

WELCOME TO THE
TUTORIAL!



Tutorial Team



Kadiray Karakaya



Stefan Schott



Jonas Klauke

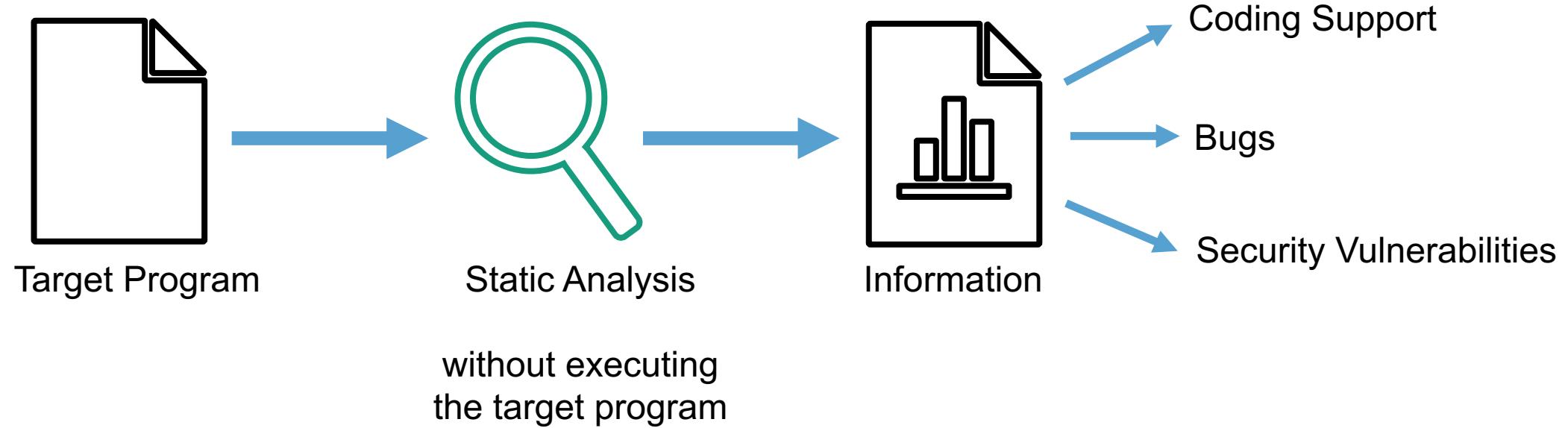


Eric Bodden



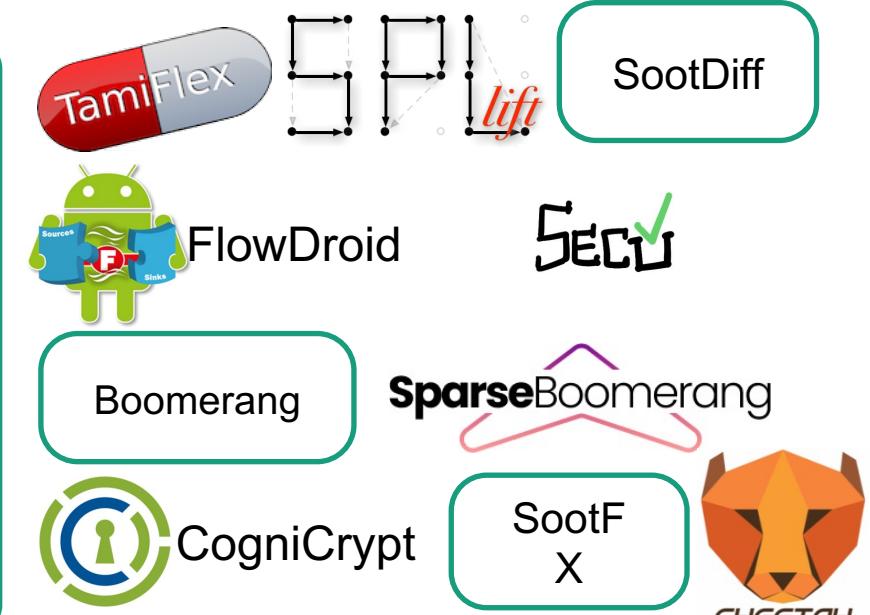
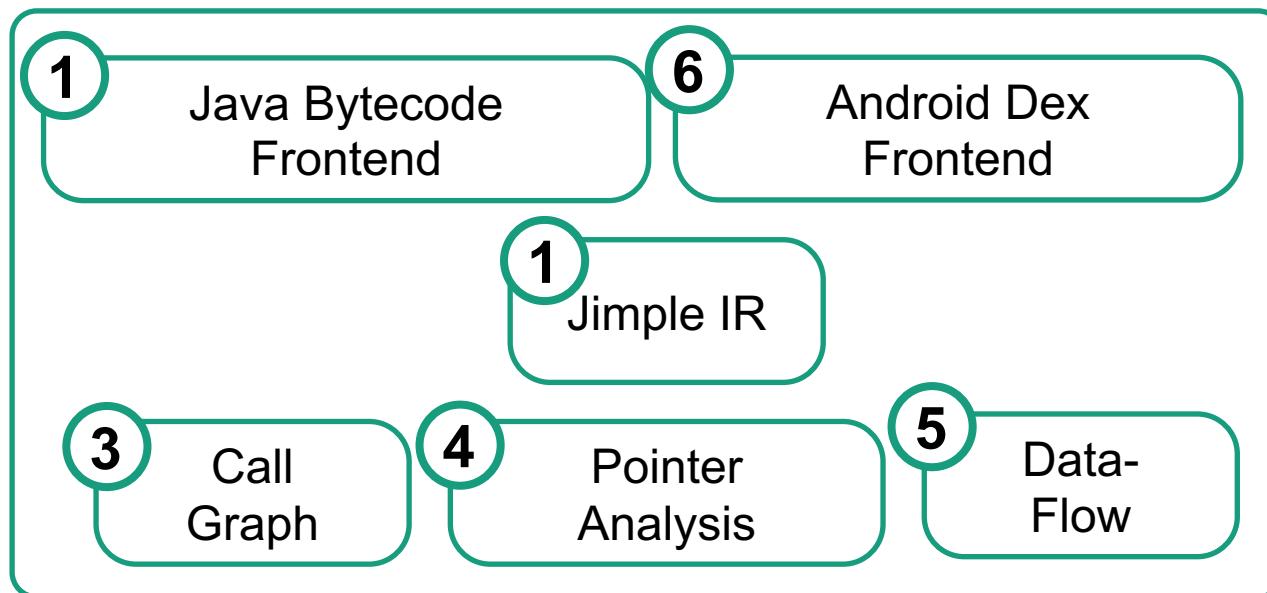
Markus Schmidt

Context: Static Program Analysis

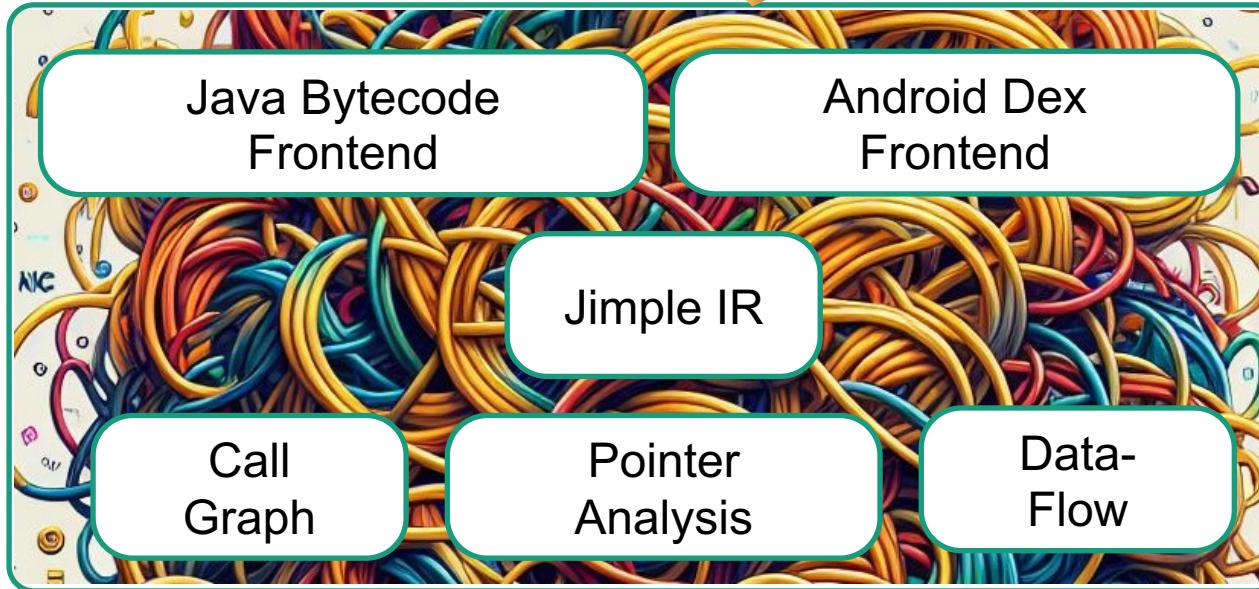


SootUp is a redesign of the **Soot** static analysis framework.

Soot: A Brief History

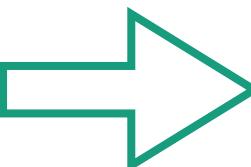


Soot: Strengths & Drawbacks



- Jimple IR
 - Local vars instead of stack in bytecode
 - No nested scopes, explicit control-flow
- Call Graph algorithms
 - CHA, RTA, DTA, VTA
- SPARK pointer analysis framework
 - Whole program
 - Context- and flow-insens.
- Heros Data-Flow Analysis framework
 - IFDS/IDE

Frequently extended
to support new features



Suboptimal Design

“Soot is very difficult to learn”

0:00 / 16:02 • Introduction >

IEEE SCAM
25 subscribers

Subscribe

12 | Dislike

Share

Download

...

@shuomao1073 2 years ago

soot is very difficult to learn, its official document is too bad.

Like 5 Dislike Reply

▼ 5 replies



Deutsche
Forschungsgemeinschaft
German Research Foundation

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

 **Fraunhofer**
IEM

SootUp aims to be beginner friendly

- Dedicated web page for:
 - Up-to-date documentation
 - Tutorials
- Dedicated repository with running examples

The screenshot shows the homepage of the SootUp v1_2_0 website. The header includes the SootUp logo and a dropdown menu for version v1_2_0. The main content area has a title "What's SootUp?" and a sub-section "SootUp is a complete overhaul" with two bullet points: "Transforms JVM bytecode" and "Provides ClassHierarchy g". The sidebar contains links for SootUp Home, Announcements, Design Decisions, Basics (Installation, Getting started), and a "v1_2_0" link.

