# Evaluating NonDex for Modern Java Ecosystem

Kaiyao Ke (kaiyaok2@illinois.edu), Darko Marinov (marinov@illinois.edu)
University of Illinois Urbana-Champaign

FTW @ ICSE 2025

## Implementation-Dependent (ID) Flaky Tests

- Code that assumes a **deterministic implementation** of **non-deterministic specification (ADINS)**

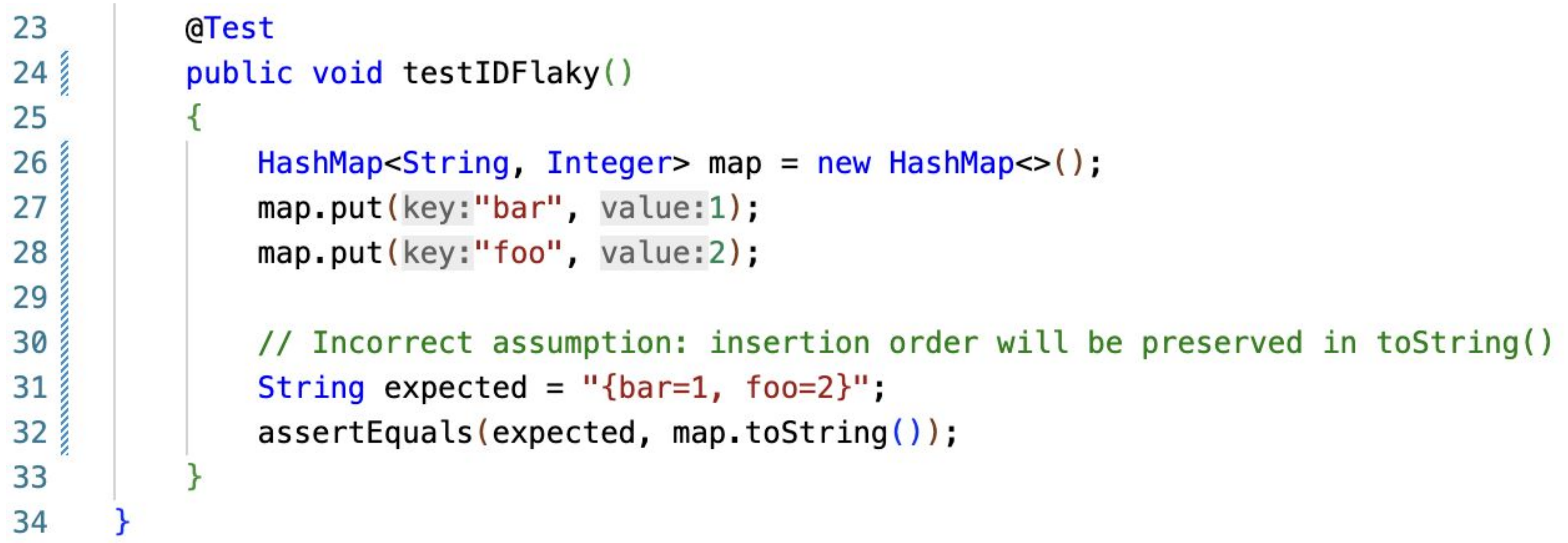
- Java API's with non-deterministic specification

  *java.lang.Class#getDeclaredFields;*
  *java.util.HashMap#entrySet;*

  ....

