

Soot Java Optimization Framework

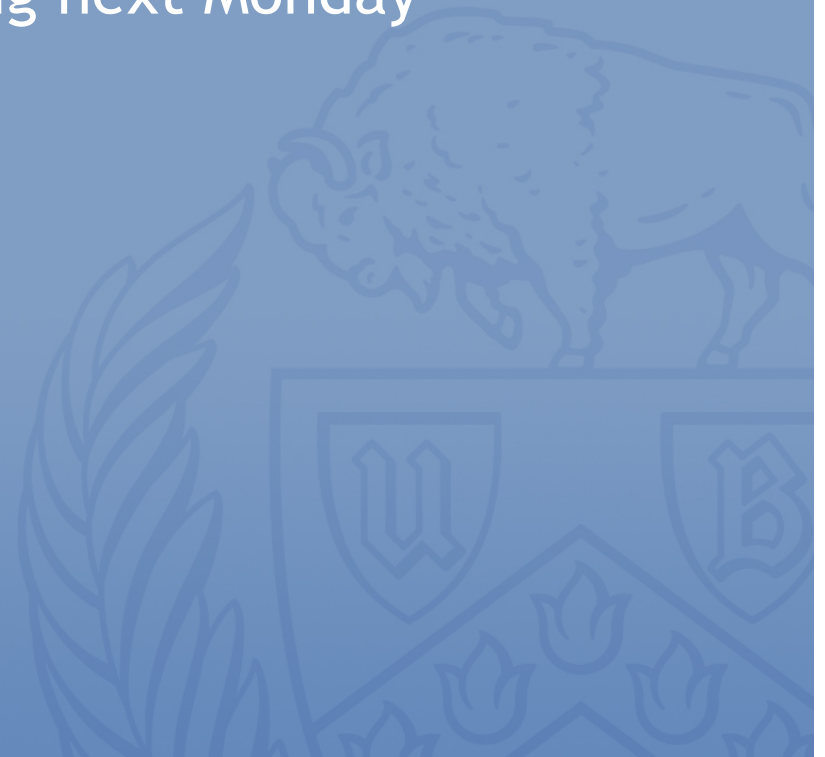
Karthik Dantu and Steve Ko





Administrivia

- Friday meetings preponed to today
- Mid-term Presentations starting next Monday





Today: Soot

- Why?
 - Good support for Android app analysis and instrumentation
- Resources:
 - Soot website: <http://www.sable.mcgill.ca/soot/>
 - Tutorial: <http://www.sable.mcgill.ca/soot/tutorial/pldi03/tutorial.pdf>



Soot

- Open-source compiler infrastructure
- Features
 - Bytecode transformation
 - Optimization
 - APIs for program analysis
- Supports many languages
 - Java, SML, Eiffel, Scheme
- Will mainly focus on Java





Bytecode Transformation

- Support
 - Baf: Compact bytecode
 - Jimple: IR
 - Shimple: Jimple, but SSA
 - Grimp: Jimple, but more compact
 - Dava: Decompiled Java
- For program analysis, mostly use Jimple or Shimple
- Demo

Detour: SSA

- Single Static Assignment
 - A variable is assigned exactly once and never gets re-assigned.
 - This makes compiler analysis easier.
- Wikipedia example
 - Code: `{ y = 1; y = 2; x = y; }`
 - SSA: `{ y1 = 1; y2 = 2; x = y2; }`

Jimple

- Core statements:

NopStmt

DefinitionStmt: IdentityStmt,
AssignStmt

- Intraprocedural control-flow:

IfStmt

GotoStmt

TableSwitchStmt, LookupSwitchStmt

- Interprocedural control-flow:

InvokeStmt

ReturnStmt, ReturnVoidStmt

Jimple

- `ThrowStmt`
throws an exception
- `RetStmt`
not used; returns from a JSR
- `MonitorStmt`: `EnterMonitorStmt`,
`ExitMonitorStmt`
mutual exclusion

Jimple (Simplified)

```
public int foo(java.lang.String) { // locals  
    r0 := @this;                  // IdentityStmt  
    r1 := @parameter0;  
  
    if r1 != null goto label0; // IfStmt  
  
    $i0 = r1.length();            // AssignStmt  
    r1.toUpperCase();             // InvokeStmt  
    return $i0;                   // ReturnStmt  
  
label0:                          // created by Printer  
    return 2;  
}
```