The screenshot shows a GitHub repository named "SootUp-Examples" which is public. It features a main branch, a code editor button, and a list of three commits by user "ReshmaSobhaNair". Each commit removed a test method from specific examples: "BasicSetupExample", "BodyInterceptorExa...", and "CallgraphExample", all done 2 months ago.

Commit	File	When
Merge ...	BasicSetupExample	Removed test method 2 months ago
Merge ...	BodyInterceptorExa...	Removed test method 2 months ago
Merge ...	CallgraphExample	Removed test method 2 months ago

Development Process

- Actively maintained at Github



- No breaking changes through CI



- High code coverage through systematic testing



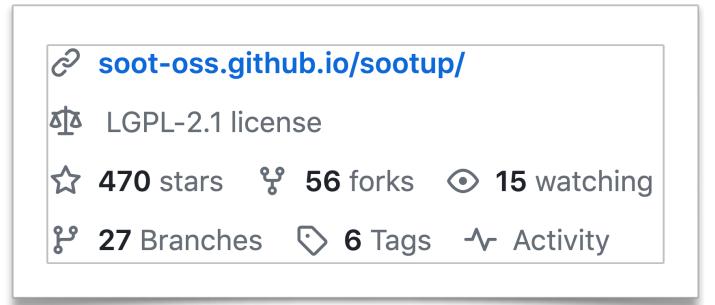
- Up-to-date documentation & tutorials



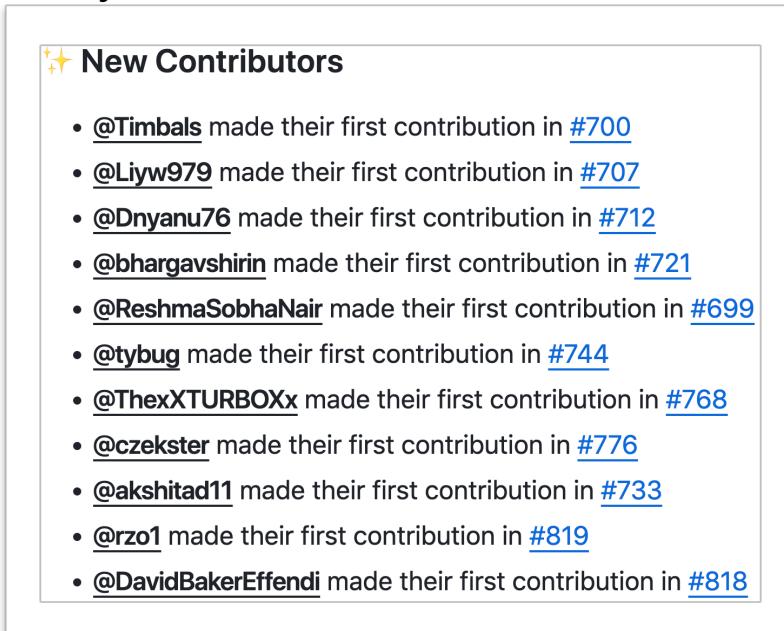
- Released at Maven Central

Building a User Base

- Already an active user base



- Many outside contributors



- In Labs:
 - SootUp is used in our static analysis course at Paderborn University
- Hackathons (CTF):
 - 1st SootUp Hackathon, at UPB in May 2023
 - 2nd SootUp Hackathon, at UPB in June 2024
- Today: SootUp Tutorial!

Tutorial Program

What's next?

- Technical Introduction
- Exercise 1: Get to know the API
- Live Demo: Call Graph Construction
- Exercise 2: Taint analysis with SootUp and Heros
- Closing

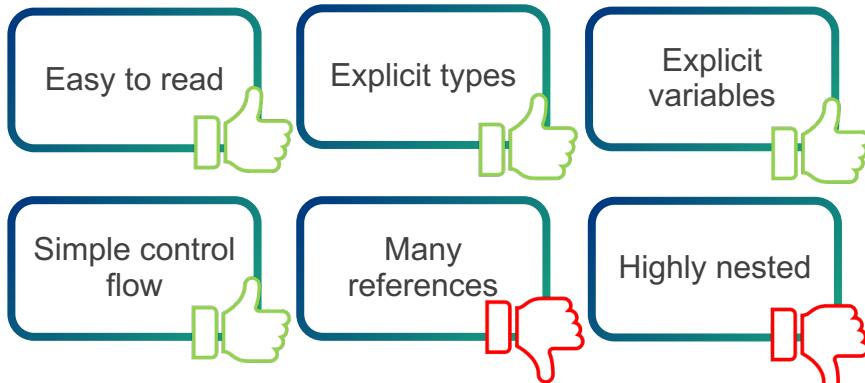
TECHNICAL INTRODUCTION

Source Code vs Bytecode vs Jimple



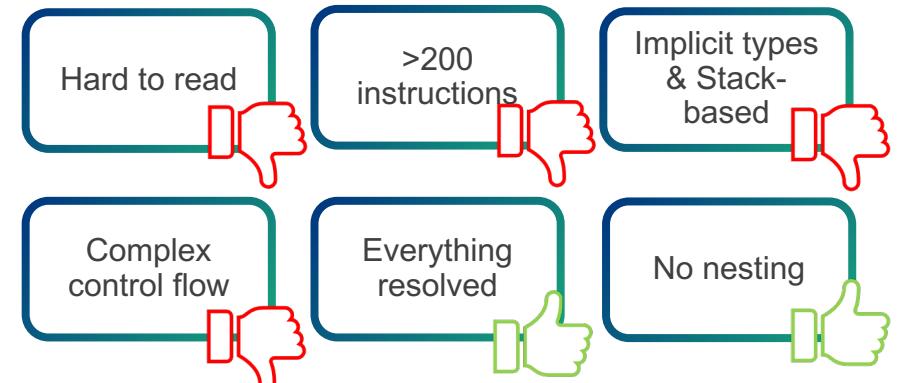
```
public static void sampleMethod(int x) {  
    if (x % 2 == 0) {  
        System.out.println("Even");  
    } else {  
        System.out.println("Odd");  
    }  
}
```

Java Source Code



```
1 public static sampleMethod(I)V  
2   L0  
3     ILOAD 0  
4     ICONST_2  
5     IREM  
6     IFNE L1  
7   L2  
8     GETSTATIC java/lang/System.out : Ljava/io/PrintStream;  
9     LDC "Even"  
10    INVOKEVIRTUAL java/io/PrintStream.println (Ljava/lang/String;)V  
11    GOTO L3  
12  L1  
13    FRAME SAME  
14    GETSTATIC java/lang/System.out : Ljava/io/PrintStream;  
15    LDC "Odd"  
16    INVOKEVIRTUAL java/io/PrintStream.println (Ljava/lang/String;)V  
17  L3  
18    FRAME SAME  
19    RETURN  
20  L4  
21    LOCALVARIABLE x I L0 L4 0  
22    MAXSTACK = 2  
23    MAXLOCALS = 1  
24 }
```

Java Bytecode



Source Code vs Bytecode vs Jimple

○ ○ ○

```
public static void sampleMethod(int x) {  
    if (x % 2 == 0) {  
        System.out.println("Even");  
    } else {  
        System.out.println("Odd");  
    }  
}
```

Java Source Code

Easy to read



Explicit types



Simple control flow



Explicit variables



○ ○ ○

```
1 public static void sampleMethod(int)  
2 {  
3     int x, $stack1;  
4     java.io.PrintStream $stack2, $stack3;  
5  
6     x := @parameter0: int;  
7  
8     $stack1 = x % 2;  
9     if $stack1 ≠ 0 goto label1;  
10  
11    $stack3 = <java.lang.System: java.io.PrintStream out>;  
12    virtualinvoke $stack3.<java.io.PrintStream:  
13        void println(java.lang.String)>("Even");  
14    goto label2;  
15  
16    label1:  
17    $stack2 = <java.lang.System: java.io.PrintStream out>;  
18    virtualinvoke $stack2.<java.io.PrintStream:  
19        void println(java.lang.String)>("Odd");  
20  
21    label2:  
22    return;  
23 }
```

Jimple

○ ○ ○

```
1 public static sampleMethod(I)V  
2     L0  
3         ILOAD 0  
4         ICONST_2  
5         IREM  
6         IFNE L1  
7     L2  
8         GETSTATIC java/lang/System.out : Ljava/io/PrintStream;  
9         LDC "Even"  
10        INVOKEVIRTUAL java/io/PrintStream.println (Ljava/lang/String;)V  
11        GOTO L3  
12     L1  
13         FRAME SAME  
14         GETSTATIC java/lang/System.out : Ljava/io/PrintStream;  
15         LDC "Odd"  
16         INVOKEVIRTUAL java/io/PrintStream.println (Ljava/lang/String;)V  
17     L3  
18         FRAME SAME  
19         RETURN  
20  
21 LOCALVARABLE x I L0 L4 0  
22 MAXSTACK = 2  
23 MAXLOCALS = 1  
24 }
```