- But "look deterministic" under specific implementation

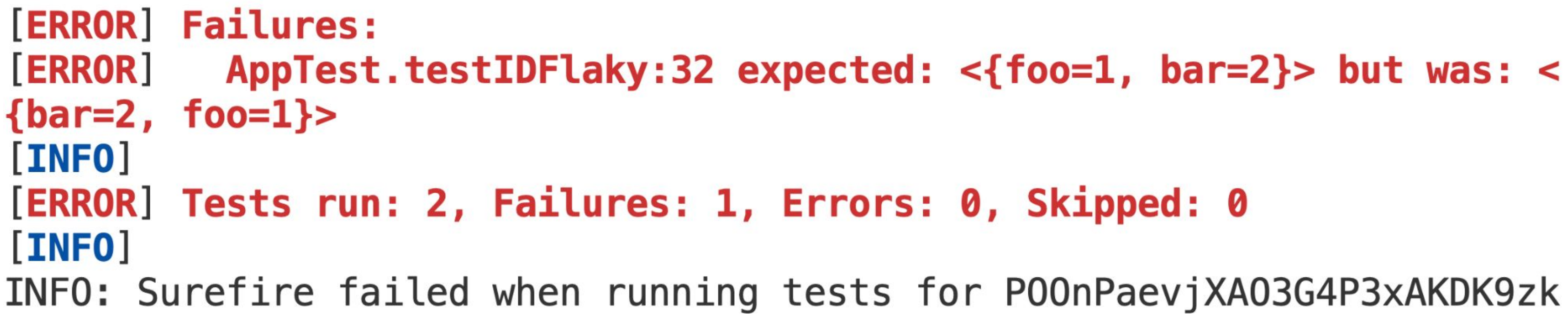
## Example: Simplified version of a Test from HubSpot/jinjava



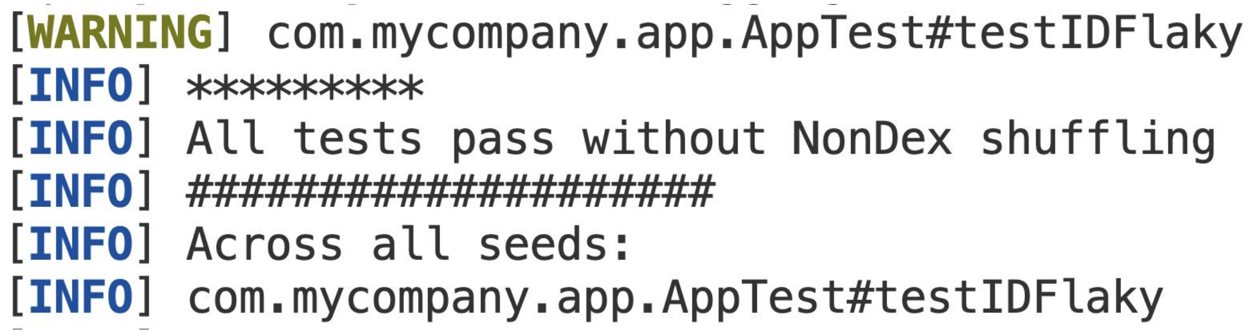
```
    @Test
    public void testIDFlaky()
    {
        HashMap<String, Integer> map = new HashMap<>();
        map.put(key:"bar", value:1);
        map.put(key:"foo", value:2);

        // Incorrect assumption: insertion order will be preserved in toString()
        String expected = "{bar=1, foo=2}";
        assertEquals(expected, map.toString());
    }
}
```

## Existing Solution: NonDex (Detector)

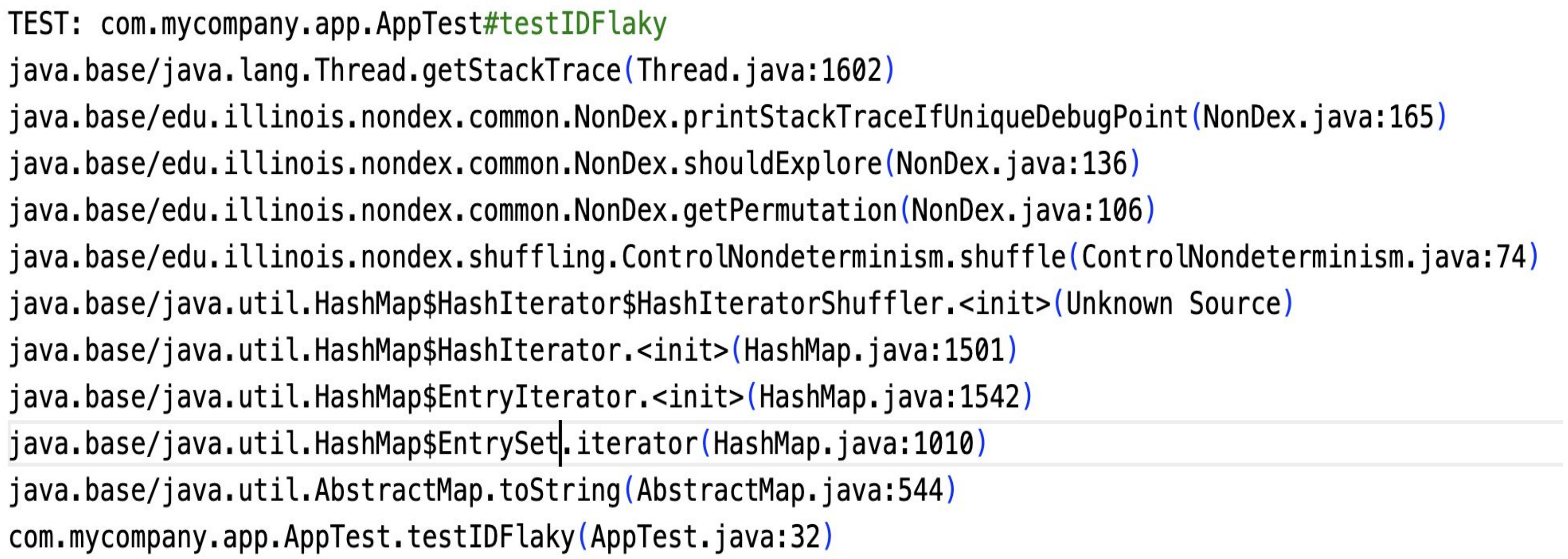


```
[ERROR] Failures:
[ERROR]   AppTest.testIDFlaky:32 expected: <{foo=1, bar=2}> but was: <
{bar=2, foo=1}>
[INFO]
[ERROR] Tests run: 2, Failures: 1, Errors: 0, Skipped: 0
[INFO]
INFO: Surefire failed when running tests for POOnPaevjXAO3G4P3xAKDK9zk
```



```
[WARNING] com.mycompany.app.AppTest#testIDFlaky
[INFO] *********
[INFO] All tests pass without NonDex shuffling
[INFO] ####################
[INFO] Across all seeds:
[INFO] com.mycompany.app.AppTest#testIDFlaky
```

## Existing Solution: NonDex (Debugger)



```
TEST: com.mycompany.app.AppTest#testIDFlaky
java.base/java.lang.Thread.getStackTrace(Thread.java:1602)
java.base/edu.illinois.nondex.common.NonDex.printStackTraceIfUniqueDebugPoint(NonDex.java:165)
java.base/edu.illinois.nondex.common.NonDex.shouldExplore(NonDex.java:136)
java.base/edu.illinois.nondex.common.NonDex.getPermutation(NonDex.java:106)
java.base/edu.illinois.nondex.shuffling.ControlNondeterminism.shuffle(ControlNondeterminism.java:74)
java.base/java.util.HashMap$HashIterator$HashIteratorShuffler.<init>(Unknown Source)
java.base/java.util.HashMap$HashIterator.<init>(HashMap.java:1501)
java.base/java.util.HashMap$EntryIterator.<init>(HashMap.java:1542)
java.base/java.util.HashMap$EntrySet.iterator(HashMap.java:1010)
java.base/java.util.AbstractMap.toString(AbstractMap.java:544)
com.mycompany.app.AppTest.testIDFlaky(AppTest.java:32)
```

## Existing Solution: NonDex

- **Instrumentation engine:**

  Modifies the classes in the standard library to add code for random exploration

- **Runner:**

  Controls behavior on instrumented library (e.g., seed, invocations to “permute”)

- **Detector:**

  Executes tests a specified number of times to explore different behaviors

- **Debugger:**

  Identifies the API invocation(s) where a wrong assumption was made.

## 2 user-facing phases

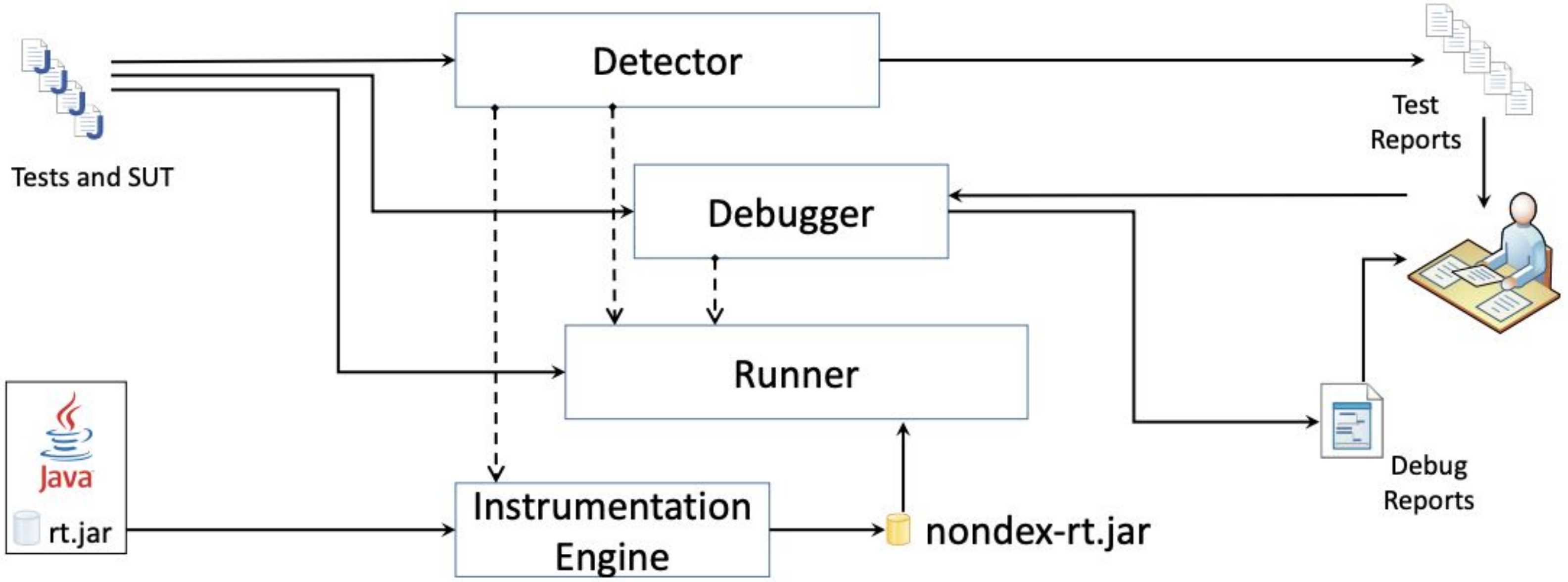
- **Detection:**

  Find tests that pass initially but fail when NonDex explores different allowed behaviors

- **Debugging:**

  Identify the invocation making such assumptions (binary search).

## NonDex v1.0.0 Architecture



## Limitations in the Original NonDex Tool

1. Supporting only Java version 8
2. Supporting only Maven
3. Debugger not reporting all false assumptions

## Our Contributions

- **Incremental Contribution to NonDex:**

  3 major improvements

- **Large-Scale study of ID Tests:**

  With interesting case studies

## Our 1st Improvement: Support Multiple Java versions

- **Handling JPMS Introduced from Java 9+:**

  Use JRT filesystem to extract runtime classes

- **Postponing NonDex Instrumentation:**

  Java 9+ loads standard library classes earlier

- **Minor Changes**

  Custom logging classes, adapt to newer ASM versions, ...

## Our 2nd Improvement: Integration with Gradle

- **Same Workflow as the Maven plugin:**

  3 Tasks: nondexTest; nondexDebug; nondexClean

- **Gradle Runner**

  Integrate NonDex-instrumented standard libraries

  and packaged nondex-common.jar to JVM boot path.