Jimple Example



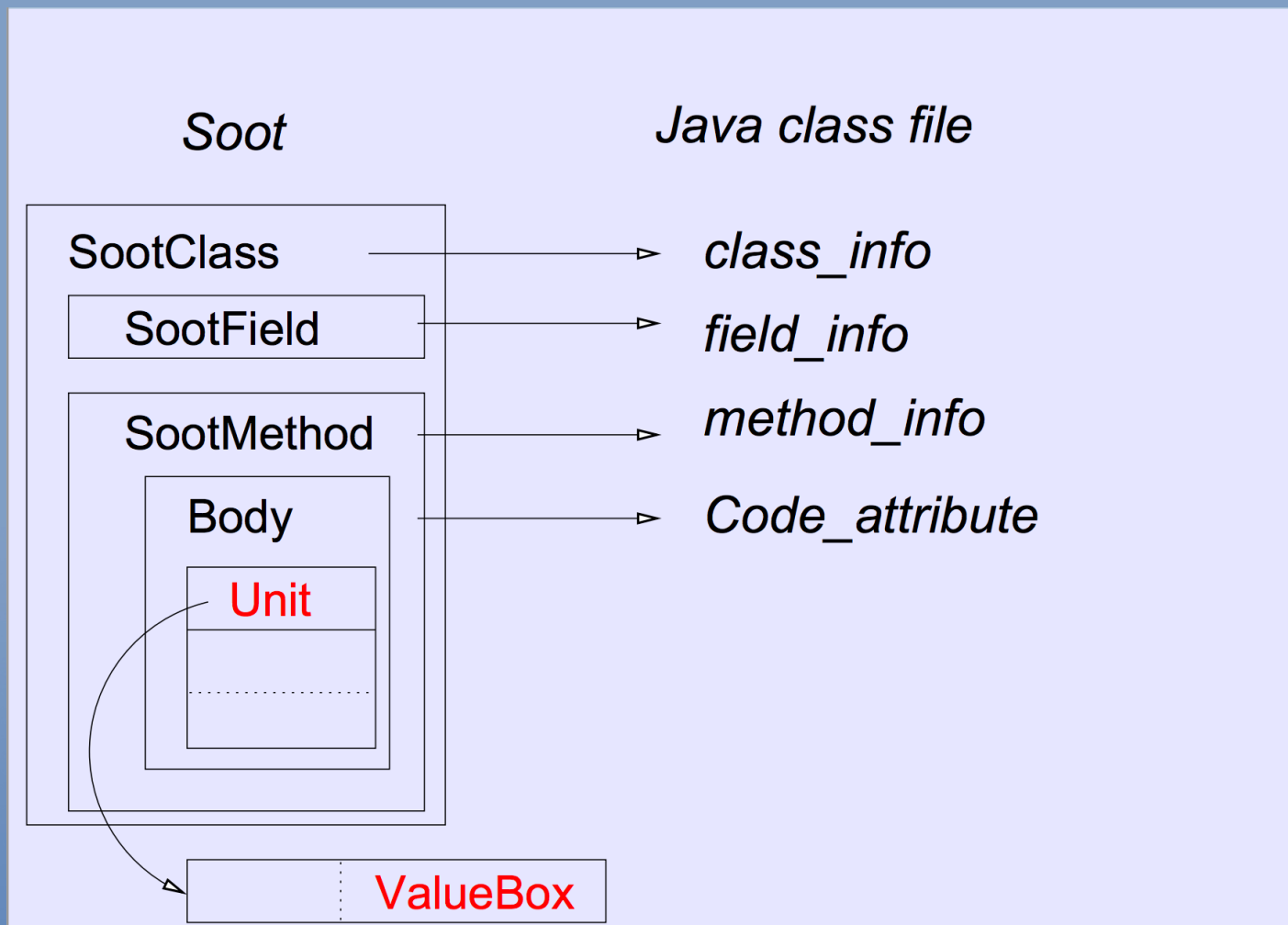


Call Types

- specialinvoke: special invoke cases
 - Constructor, methods in super, private methods
- virtualinvoke
 - Regular method call
- interfaceinvoke
 - Interface method call
- staticinvoke
 - Static method call