Java Bytecode

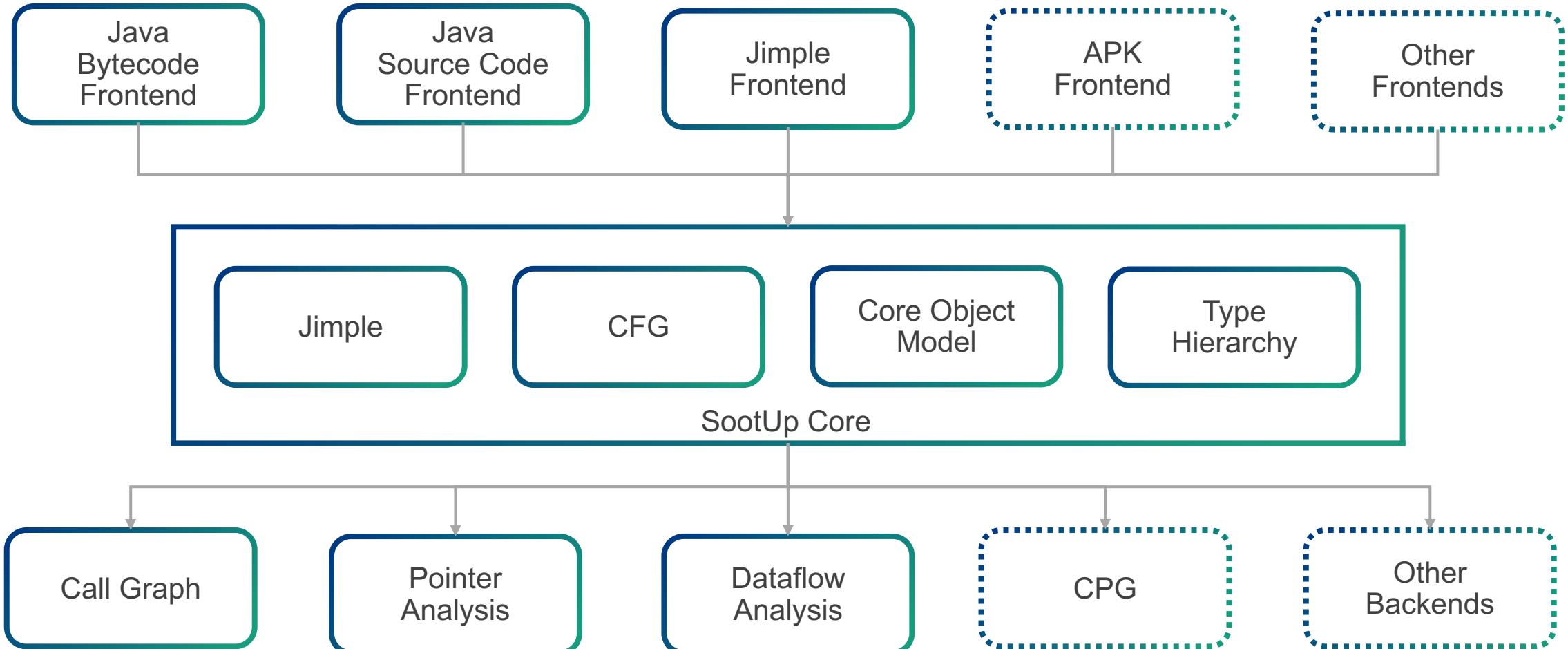
Everything resolved



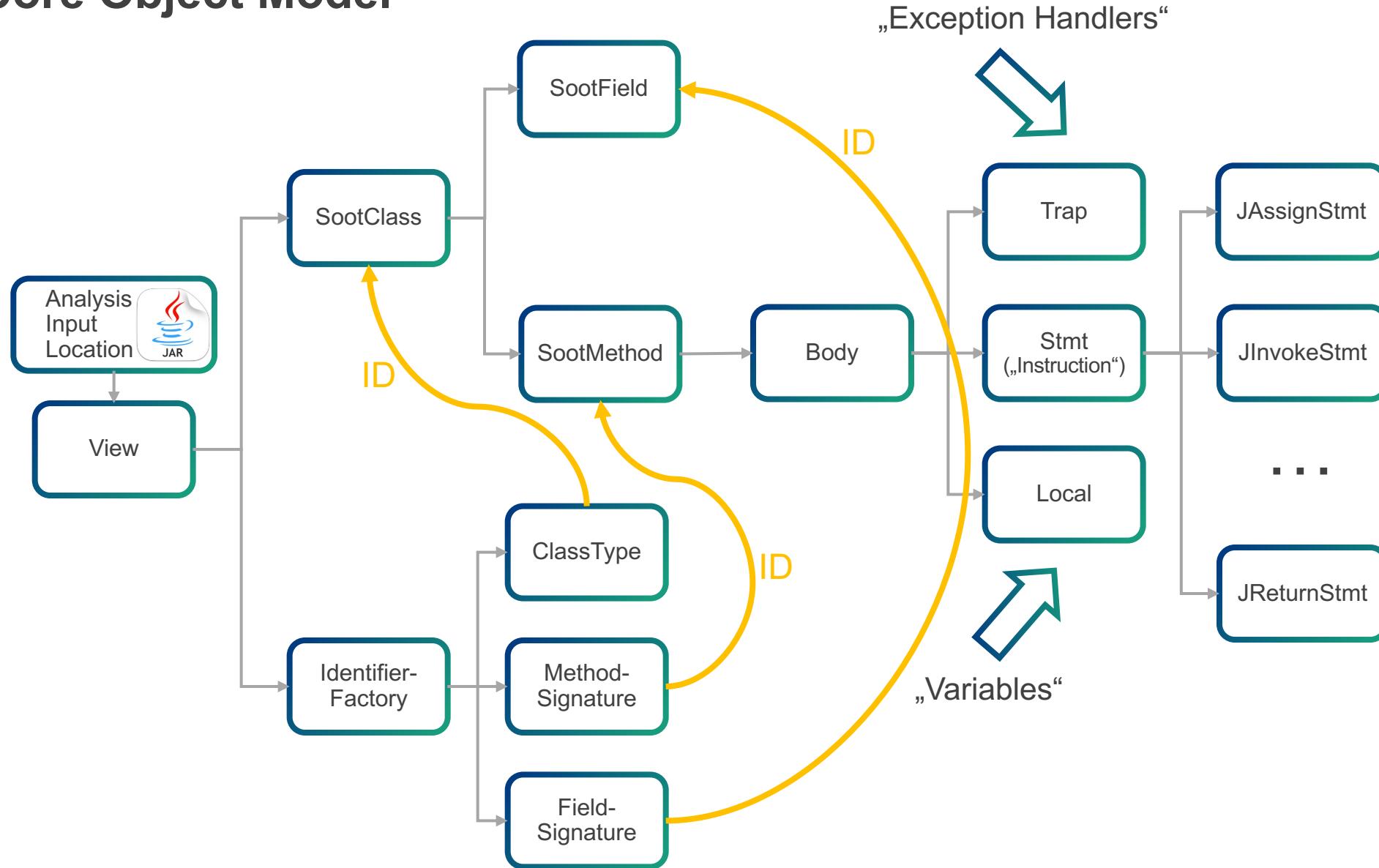
No nesting (3-address)



Architecture and Modules



Core Object Model



Basic API Usage



```
1 AnalysisInputLocation inputLocation =
2         new JavaClassPathAnalysisInputLocation("path2Binary");
3
4 JavaView view = new JavaView(inputLocation);
5
6 JavaClassType classType =
7         view.getIdentifierFactory().getClassType("example.HelloWorld");
8
9 Optional<JavaSootClass> sootClassOpt = view.getClass(classType);
10 JavaSootClass sootClass = sootClassOpt.get();
```

Helpful Resources



SootUp library

This is the home of the [SootUp](#) project. A complete overhaul of the good, old static analysis framework [Soot](#).

What is SootUp

- Transforms JVM bytecode (and other inputs!) to the intermediate representation Jimple.
 - Provides ClassHierarchy generation
 - CallGraph generation with different algorithms/precisions
 - Inter-procedural Data-flow Analysis with the IDE/IFDS framework enabled by [Heros](#)
 - Applies simple transformations on retrieving a methods Body (see [BodyInterceptor](#))

Getting Started	What's SootUp?
Installation	SootUp is a complete overhaul of the good, old static analysis framework Soot .
First Steps	
Analysis Input	
Examples	
Basics	
Jimple IR	<ul style="list-style-type: none">Transforms JVM bytecode to the intermediate representation Jimple.
Jimple Body	<ul style="list-style-type: none">Provides Class-Hierarchy generation
Jimple Statements	<ul style="list-style-type: none">CallGraph generation with different algorithms/precisions
Jimple Types	<ul style="list-style-type: none">Inter-procedural data-flow analysis with the IDE/IFDS framework enabled by Heros
Jimple Values	<ul style="list-style-type: none">Applies simple transformations on retrieving a methods Body (see BodyInterceptor)
	<ul style="list-style-type: none">Provides serialization of the Jimple IR.
Advanced Topics	
BodyInterceptors	
Callgraphs	
BuiltIn Analyses	
How to...	<p> Important</p> <p>SootUp is not a <i>version</i> update to Soot, it is a <i>completely new implementation</i> written from scratch that aims to be a leaner, modernized and developer friendly successor of Soot. It is not a Drop-In Replacement! The new architecture and API renders it not trivial to update existing projects that were built on soot. Therefore we recommend using SootUp for greenfield projects. We hope improved type safety and streamlined mechanisms will aide you implementing and debugging your analysis tool. Unfortunately not every feature has been ported - if you miss something feel free to contribute a feature you miss from good old Soot.</p>
Write a Dataflow analysis	
Incorporate Pointer Analysis	
Misc & More Information	
Announcements	
Design Decisions	
Migration Help	
Contributing	
Code of Conduct	
Code of Silence	

■ Repository

<https://github.com/soot-oss/SootUp>

- Documentation

<https://soot-oss.github.io/SootUp>

- Javadoc

<https://javadoc.io/doc/org.soot-oss/sootup.core>

EXERCISE 1: GET TO KNOW THE API

Exercise 1: Get to know the API

- Task: Extract the following properties from the given Jar:
 1. Number of **classes**
 2. Number of **private methods**
 3. Number of **static fields**
 4. Number of **methods** where the **return type** is the same as its first **parameter type** (if exists)
 5. Number of **assignment statements** in method „*boolean test(java.lang.String)*“ in class „*org.reflections.util.FilterBuilder*“

Steps to access the exercise:

1. git clone <https://github.com/soot-oss/SootUp-Tutorial.git>
2. git checkout 1_Exercise-1

OR

Download from: <https://sootup-tutorial.surge.sh>

Exercise 1: Helpful APIs

AnalysisInputLocation

```
new JavaClassPathAnalysisInputLocation(String path2jar)
```

JavaView

```
new JavaView(AnalysisInputLocation inputLocation)
```

IdentifierFactory

```
getIdentifierFactory()
```

Collection<JavaSootClass>

```
getClasses()
```

Optional<JavaSootMethod>

```
getMethod(MethodSignature sig)
```

IdentifierFactory

ClassType	<code>getClassType(String fullyQualifiedClassName)</code>
-----------	---

MethodSignature	<code>getMethodSignature(ClassType ct, String methodName, String returnType, List<String> parameterTypes)</code>
-----------------	--

JavaSootClass

Optional<JavaSootMethod>	<code>getMethod(String name, Iterable<Type> parameterTypes)</code>
--------------------------	--

Set<JavaSootMethod>	<code>getMethods()</code>
---------------------	---------------------------

Set<JavaSootField>	<code>getFields()</code>
--------------------	--------------------------

JavaSootField

boolean	<code>isStatic()</code>
---------	-------------------------

JavaSootMethod

boolean	<code>isPrivate()</code>
---------	--------------------------

int	<code>getParameterCount()</code>
-----	----------------------------------

Type	<code>getReturnType()</code>
------	------------------------------

Type	<code>getParameterType(int index)</code>
------	--

boolean	<code>hasBody()</code>
---------	------------------------

Body	<code>getBody()</code>
------	------------------------

Body

List<Stmt>	<code>getStmts()</code>
------------	-------------------------

Stmt

Subtypes: `JAssignStmt`, `JInvokeStmt`, `JReturnStmt`, `JIfStmt`, ... (`instanceof`)

