




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 **Java**


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



Java static code analysis

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

All rules 632

Vulnerability 53

Bug 154


Security Hotspot 36

Code Smell 389


Quick Fix 42


Tags ▾

Search by name... 🔍


 Bug


Assertions comparing incompatible types should not be made




 Bug


JUnit5 inner test classes should be annotated with @Nested




 Bug


Only one method invocation is expected when testing checked exceptions




 Bug


Assertion methods should not be used within the try block of a try-catch catching an Error




 Bug


Getters and setters should access the expected fields



 Bug

Zero should not be a possible denominator



 Bug

Locks should be released

 Bug

"runFinalizersOnExit" should not be called

 Bug

"ScheduledThreadPoolExecutor" should not have 0 core threads

 Bug


"Random" objects should be reused


 Bug


The signature of "finalize()" should match that of "Object.finalize()"

"switch" statements should not contain non-case labels

Analyze your code

 Code Smell

 Blocker

 suspicious

Even if it is legal, mixing case and non-case labels in the body of a switch statement is very confusing and can even be the result of a typing error.

Noncompliant Code Example

```
switch (day) {
    case MONDAY:
    case TUESDAY:
    WEDNESDAY: // Noncompliant; syntactically correct, but b
        doSomething();
        break;
    ...
}

switch (day) {
    case MONDAY:
        break;
    case TUESDAY:
        foo:for(int i = 0 ; i < X ; i++) { // Noncompliant; the
            /* ... */
            break foo; // this break statement doesn't relate t
            /* ... */
        }
        break;
    /* ... */
}
```

Compliant Solution








```
switch (day) {
    case MONDAY:
    case TUESDAY:
    case WEDNESDAY:
        doSomething();
        break;
    ...
}

switch (day) {
    case MONDAY:
        break;
    case TUESDAY:
        compute(args); // put the content of the labelled "for"
        break;

    /* ... */
}
```

https://rules.sonarsource.com/java/RSPEC-1219

1/2

<div><div>Jump statements should not occur in "finally" blocks</div><div> Bug</div></div>	<div>Available In:</div> <div>  </div>
<div><div>"super.finalize()" should be called at the end of "Object.finalize()" implementations</div><div> Bug</div></div>	
<div><div>Using slow regular expressions is security-sensitive</div><div> Security Hotspot</div></div>	
<div><div>Using publicly writable directories is security-sensitive</div><div> Security Hotspot</div></div>	

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)