- **Released to Gradle Plugin Portal**

## Our 3rd Improvement: Enhanced Functionality

- **Debugger Reports Multiple Invocations:**

  Searches for all individual ADINS invocations

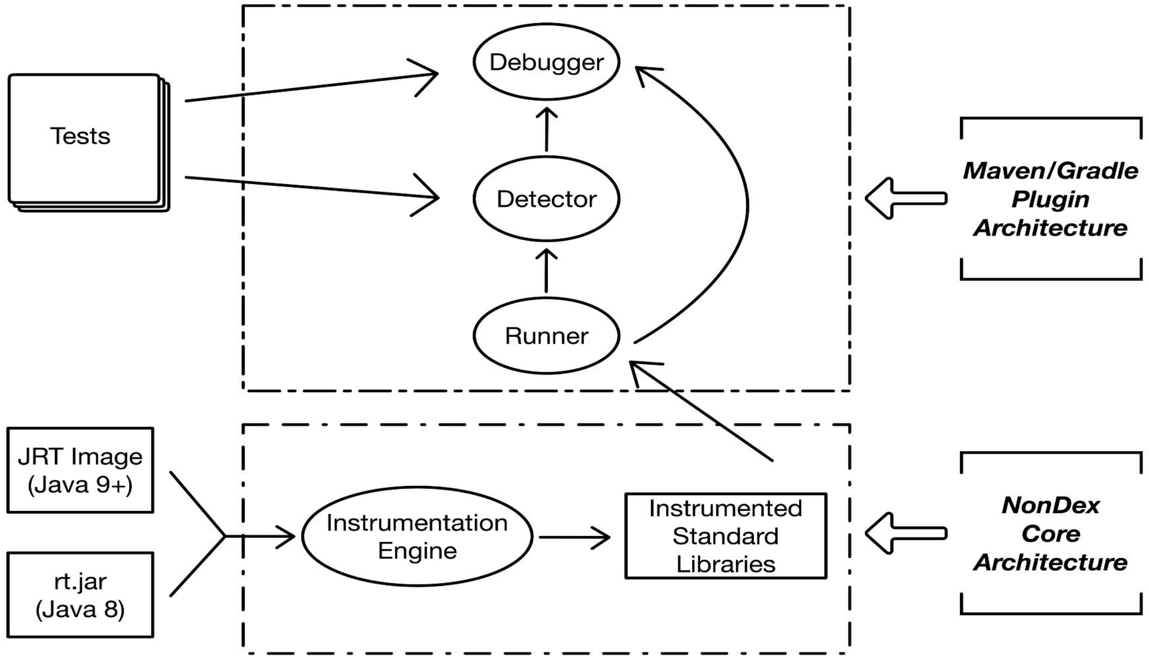
- **Configurable number of runs without shuffling**

  Avoid “debugging” non-ID tests

- **Support instrumenting new under-determined APIs**

  e.g., Spliterators

## Overview of the Latest Version of NonDex



## Large-Scale Experiments

- **Project Selection**

  207 Maven Projects (1500+ stars); 244 Gradle Projects (200+ Stars)

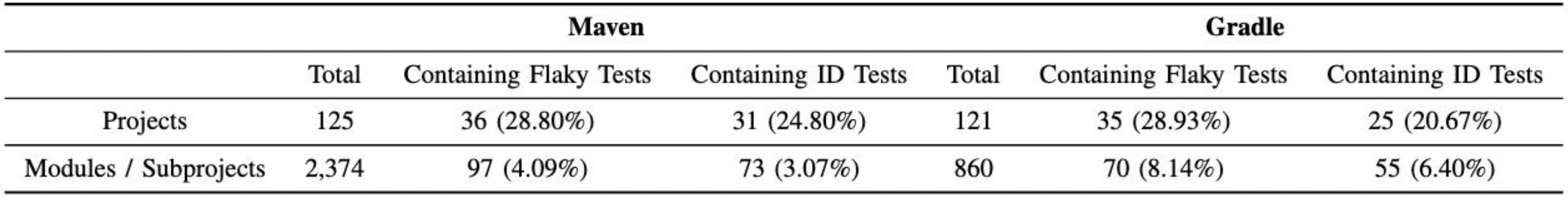
- **Choose Long-Term Support (LTS) Java versions**