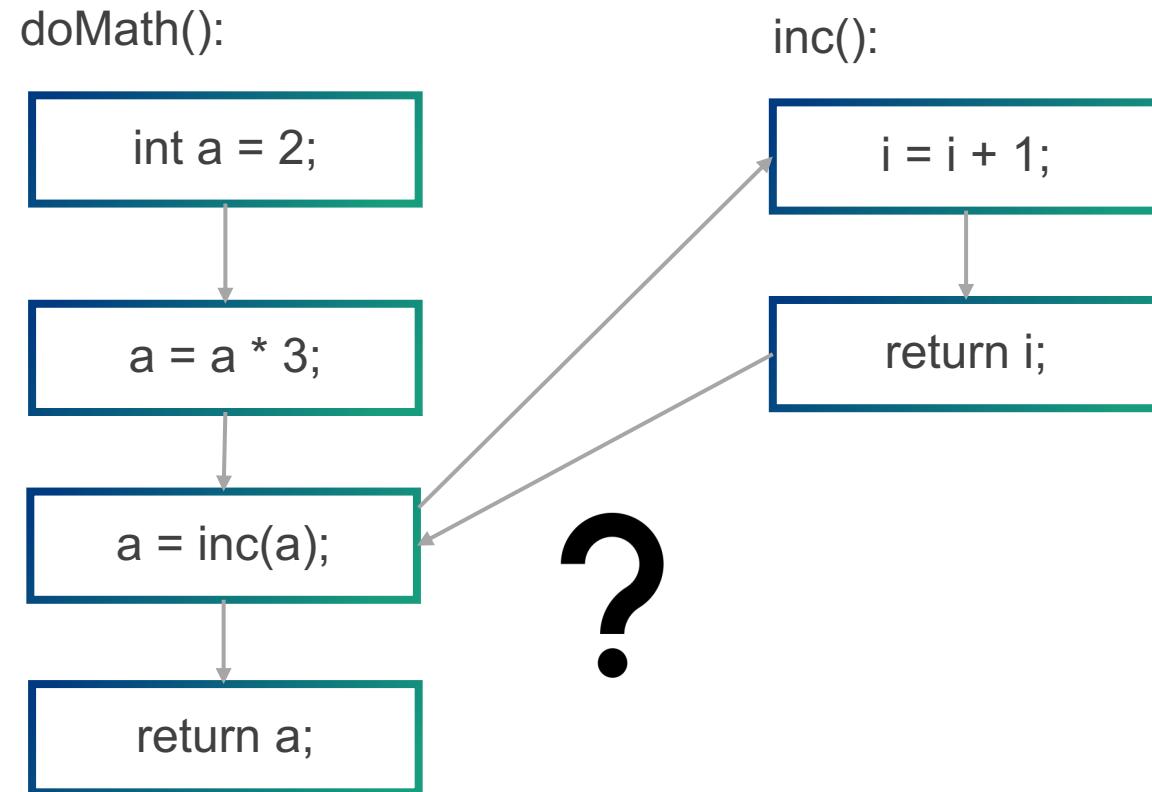
For more help check:

- Documentation: soot-oss.github.io/SootUp
- Javadoc: soot-oss.github.io/SootUp/apidocs

LIVE DEMO: CALL GRAPHS

Interprocedural Analysis

```
public int doMath(){  
    int a = 2;  
    a= a*3  
    a = inc(a);  
    return a  
}  
  
public int inc(int i){  
    return i + 1;  
}
```

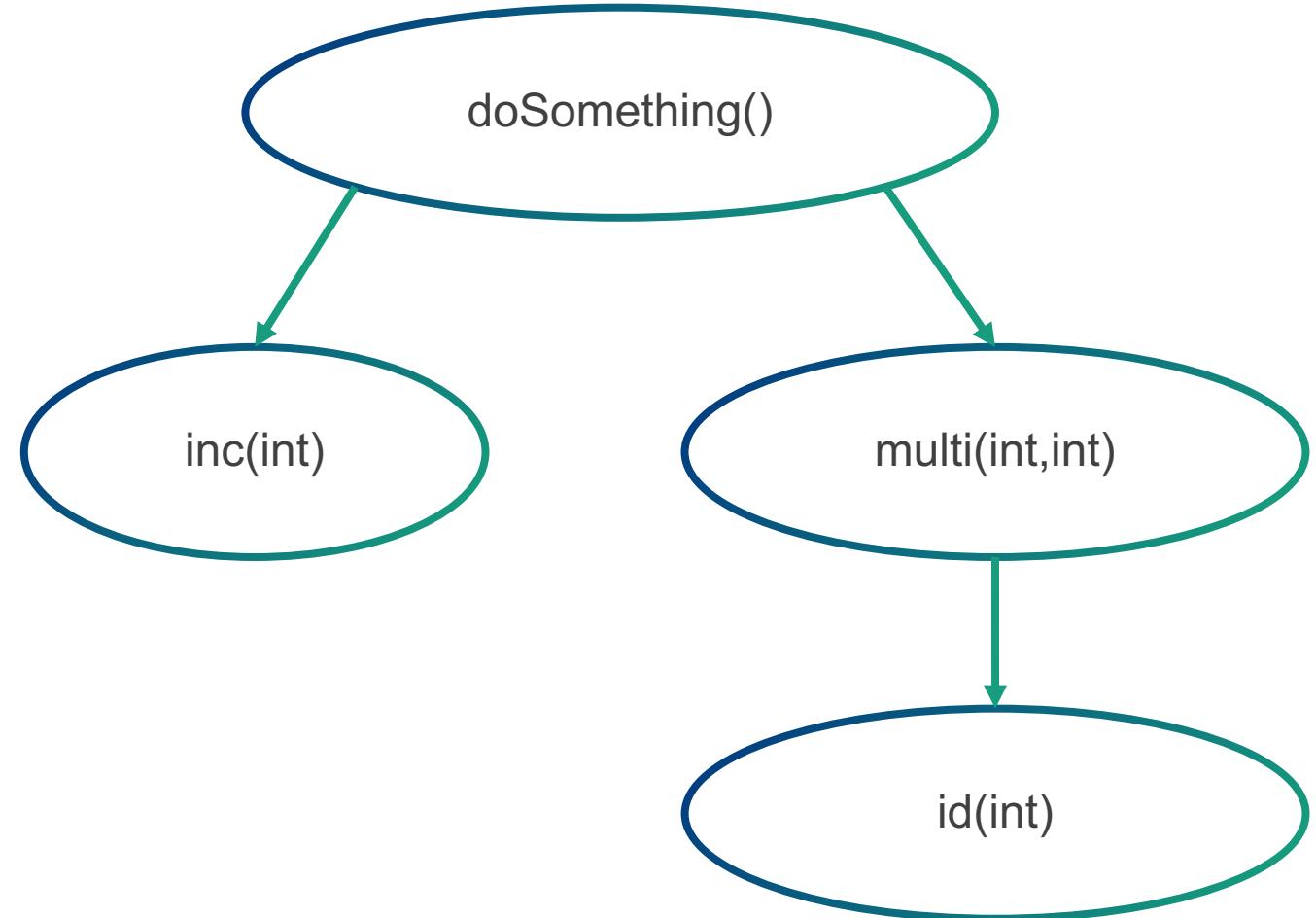


Call Graph

```
public int doSomething(){  
    int a = 2;  
    a = inc(a);  
    a = inc(a);  
    return multi(a,2);  
}
```

```
public int inc(int i){  
    return i + 1;  
}
```

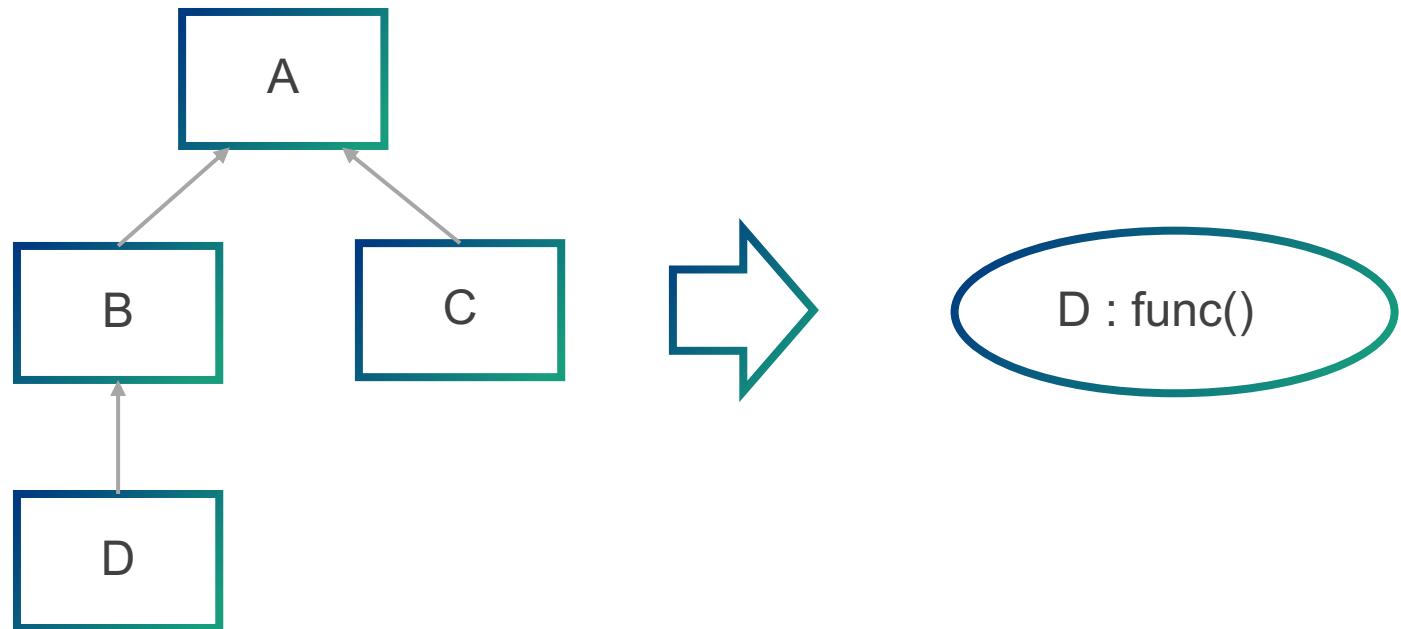
```
public int multi(int i, int d){  
    return id(i) * d;  
}  
public int id(int i){  
    return i;  
}
```



Polymorphism in Java

Virtual invokes

```
public void start (){
    C c = new C();
    A a = new D();
    a.func();
}
```



Supported Call Graph Algorithm

Class Hierarchy Algorithm
(CHA)

Rapid Type Algorithm
(RTA)

