 **sonar** RULES

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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags ▾

Search by name... 🔍

@Atomic values should not be set to null

Bug

"Iterator.next()" methods should throw "NoSuchElementException"

Bug

"compareTo" results should not be checked for specific values

Bug

Math operands should be cast before assignment

Bug

Ints and longs should not be shifted by zero or more than their number of bits-1

Bug

"compareTo" should not return "Integer.MIN\_VALUE"

Bug

Boxing and unboxing should not be immediately reversed

Bug

"equals(Object obj)" should test argument type

Bug

"Serializable" inner classes of non-serializable classes should be "static"

Bug

The non-serializable super class of a "Serializable" class should have a no-argument constructor

Bug

Method parameters, caught exceptions and foreach variables' initial values should not be ignored

Bug

## ".equals()" should not be used to test the values of "Atomic" classes

Analyze your code

Bug

Major

multi-threading

AtomicInteger, and AtomicLong extend Number, but they're distinct from Integer and Long and should be handled differently. AtomicInteger and AtomicLong are designed to support lock-free, thread-safe programming on single variables. As such, an AtomicInteger will only ever be "equal" to itself. Instead, you should .get() the value and make comparisons on it.

This applies to all the atomic, seeming-primitive wrapper classes: AtomicInteger, AtomicLong, and AtomicBoolean.

### Noncompliant Code Example

```
AtomicInteger aInt1 = new AtomicInteger(0);
AtomicInteger aInt2 = new AtomicInteger(0);

if (aInt1.equals(aInt2)) { ... } // Noncompliant
```

### Compliant Solution

```
AtomicInteger aInt1 = new AtomicInteger(0);
AtomicInteger aInt2 = new AtomicInteger(0);

if (aInt1.get() == aInt2.get()) { ... }
```

Available In:

sonarlint

sonarcloud




sonarqube

© 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](#)

https://rules.sonarsource.com/java/RSPEC-2204

1/2

<p>"equals(Object obj)" and "hashCode()" should be overridden in pairs</p> <p> Bug</p>
<p>Disclosing fingerprints from web application technologies is security-sensitive</p> <p> Security Hotspot</p>
<p>Having a permissive Cross-Origin Resource Sharing policy is security-sensitive</p> <p> Security Hotspot</p>
<p>Delivering code in production with debug features activated is security-sensitive</p>