  Java 17 and Java 21

- **Experiments ran on:**

  125 Maven Projects ; 121 Gradle Projects

## Results



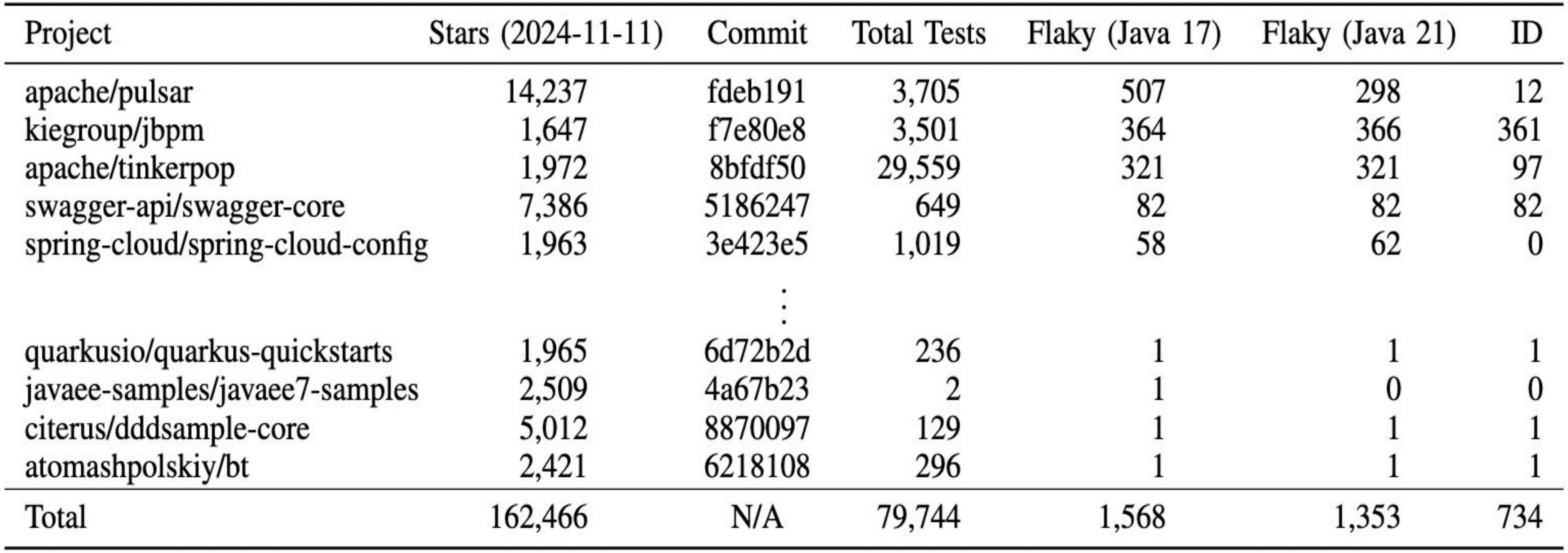
| | Maven | | | Gradle | | |
|---|---|---|---|---|---|---|
| | Total | Containing Flaky Tests | Containing ID Tests | Total | Containing Flaky Tests | Containing ID Tests |
| Projects | 125 | 36 (28.80%) | 31 (24.80%) | 121 | 35 (28.93%) | 25 (20.67%) |
| Modules / Subprojects | 2,374 | 97 (4.09%) | 73 (3.07%) | 860 | 70 (8.14%) | 55 (6.40%) |

Main Finding: **Number of ID Flaky tests “increase over time”**

Prior Work (2016) vs. Our Work:

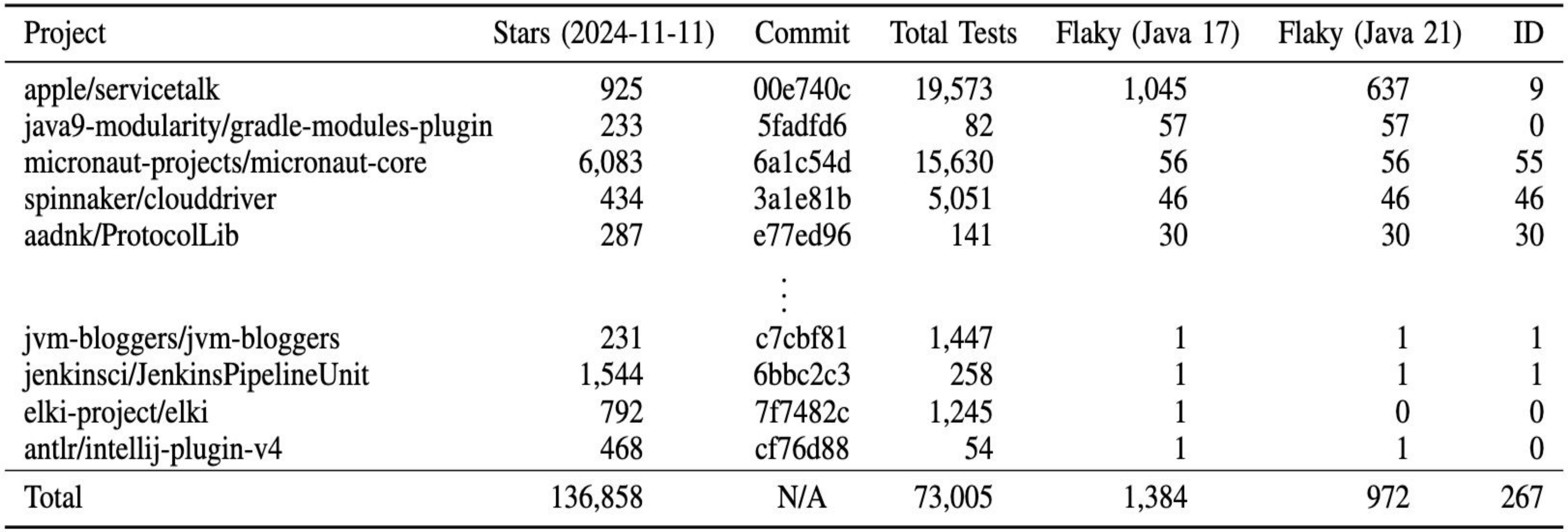
- Projects contain ID test under NonDex: **10.77%** vs. **22.76%**
- Maximum number of ID test in single project: **8** vs. **361**