Qilin-based Algorithm

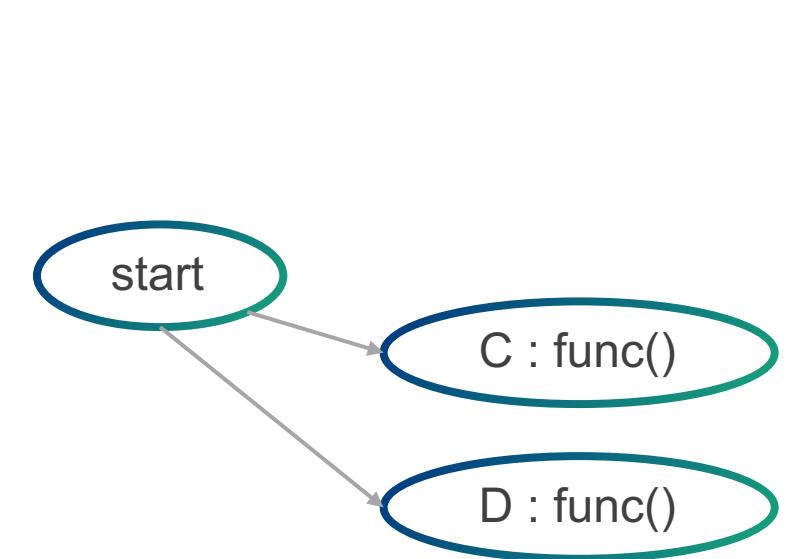
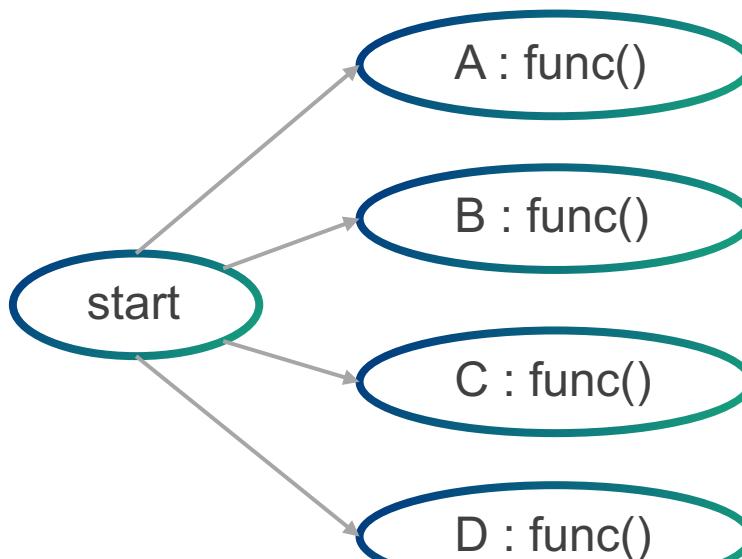
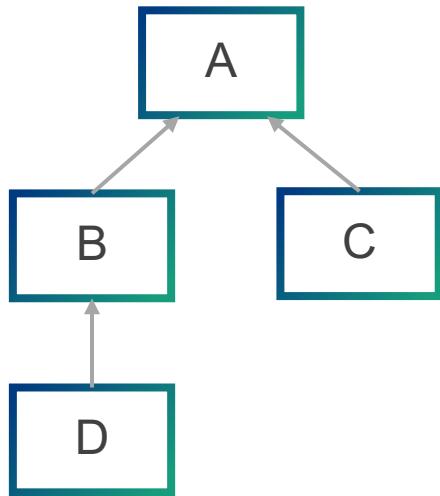
Precision increases

Supported Call Graph Algorithm

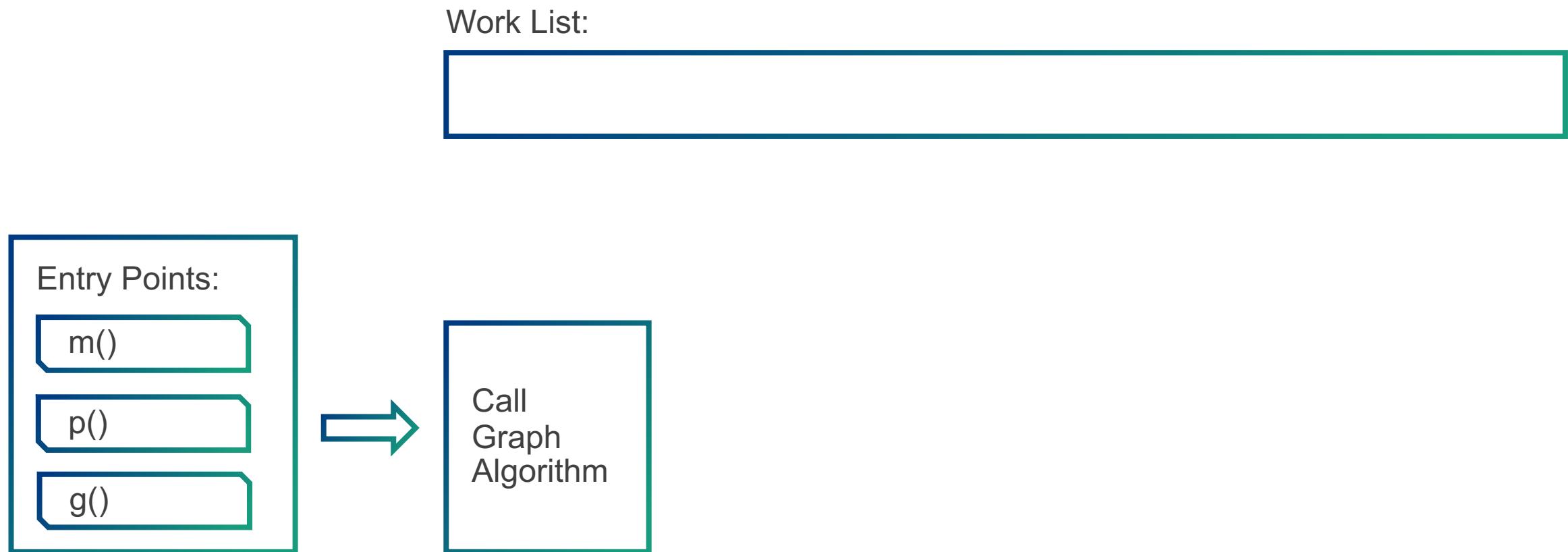
```
public void start (){
    C c = new C();
    A a = new D();
    a.func();
}
```

Class Hierarchy Algorithm
(CHA)

Rapid Type Algorithm
(RTA)



How to Create a Call Graph



How to Create a Call Graph

Work List:

m()

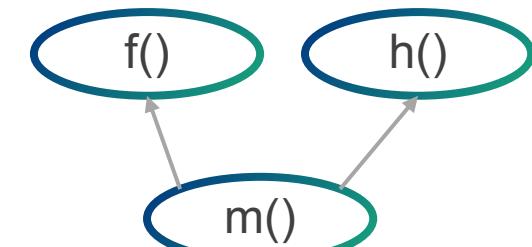
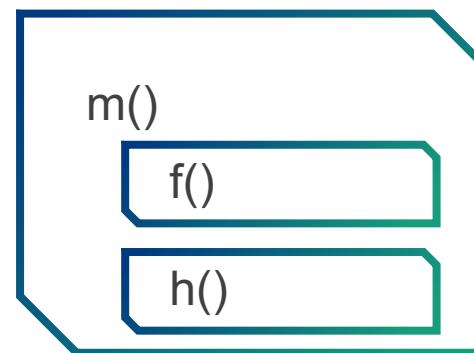
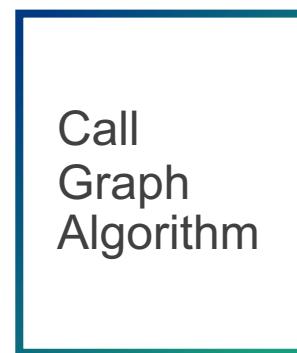
p()

g()

Call
Graph
Algorithm

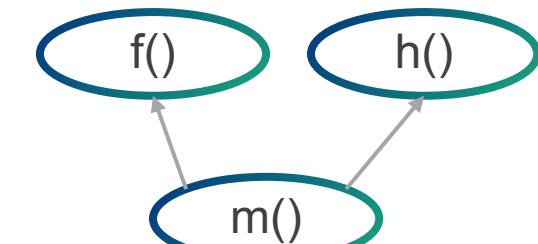
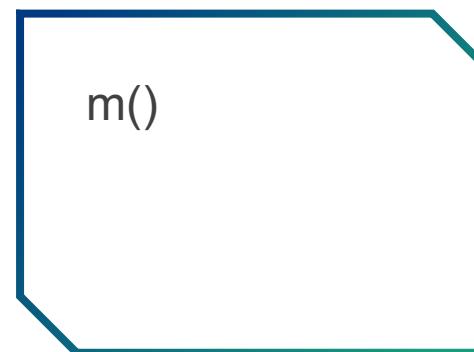
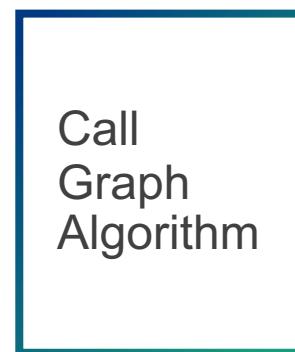
How to Create a Call Graph

Work List:



How to Create a Call Graph

Work List:



Implicit Calls

Static Initializer

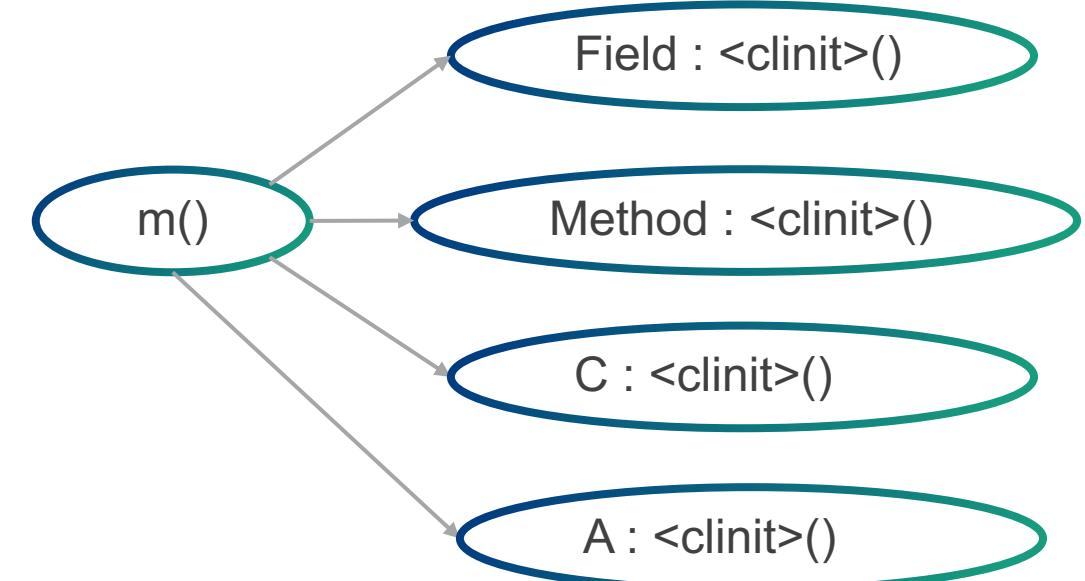
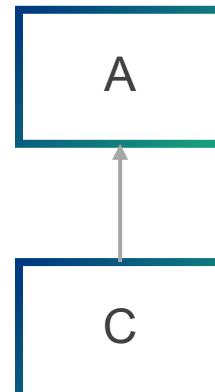
```
public class Field{  
    static int f = doMath();  
}
```