Soot Internal Representation

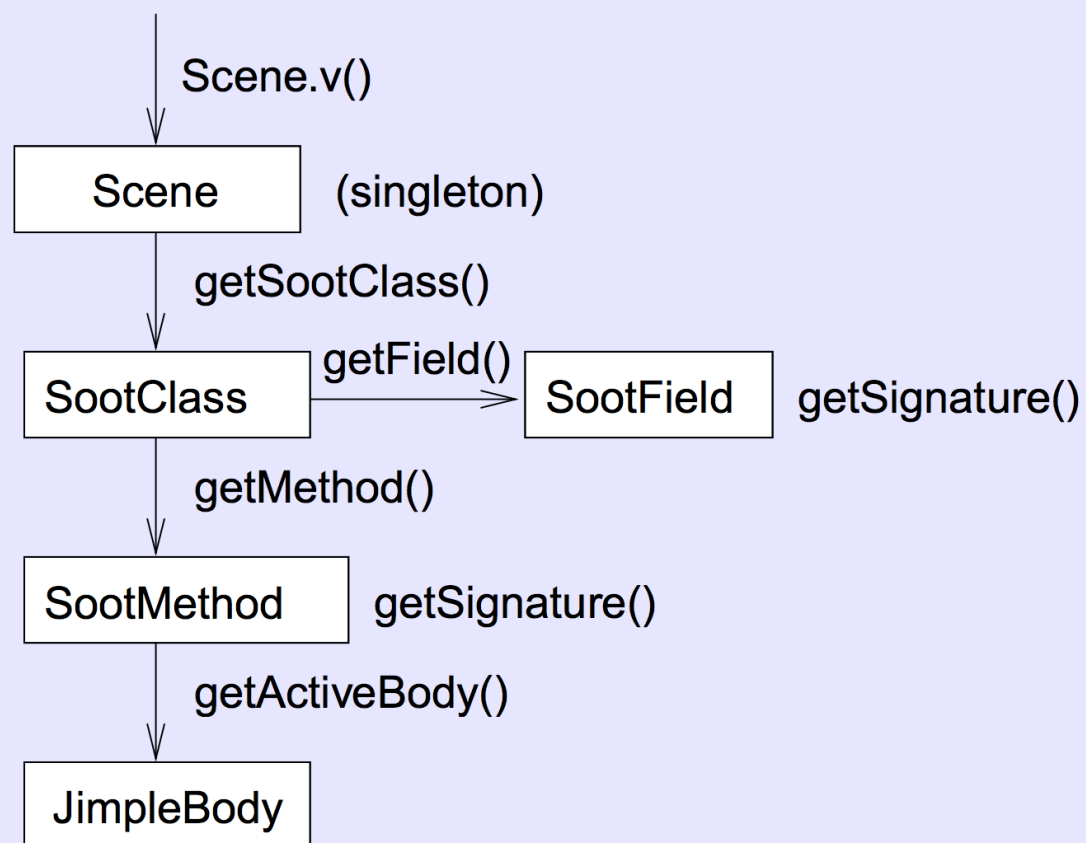




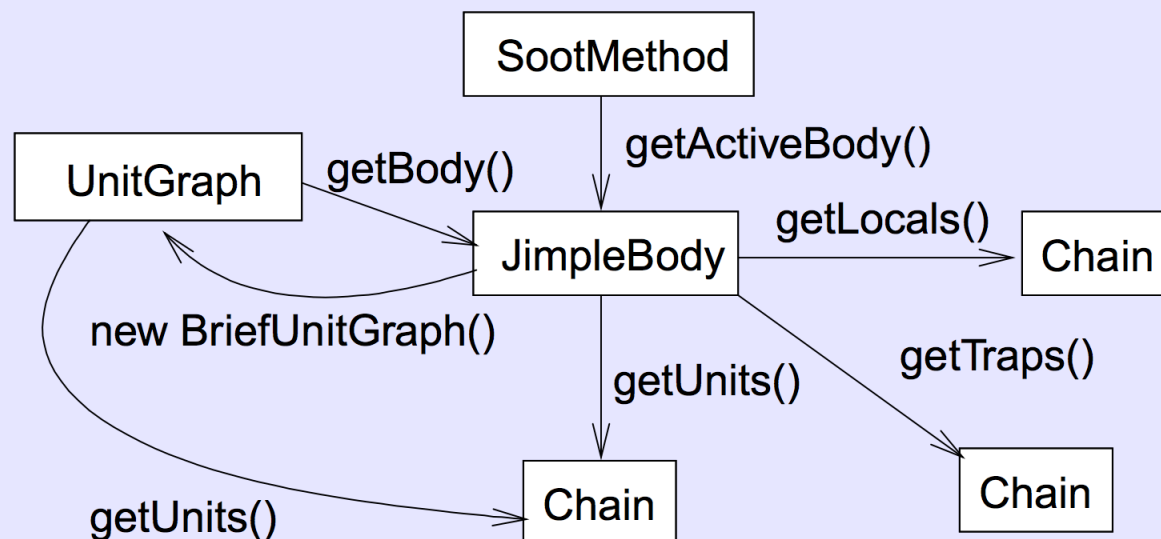
APIs for Program Analysis

- Scene: data structure for a whole program
- SootClass: data structure for classes
- SootMethod: data structure for methods
- SootField: data structure for fields
- Method bodies (e.g., JimpleBody): data structure for method body (code)

