CS230 Project Proposal

Team Members:
Zhinan Guan 004118232
Alexandre Wimmers 604680407
Roman Chang 303567574

Overall Description:

We will assess the effectiveness of a Java static-analyzer tool called FindBugs, implemented in the cited paper. It claims to use simpler static analysis techniques such as class hierarchy analysis, linear code scan, control-flow graph, and some dataflow analysis. (Nothing like abstract interpretation or model checking.)

There are 423 different bug patterns reported by FindBugs 3.0.1, the most recent version. We will individually assess as many of these as is reasonable within our time frame. We will prioritize assessment of patterns across multiple categories rather than assessments of multiple patterns within a single category.

Our large-scale assessment will be on Apache Tomcat. We will run the source code through FindBugs, and categorize the warning it generates, and look for false positive and false negative warnings.

Assessment criteria:

We have two criteria for assessment:

- Soundness - For each bug pattern assessed, create an example that triggers the bug pattern despite being a perfectly legitimate use case of 'not a bug'. (False positives.) An example is shown on the next page.
- Completeness - For each bug pattern assessed, attempt to create an example that should trigger the bug pattern but does not. (False negatives.)

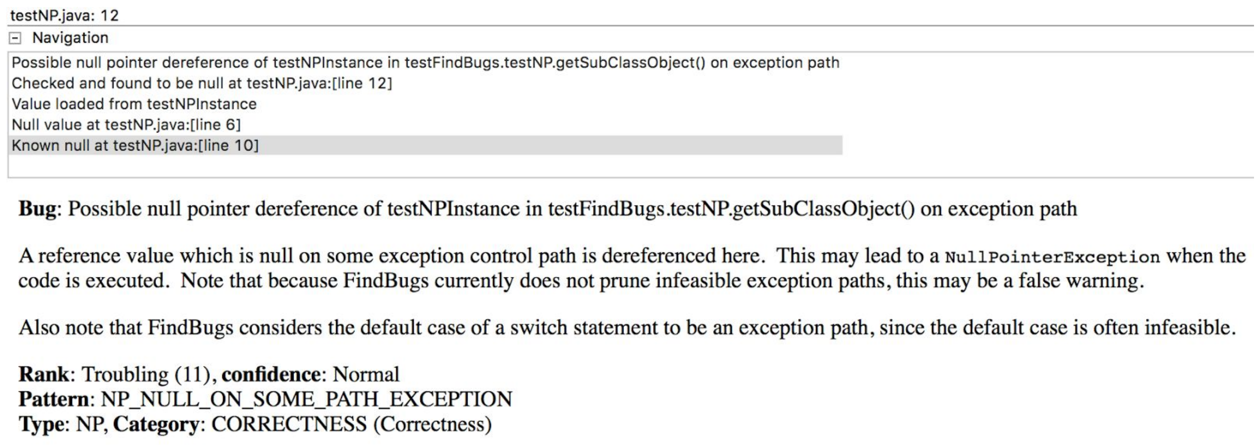
References:

David Hovemeyer and William Pugh. 2004. Finding bugs is easy. SIGPLAN Not. 39, 12 (December 2004), 92-106. DOI=http://dx.doi.org/10.1145/1052883.1052895

Example of False Positive by FindBugs:

```
public class testNP {
        public testNP getSubClassObject() {
                testNP testNPInstance = null;
                try {
                        testNPInstance = this.getClass().newInstance();
                }
                catch(Exception e) {
                }
                if (testNPInstance == null) {
                        throw new NullPointerException("pointer is null at
testNP.getSubClassObject()");
                }
                return testNPInstance;
        }
}
```

`testNPInstance` is never dereferenced, however, FindBugs generates the following bug:



testNP.java: 12

Navigation

```
Possible null pointer dereference of testNPInstance in testFindBugs.testNP.getSubClassObject() on exception path
Checked and found to be null at testNP.java:[line 12]
Value loaded from testNPInstance
Null value at testNP.java:[line 6]
Known null at testNP.java:[line 10]
```

Bug: Possible null pointer dereference of testNPInstance in testFindBugs.testNP.getSubClassObject() on exception path

A reference value which is null on some exception control path is dereferenced here. This may lead to a `NullPointerException` when the code is executed. Note that because FindBugs currently does not prune infeasible exception paths, this may be a false warning.

Also note that FindBugs considers the default case of a switch statement to be an exception path, since the default case is often infeasible.

Rank: Troubling (11), **confidence**: Normal
Pattern: NP_NULL_ON_SOME_PATH_EXCEPTION
Type: NP, **Category**: CORRECTNESS (Correctness)