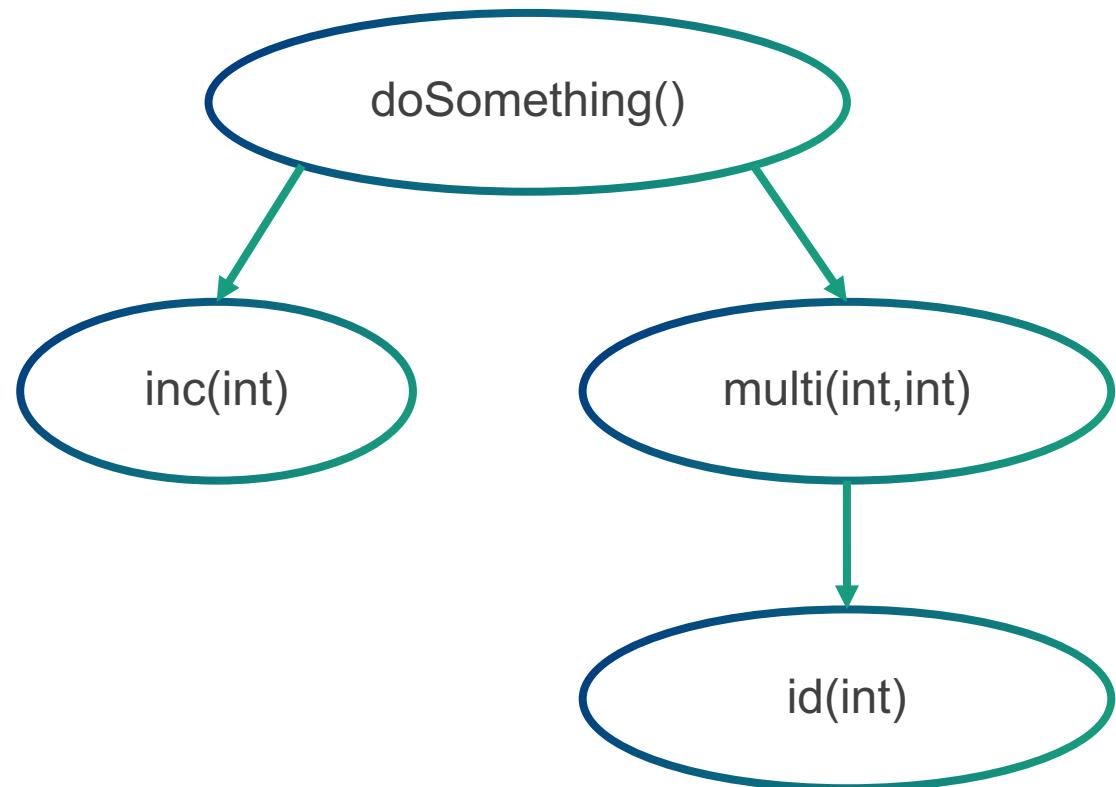
Bytecode:

```
public class Field{  
    static int f;  
    public void <clinit>(){  
        f = doMath();  
    }  
}
```

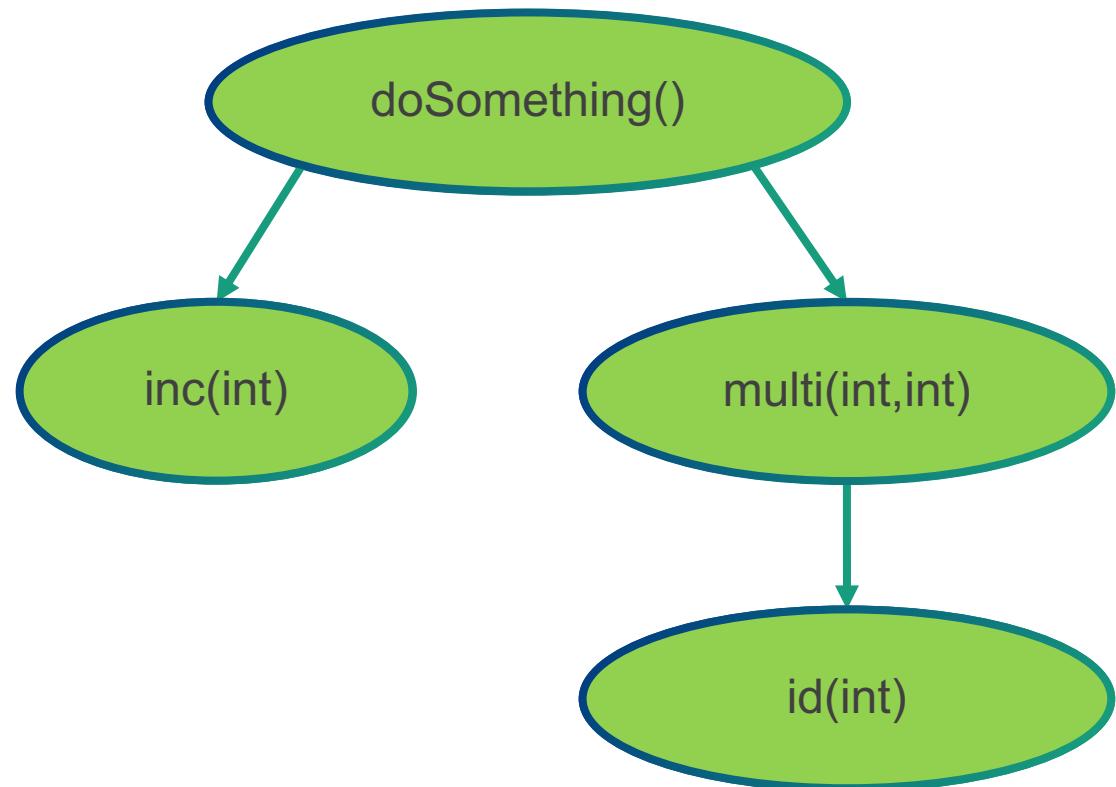
```
public void m (){  
    int a = Field.f;  
    Method.m();  
    A a = new C();  
}
```



Call Graph API

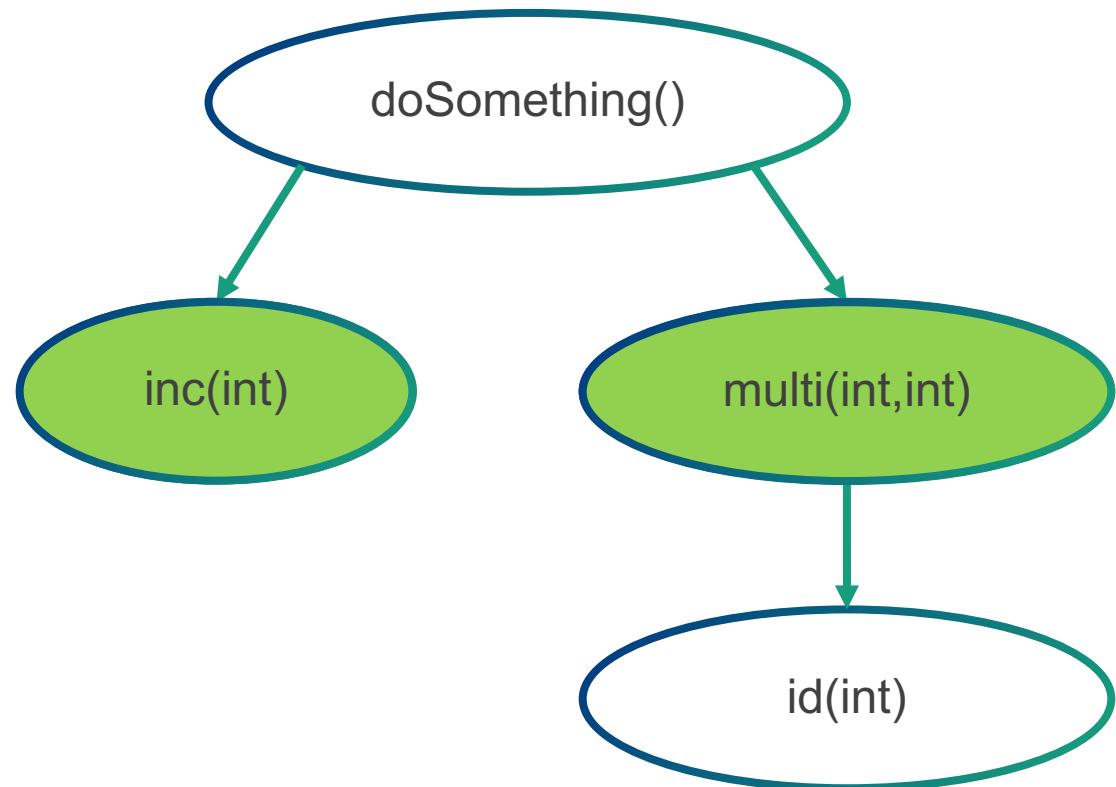


Call Graph API



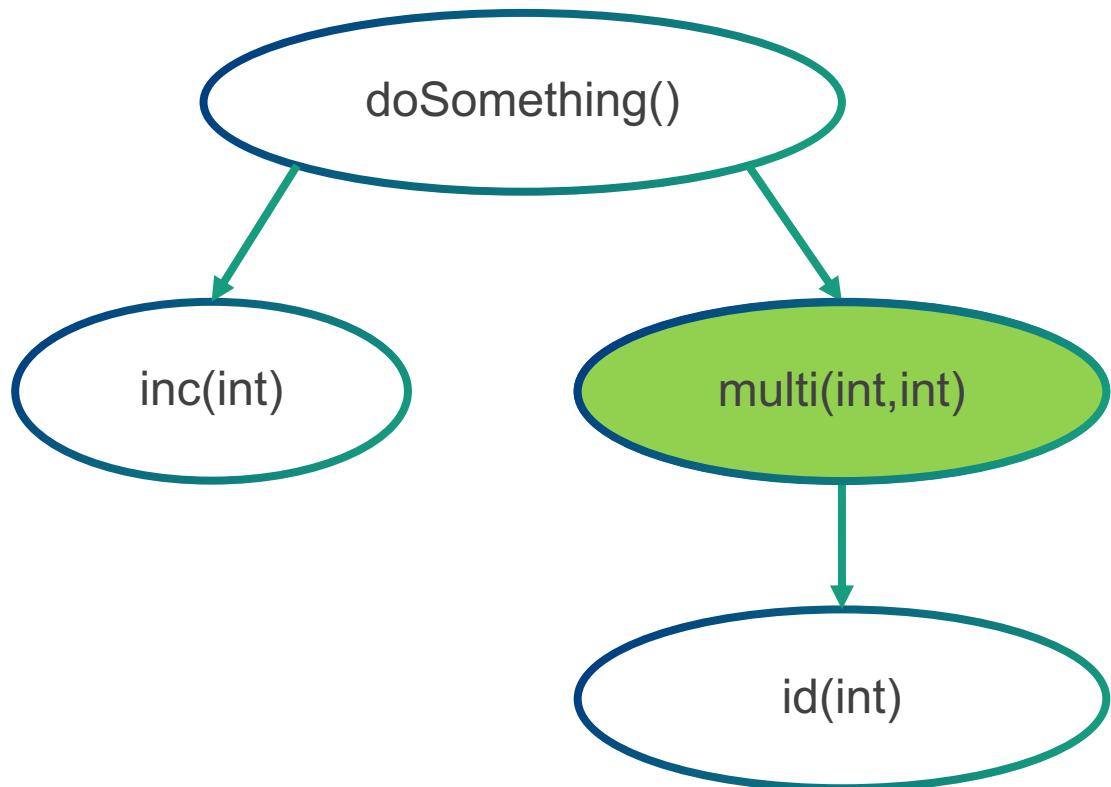
getMethodSignatures() :

