.jimple.AbstractJimpleValueSwitch