




 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... 🔍


annotated with @injected

 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()"

 Bug

Jump statements should not occur in "finally" blocks

 Bug

"super.finalize()" should be called at the end of "Object.finalize()" implementations

 Bug

Future keywords should not be used as names

Analyze your code

Code Smell

Blocker

obsolete pitfall

Through Java's evolution keywords have been added. While code that uses those words as identifiers may be compilable under older versions of Java, it will not be under modern versions.

Following keywords are marked as invalid identifiers

Keyword	Added
—	9
enum	5.0

assert and strictfp are another example of valid identifiers which became keywords in later versions, but are not supported by this rule.

Noncompliant Code Example

```
public void doSomething() {
    int enum = 42;           // Noncompliant
    String _ = "";          // Noncompliant
}
```

Compliant Solution

```
public void doSomething() {
    int magic = 42;
}
```

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-1190

1/2

 Bug
<div>Using slow regular expressions is security-sensitive</div> <div> Security Hotspot</div>
<div>Using publicly writable directories is security-sensitive</div> <div> Security Hotspot</div>
<div>Using clear-text protocols is security-sensitive</div> <div> Security Hotspot</div>
<div>Accessing Android external storage is security-sensitive</div>