## Results (Maven)



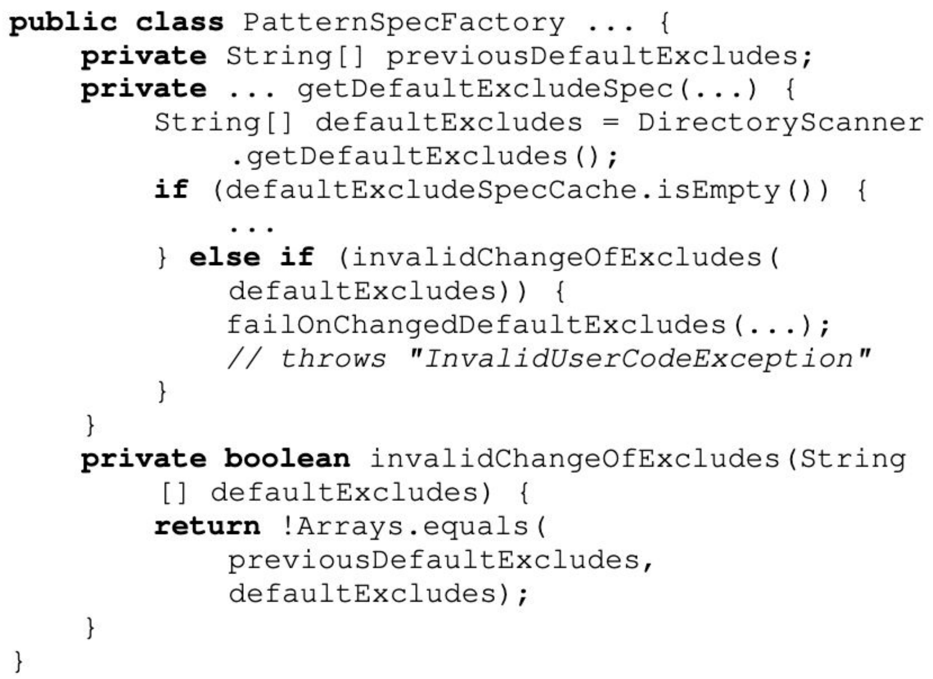
| Project | Stars (2024-11-11) | Commit | Total Tests | Flaky (Java 17) | Flaky (Java 21) | ID |
|---|---|---|---|---|---|---|
| apache/pulsar | 14,237 | fdeb191 | 3,705 | 507 | 298 | 12 |
| kiegroup/jbpm | 1,647 | f7e80e8 | 3,501 | 364 | 366 | 361 |
| apache/tinkerpop | 1,972 | 8bfdf50 | 29,559 | 321 | 321 | 97 |
| swagger-api/swagger-core | 7,386 | 5186247 | 649 | 82 | 82 | 82 |
| spring-cloud/spring-cloud-config | 1,963 | 3e423e5 | 1,019 | 58 | 62 | 0 |
| | | ⋮ | | | | |
| quarkusio/quarkus-quickstarts | 1,965 | 6d72b2d | 236 | 1 | 1 | 1 |
| javaee-samples/javaee7-samples | 2,509 | 4a67b23 | 2 | 1 | 0 | 0 |
| citerus/dddsample-core | 5,012 | 8870097 | 129 | 1 | 1 | 1 |
| atomashpolskiy/bt | 2,421 | 6218108 | 296 | 1 | 1 | 1 |
| Total | 162,466 | N/A | 79,744 | 1,568 | 1,353 | 734 |