Call Graph API



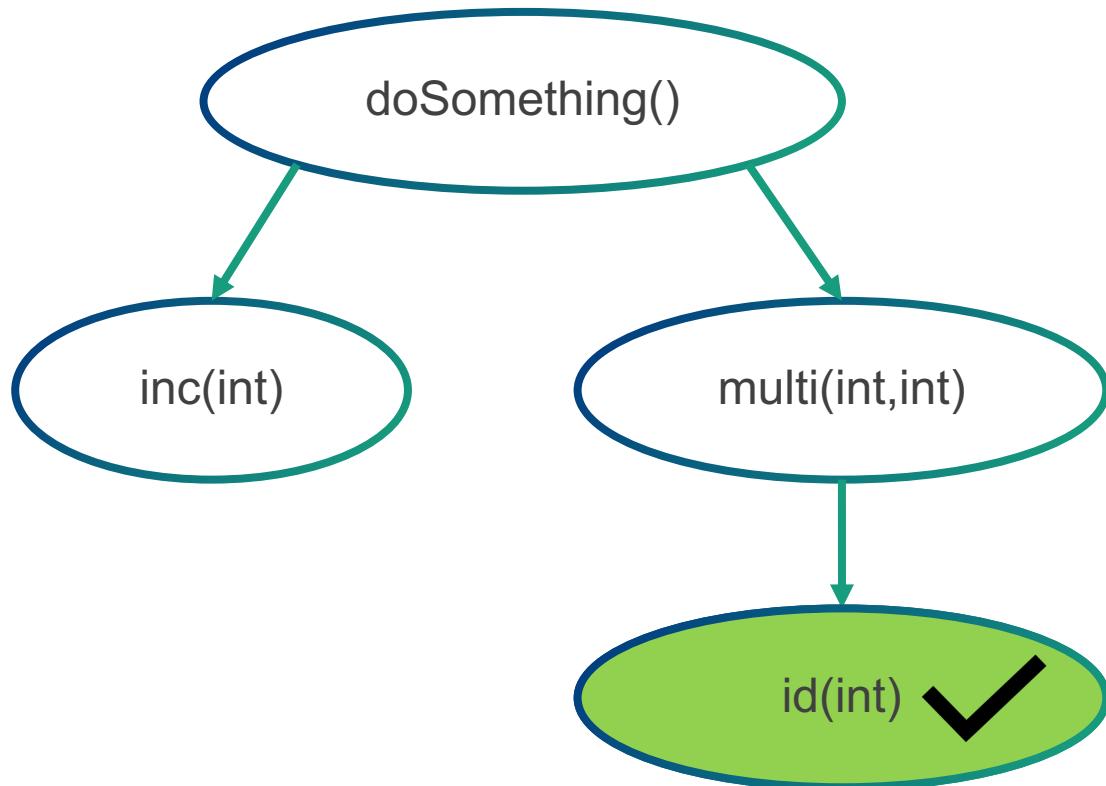
getMethodSignatures()
callsFrom(„doSomething()“)

Call Graph API



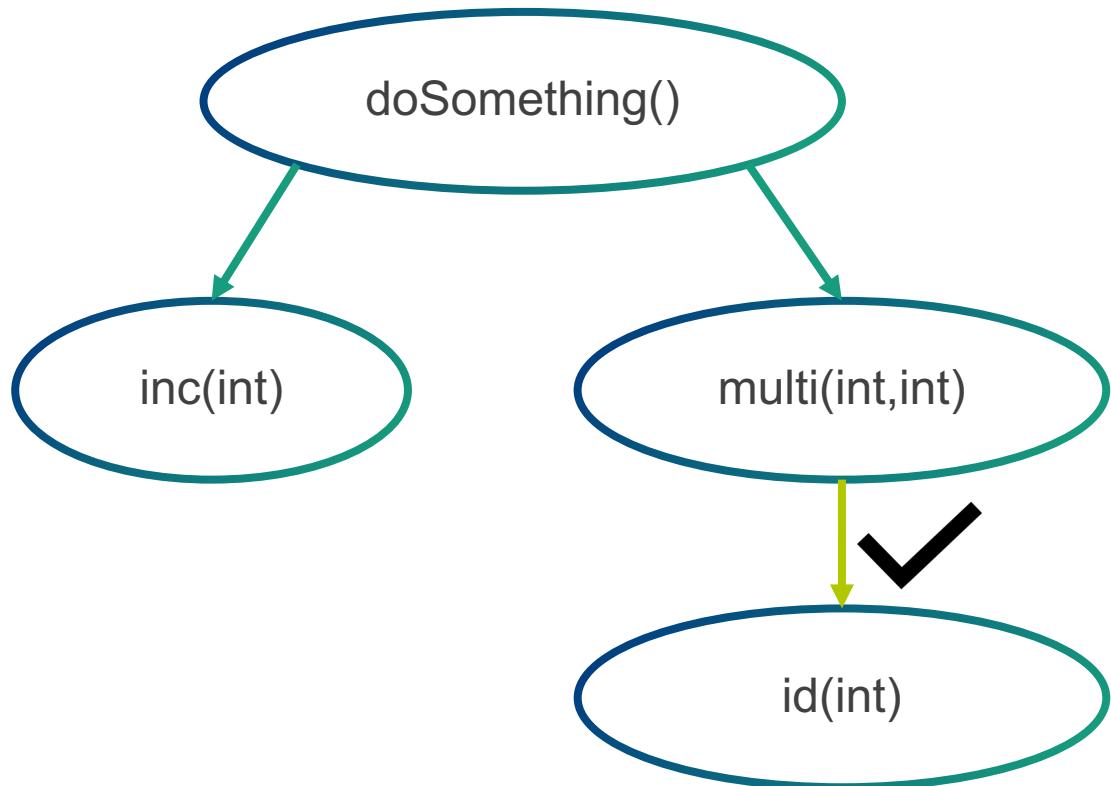
getMethodSignatures()
callsFrom(„doSomething()“)
callsTo(„id(int)“)

Call Graph API



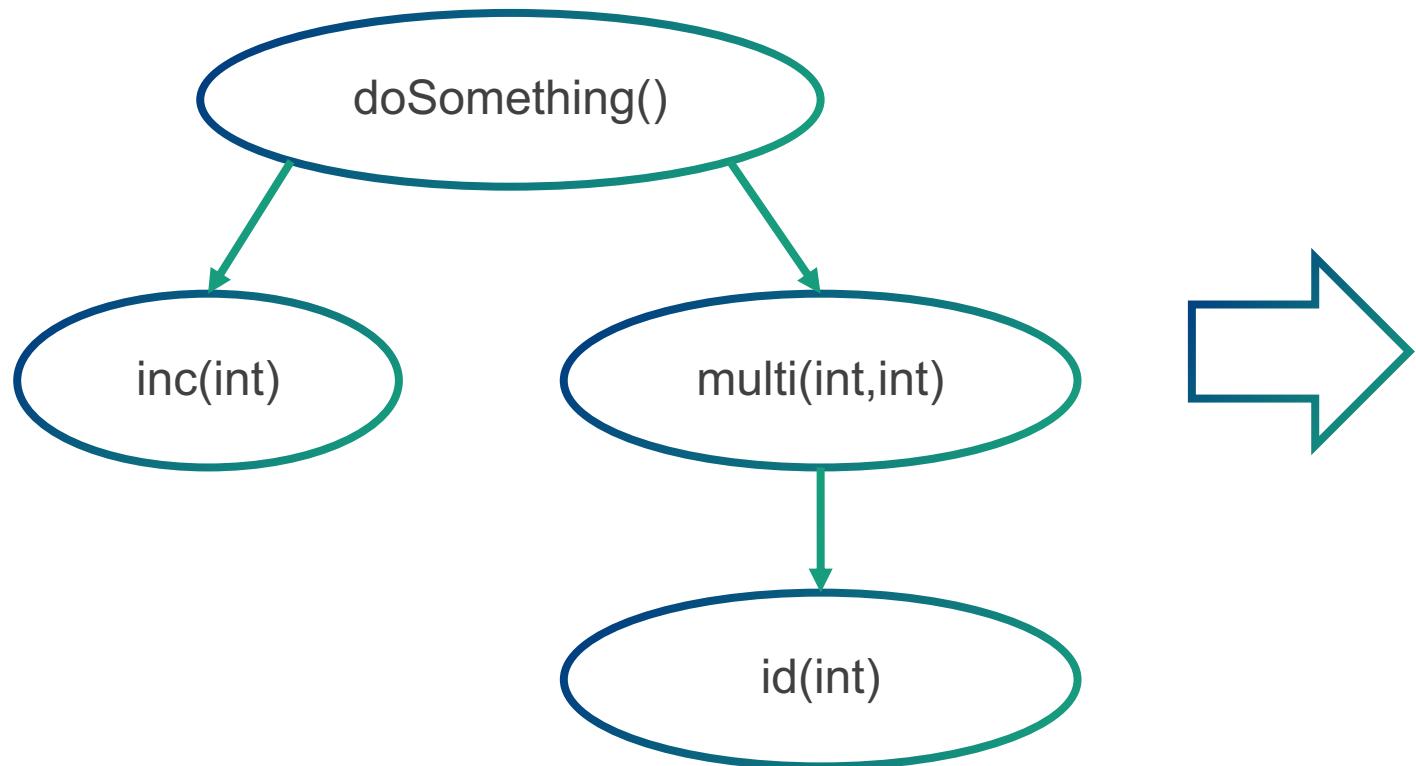
getMethodSignatures()
callsFrom(„doSomething()“)
callsTo(„id(int)“)
containsMethod(„id(int)“)

Call Graph API



getMethodSignatures()
callsFrom(„doSomething()“)
callsTo(„id(int)“)
containsMethod(„id(int)“)
containsCall(„multi(int,int)“, „id(int)“)

Dot Exporter



```
digraph : {  
    doSomething() -> inc(int)  
    doSomething() -> multi(int,int)  
    multi(int,int) -> id(int)  
}
```

