Nullable Method Detection

Don’t Repeat The Mistakes Others Have Already Fixed

Manuel Leuenberger

Master thesis

Problem

Terms terms = fields.terms(field);
TermsEnum termsEnum = terms.iterator();
Problem

Terms terms = fields.terms(field);
TermsEnum termsEnum = terms.iterator();

NPE!
Problem

Terms terms = fields.terms(field);
TermsEnum termsEnum = terms.iterator();
Related Work
Related Work

null check is most frequent bug-fix pattern
Related Work

null check is most frequent bug-fix pattern

checked values are mostly method return values
Related Work

null check is most frequent bug-fix pattern

checked values are mostly method return values

dereference verification by static analysis
Related Work

null check is most frequent bug-fix pattern

checked values are mostly method return values

dereference verification by static analysis

false-positives due to infeasible paths
Idea

Don’t Repeat The Mistakes Others Have Already Fixed

empirical evidence instead of verification
Goal

Terms terms = fields.terms(field);
TermsEnum termsEnum = terms.iterator();
Goal

77% (535/699) checked for null

```java
Terms terms = fields.terms(field);
TermsEnum termsEnum = terms.iterator();
```
Method UUID

- lookup requires method UUID
- method contracts may evolve
- (class, method) not universally unique

→ (Maven artifact, class, method) is unique

```java
groupId: org.apache.lucene
artifactId: lucene-core
version: 6.2.0
class: o.a.l.i.Fields
method: o.a.l.i.Terms terms(j.l.String)
```
Nullable Method Detection

`nullable method := “callers check return value for null”`
Nullable Method Detection

nullable method := “callers check return value for null”

Terms terms = fields.terms(field);
if (terms == null) {
   // this can happen [...]
   continue;
}
TermsEnum termsEnum = terms.iterator();
Nullable Method Detection

\[ \text{nullable method} := \text{“callers check return value for null”} \]

Terms terms = fields.terms(field);
if (terms == null) {
    // this can happen[...]
    continue;
}
TermsEnum termsEnum = terms.iterator();
Nullable Method Detection

nullable method := “callers check return value for null”

```java
Terms terms = fields.terms(field);
if (terms == null) {
    // this can happen[...]
    continue;
}
TermsEnum termsEnum = terms.iterator();
```
Nullable Method Detection

nullable method := “callers check return value for null”

```java
org.apache.lucene.index.Fields
org.apache.lucene.index.Terms terms(java.lang.String)

Terms terms = fields.terms(field);
if (terms == null) {
    // this can happen[...]
    continue;
}
TermsEnum termsEnum = terms.iterator();
```
Usage of **Apache** Artifacts
Usage of **Apache** Artifacts
Usage of **Apache** Artifacts

- rhino
- xmlbeans
- neo4j
- maven-core
- spring-core
- doxia
- lucene-core
- tomcat
- juli
- trick
Architecture

```
groupId LIKE '%apache%'
```

JMS
- input queue
- dependent queue
- output queue

Spring Batch
- match job
- dependent job
- analysis job

Soot
- analysis

Neo4j
Results

• analyzed
  • ~45’000 artifacts
  • ~5’000’000 classes
  • ~37’000’000 methods
• found
  • ~150’000’000 invocations
  • ~10’000’000 null checks
nullable method distribution, #method = 19363, #invocation >= 100

\[
\text{ratio} := \frac{\#\text{checked}}{\#\text{invocation}}
\]
nullable method distribution, \#method = 5342, \#invocation \geq 100, \(0 < \text{ratio} < 1\)
nullable method distribution, #method = 19363, #invocation >= 100

- ratio = 0: 69.8%
- 0 < ratio < 1: 27.6%
- ratio = 1: 2.6%
Limitations

- only ~20’000 methods with >100 invocations
- analysis only detects single expression checked for null in if statements

```java
// not detected
Objects.notNull(value)
if (value instanceof String) {
    if (value1 > 0 && value2 != null) {
```
Conclusion

• Focussed API usage analysis
  • build thousands of class-paths per hour
  • analyze hundreds of artifacts per hour
• 70% of methods are never checked
• 27% of methods are sometimes checked
• 3% of methods are always checked

→ only invocations of 27% of methods need further analysis
Future Work

• IDE plugin that looks up invocations on nullability

• analyze whole Maven repositories

• intra-procedural: detect bugs

• inter-procedural: prune call-graphs with knowledge about nullable methods
Lessons Learned

• analyzing binaries with Soot is easy, and you get precise types for free

• JMS works well as stream buffers

• Neo4j scales, writing with kernel is nice, querying with cypher is ok, query planner is not always clever, transactions have isolation issues

• Spring is still not a close friend

• still love Java 8 streams

• crunching numbers and plotting with R is surprisingly easy
References