## Results (Gradle)



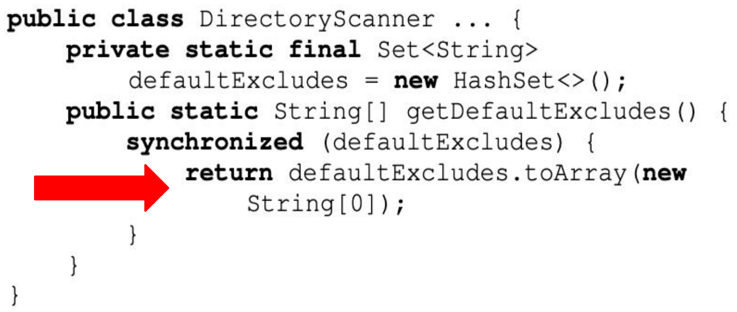
| Project | Stars (2024-11-11) | Commit | Total Tests | Flaky (Java 17) | Flaky (Java 21) | ID |
|---|---|---|---|---|---|---|
| apple/servicetalk | 925 | 00e740c | 19,573 | 1,045 | 637 | 9 |
| java9-modularity/gradle-modules-plugin | 233 | 5fadfd6 | 82 | 57 | 57 | 0 |
| micronaut-projects/micronaut-core | 6,083 | 6a1c54d | 15,630 | 56 | 56 | 55 |
| spinnaker/clouddriver | 434 | 3a1e81b | 5,051 | 46 | 46 | 46 |
| aadnk/ProtocolLib | 287 | e77ed96 | 141 | 30 | 30 | 30 |
| | | ⋮ | | | | |
| jvm-bloggers/jvm-bloggers | 231 | c7cbf81 | 1,447 | 1 | 1 | 1 |
| jenkinsci/JenkinsPipelineUnit | 1,544 | 6bbc2c3 | 258 | 1 | 1 | 1 |
| elki-project/elki | 792 | 7f7482c | 1,245 | 1 | 0 | 0 |
| antlr/intellij-plugin-v4 | 468 | cf76d88 | 54 | 1 | 1 | 0 |
| Total | 136,858 | N/A | 73,005 | 1,384 | 972 | 267 |

## Case Study: ID flakiness in Gradle Build Tool itself!



```
public class PatternSpecFactory ... {
    private String[] previousDefaultExcludes;
    private ... getDefaultExcludeSpec(...) {
        String[] defaultExcludes = DirectoryScanner
            .getDefaultExcludes();
        if (defaultExcludeSpecCache.isEmpty()) {
            ...
        } else if (invalidChangeOfExcludes(
            defaultExcludes)) {
            failOnChangedDefaultExcludes(...);
            // throws "InvalidUserCodeException"
        }
    }
    private boolean invalidChangeOfExcludes(String
        [] defaultExcludes) {
        return !Arrays.equals(
            previousDefaultExcludes,
            defaultExcludes);
    }
}
```

Fig. 2. Snippet of the PatternSpecFactory class from the Gradle core



```
public class DirectoryScanner ... {
    private static final Set<String>
        defaultExcludes = new HashSet<>();
    public static String[] getDefaultExcludes() {
        synchronized (defaultExcludes) {
            return defaultExcludes.toArray(new
                String[0]);
        }
    }
}
```

Fig. 3. Snippet of the DirectoryScanner class from the Gradle core

## Study of ID Test “Propagation”

- **Project Selection**

  69 “random” Gradle projects (without star threshold)

- **Detected 322 “Hard-to-Fix” ID Tests**

  Project has no way to control ID flakiness without **removing** the dependency on some external code

- **Our accepted PR to Gradle:**

  Resolves 109 of these, and potentially many more!

## Summary

- **NonDex**

  Works for multiple Java versions and build tools

- **ID Flaky Tests**

  More prevalent in modern Java projects

- **Propagation of Flaky Tests**

  Practicality to fix flaky tests