EXERCISE 2: TAINT ANALYSIS WITH SOOTUP AND HEROS

Taint Analysis

- A taint is a sensitive information that we want to track throughout the application run.
 - Hardcoded credentials that might leak
 - Untrusted input that might reach critical methods
- Terminology
 - Source:
 - $x = \text{„SECRET“}$
 - Sink:
 - $\text{sink}(x)$

```
void entryPoint() {  
    String a = "SECRET";  
  
    String b = a;  
  
    String c = id(b);  
  
    SinkClass sc = new SinkClass();  
  
    sc.sink(c);  
}
```

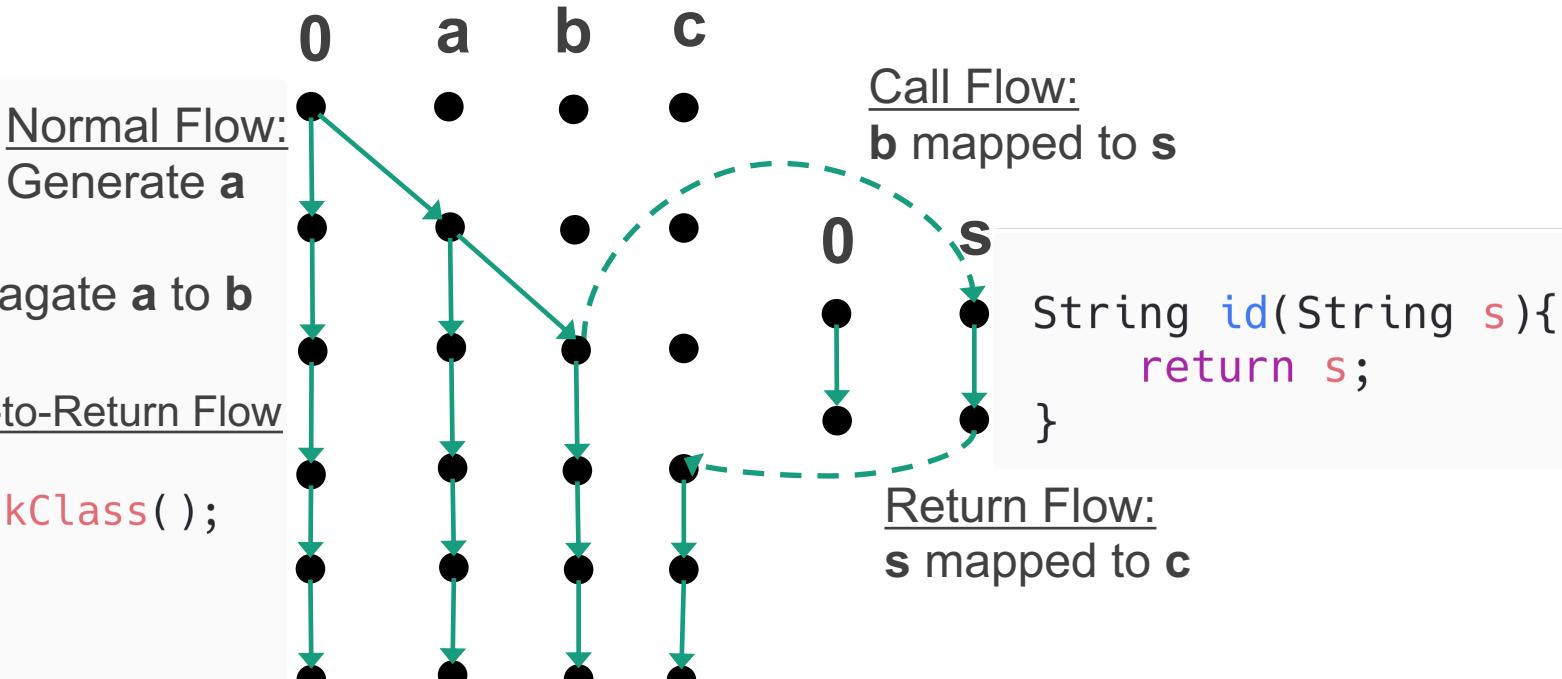
Taint Analysis with SootUp and Heros

- Heros is an IFDS solver
- IFDS framework let's us define data-flow analysis problems:
 - Interprocedural
 - Across method boundaries
 - Flow- and context-sensitive
 - Maintaining order statements
 - Precisely tracking effects in all method contexts
 - Flow Functions to define effects of each statement

Taint Analysis with SootUp and Heros

Flow Functions

```
void entryPoint() {  
    String a = "SECRET";  
  
    String b = a;      Propagate a to b  
  
    String c = id(b); Call-to-Return Flow  
  
    SinkClass sc = new SinkClass();  
  
    sc.sink(c);  
}
```



Exercise 2: Implement a Taint Analysis Problem

- Task: Implement the Flow Functions in TaintAnalysisProblem.java
 - 1. **getNormalFlow()**
 - 1. Generating a Taint: only for this specific assignment, $x = "SECRET"$
 - 2. Propagating (Transferring) a Taint between different variables
 - 2. **getCallFlow()**
 - 1. Mapping call arguments to local parameters, $\text{foo}(x)$
 - 3. **getReturnFlow()**
 - 1. Mapping returned variable to assigned variable, $y = \text{foo}(x)$

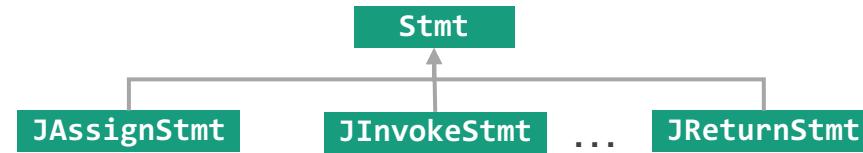
Steps to access the exercise:

1. git clone <https://github.com/soot-oss/SootUp-Tutorial.git>
2. git checkout 3_Exercise-2

OR

Download from: <https://sootup-tutorial.surge.sh>

Exercise 2: Helpful APIs



StringConstant

String	<code>getValue()</code>
--------	-------------------------

JAssignStmt

Value	<code>getLeftOp()</code>
Value	<code>getRightOp()</code>

JReturnStmt

Value	<code>getOp()</code>
-------	----------------------

Gen<D>

`Gen(D genValue, D zeroValue)`

Transfer<D>

`Transfer(D toValue, D fromValue)`

Identity<D>

`Identity<D> v()`

KillAll<D>

`KillAll<D> v()`

For more help check:

- Documentation: soot-oss.github.io/SootUp
- Javadoc: soot-oss.github.io/SootUp/apidocs
- Heros: javadoc.io/doc/de.upb.cs.swt/heros/latest/index.html

CLOSING

Future Work

In Progress

- Android frontend



- Scalable demand-driven pointer analysis

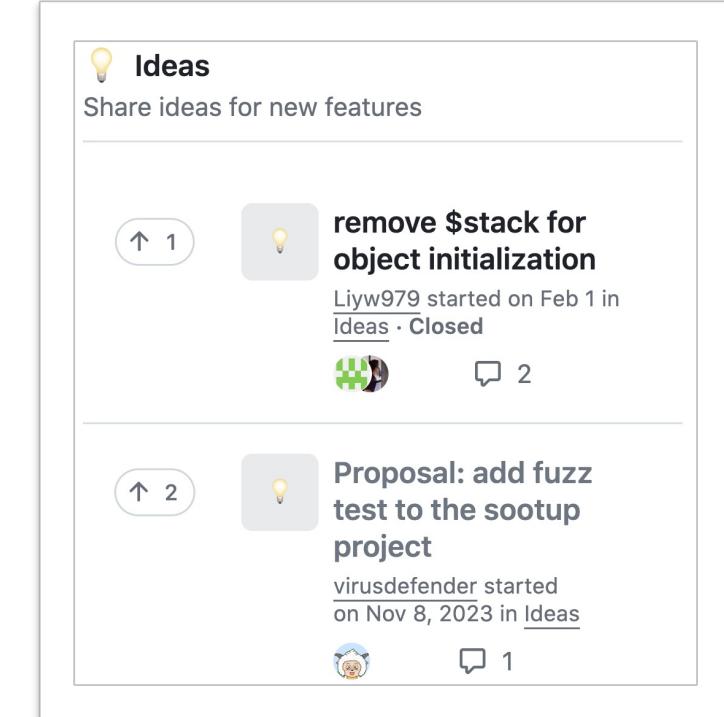


In Near Future

- Frontends for other languages: e.g. Python, Javascript
- Partial code analysis

We are always open to suggestions!

Feel free to propose your ideas in discussions:

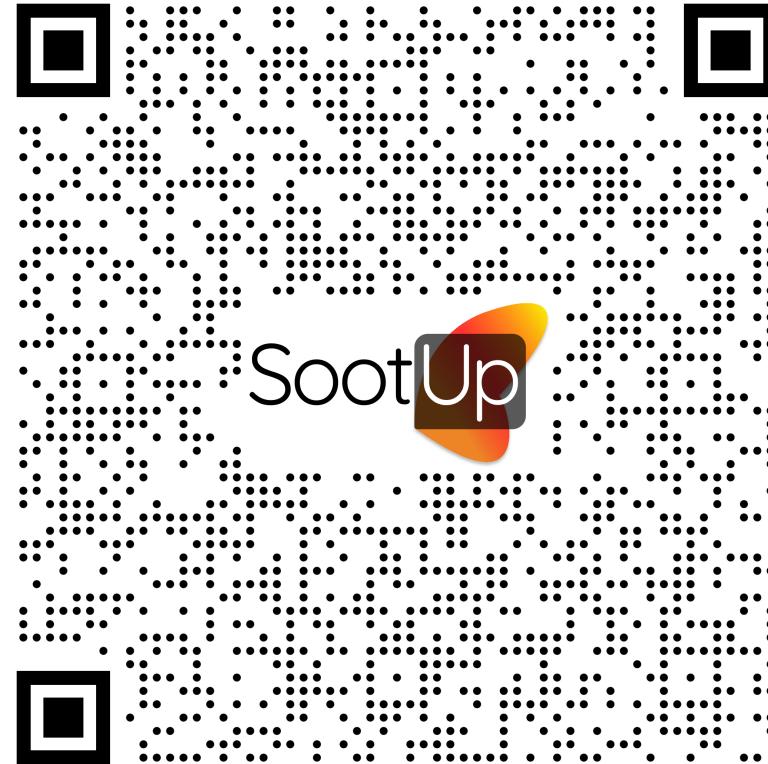


Ideas Share ideas for new features

remove \$stack for object initialization
Liyw979 started on Feb 1 in Ideas · Closed
1 upvote 2 comments

Proposal: add fuzz test to the sootup project
virusdefender started on Nov 8, 2023 in Ideas
2 upvotes 1 comment

SUS Questionnaire



We are looking
forward to see you be
part of the SootUp
community!



<https://github.com/soot-oss/SootUp>

