Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

JUnit5 inner test classes should be annotated with @Nested

Analyze your code

Bug Critical junit tests

If not annotated with `@Nested`, an inner class containing some tests will never be executed during tests execution. While you could still be able to manually run its tests in an IDE, it won't be the case during the build. By contrast, a static nested class containing some tests should not be annotated with `@Nested`, JUnit5 will not share setup and state with an instance of its enclosing class.

This rule raises an issue on inner classes and static nested classes containing JUnit5 test methods which has a wrong usage of `@Nested` annotation.

Note: This rule does not check if the context in which JUnit 5 is running (e.g. Maven Surefire Plugin) is properly configured to execute static nested classes, it could not be the case using the default configuration.

Noncompliant Code Example

```
import org.junit.jupiter.api.Test;

class MyJunit5Test {
  @Test
  void test() { /* ... */ }

  class InnerClassTest { // Noncompliant, missing @Nested an
    @Test
    void test() { /* ... */ }
  }

  @Nested
  static class StaticNestedClassTest { // Noncompliant, inva
    @Test
    void test() { /* ... */ }
  }
}
```

Compliant Solution

```
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.Nested;

class MyJunit5Test {
  @Test
  void test() { /* ... */ }

  @Nested
  class InnerClassTest {
    @Test
    void test() { /* ... */ }
  }

  static class StaticNestedClassTest {
    @Test
```

XML parsers should not be vulnerable to Denial of Service attacks

Vulnerability

XML parsers should not load external schemas

Vulnerability

Mobile database encryption keys should not be disclosed

Vulnerability

Reflection should not be vulnerable to injection attacks

Vulnerability

```
    void test() { /* ... */ }
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube