Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Pooja Rani

Solution

Assignment 08 — 07/11/2018 – v1.0
Static Program Analysis

Please submit this exercise by mail to sma@list.inf.unibe.ch before 14 November 2018, 10:15am.

Installation

Download and unpack this project, then import it as Maven project into your IDE. Run `ReachingDefinitionsAnalysis` to start a reaching definition analysis on the provided `TestClass`. `ReachingDefinitionsAnalysis` shows an example of a dataflow analysis. Its output is stored using the file formats *Jimple* and *Dot* (on non-Windows computers also as *PDF* if you have the `dot` application installed that converts *Dot* files into the *PDF* file format). If the execution fails with an error you probably miss the `dot` application. Nevertheless, you can still visualize the *Dot* files by copying the content of a `*.dot` file into this online renderer.

Exercise

Your task is to assess how *Guava v18.0*, a utility library from Google, makes use of its own APIs. You do this by statically analyzing the *Guava* code and by counting how often each internal method is invoked.

Task 1: Completion of `InternalInvocationAnalysis` (6 Points)

`InternalInvocationAnalysis` counts how often a method of an application is invoked internally, i.e., how many calls it makes to its own methods. Your task is to implement `InternalInvocationAnalysis#process()` that is applied on all methods defined within the analyzed application. Look at the class and its comments to get an idea on what is left to do. If you run `InternalInvocationAnalysis` with an implemented `InternalInvocationAnalysis#process`, you will be shown the 10 most used methods. You might want to look into the Soot wiki to better understand Soot.

a) Submit the code of the `InternalInvocationAnalysis#process` method. **Answer:**

```
for (Unit unit :  body.getUnits()) {
  Stmt stmt = (Stmt) unit;

  if (!stmt.containsInvokeExpr()) {
    continue;
  }

  InvokeExpr invokeExpr = stmt.getInvokeExpr();
  SootMethod method = invokeExpr.getMethod();

  if (!this.isInternal(method.getDeclaringClass())) {
    continue;
  }

  int i = internalInvocations.getOrDefault(method, 0);
  internalInvocations.put(method, i + 1);
}
```

b) Submit the output of the analysis. **Answer:**

```
Analyzing /Users/maenu/.m2/repository/com/google/guava/guava/18.0/guava-18.0.jar
Soot started on Tue Nov 06 14:34:19 CET 2018
Soot finished on Tue Nov 06 14:34:21 CET 2018
Soot has run for 0 min.  2 sec.
872 <com.google.common.base.Preconditions:  java.lang.Object checkNotNull(java.lang.Object)>
162 <com.google.common.base.Preconditions:  void checkArgument(boolean,java.lang.String,java.lang.Object[])>
91 <com.google.common.base.Preconditions:  void checkArgument(boolean)>
75 <com.google.common.collect.Ordering:  com.google.common.collect.Ordering natural()>
57 <com.google.common.net.MediaType:  com.google.common.net.MediaType createConstant(java.lang.String,java.lang.String)>
54 <com.google.common.base.Objects:  boolean equal(java.lang.Object,java.lang.Object)>
49 <com.google.common.io.Closer:  void close()>
46 <com.google.common.cache.LocalCache$Segment:  void postWriteCleanup()>
45 <com.google.common.collect.Multiset$Entry:  java.lang.Object getElement()>
44 <com.google.common.util.concurrent.Monitor:  void leave()>
```

Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Pooja Rani

Task 2: Inspection of results (4 Points)

Inspect the code (and the documentation) of the 10 most used methods in the Guava v18.0 repository with your implementation of `InternalInvocationAnalysis`. Summarize how Guava "eats its own dogfood", i.e., what are the features that Guava uses from itself?

a) Submit a short summary of your inspection results. **Answer:**

According to the output, we can see that Guava most frequently performs precondition null checks (872 occurrences), followed by precondition argument checks with multiple parameters (162 occurrences) and checks with single parameters (91 occurrences). It appears that Guava is mostly busy with itself ensuring that preconditions are met when methods or constructors get called. Furthermore, ordering seems also to be heavily used throughout the library (75 occurrences).