Hierarchy



Examining a Body





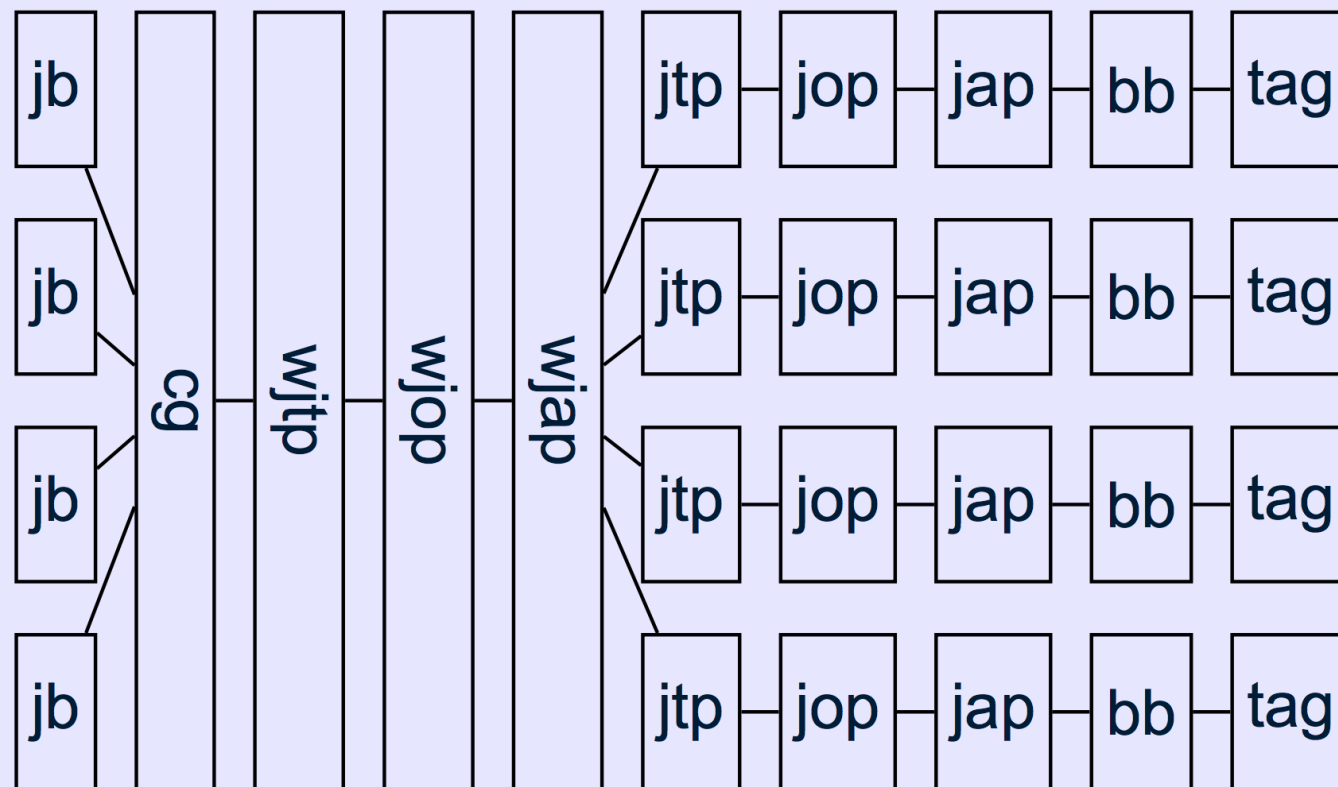
Extending Soot

- Ask Soot to run your analysis code
- Process
 - Write a “transformer”
 - Add your transformer to a phase
 - Run soot
- Example





Soot Phases





Soot Phases

- jb: Jimple body creation
- cg: Call graph generation
- wjtp: Whole program transformation (user-defined)
- wjop: Whole program optimization
- wjap: Whole program annotation generation
- jtp, jop, jap: Intraprocedural versions

Soot Code Example





Program Analysis

- Intraprocedural flow analysis
- Interprocedural flow analysis
- Reachability analysis
- Points-to analysis
- Etc.





Summary

- Soot Java optimization framework
- Features
 - Bytecode transformation
 - Built-in optimization
 - APIs for program analysis
- Java and Android

