Università della Svizzera italiana

Faculty of Informatics

Bytecode to source code mapper

Extending a Java compiler to provide an accurate positional mapping

Student: Renzo Cotti **Advisor:** Prof. Matthias Hauswirth **Assistant:** Mohammad Reza A.

Problem

Before

```
step 1-3:   int a = m(3) + m(4) + m(5);
```

After

```
step 1:     int a = m(3) + m(4) + m(5);
step 2:     int a = m(3) + m(4) + m(5);
step 3:     int a = m(3) + m(4) + m(5);
step 4:     int a = m(3) + m(4) + m(5);
step 5:     int a = m(3) + m(4) + m(5);
step 6:     int a = m(3) + m(4) + m(5);
```

Currently, if Java code gets debugged, there is no way to know precisely what part of a line is being executed.
This project wants to solve this issue by embedding inside a .class file information about the source code position of every bytecode instruction.
This would allow for further developing of many useful software analysis tools.

Mapping

PC	Bytecode
0	ALOAD_0
1	ICONST_3
2	INVOKEVIRTUAL #23
...	...
16	IADD
17	ISTORE_1

LineNumberTable

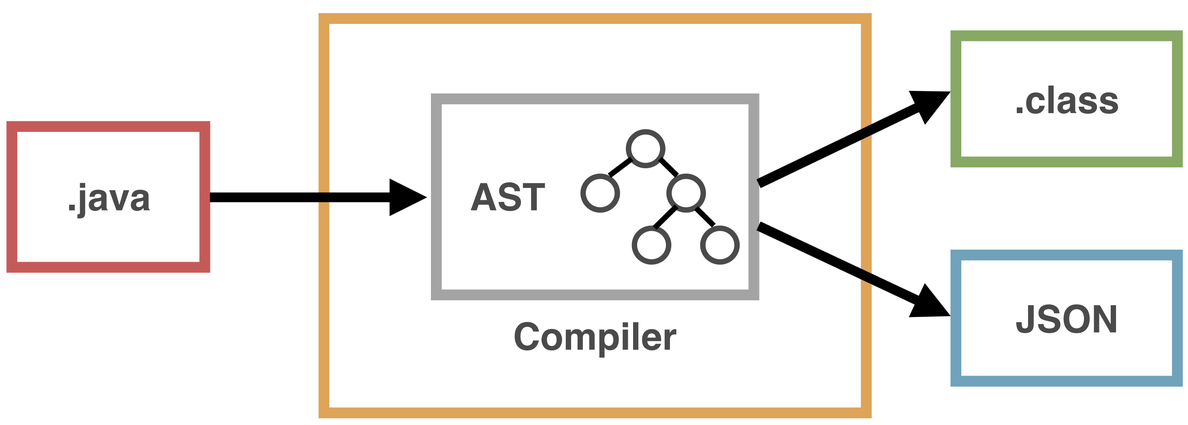
PC	Line
0	1

PositionTable

P	S	S	E	E
0	1	9	1	12
1	1	11	1	11
2	1	9	1	12
...	...	...	...	...
16	1	9	1	26
17	1	5	1	26

As the code gets parsed, we save information about the position inside the Abstract Syntax Tree.
When we write the .class file, we save the positional information of each bytecode inside a new attribute, PositionTable.
To complement this, information about the code structure is written in an external JSON file.

Approach



We decided to modify the code with a top-down approach – working from when ASTNodes get translated into bytecode downwards, so to reduce the amount of refactoring needed.
The idea was to write all information directly into the .class file to ease access to it, however that proved to be a problem for the JSON code structure due to size constraints.
We chose the JSON format for better accessibility to somebody using this project to develop a tool.

Evaluation

Version	Coverage, %	BC Mapped / Total
Java 1.4	91.19	145/159
Java 1.5	91.32	169/185
Java 1.7	92.66	202/218
Java 1.8	90.22	203/225

During this project, we developed a tool to evaluate the progress.
It analyses a produced .class file, and it provides the positions in a readable format, the list of unmapped bytecodes remaining, the overall percentage of coverage and the corresponding source code string for each bytecode.
We tested the system on Java 1.4 to 1.8 and obtained the coverage you see on the left.