sonar RULES

Products

- 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...

Classes with only "static" methods should not be instantiated

Code Smell

"Threads" should not be used where "Runnables" are expected

Code Smell

Inner class calls to super class methods should be unambiguous

Code Smell

Unused type parameters should be removed

Code Smell

Parameters should be passed in the correct order

Code Smell

"ResultSet.isLast()" should not be used

Code Smell

"static" members should be accessed statically

Code Smell

Silly math should not be performed

Code Smell

Classes named like "Exception" should extend "Exception" or a subclass

Code Smell

Exceptions should be either logged or rethrown but not both

Code Smell

Objects should not be created only to "getClass"

Code Smell

Primitives should not be boxed just for "String" conversion

Map values should not be replaced unconditionally

Analyze your code

Bug Major suspicious

It is highly suspicious when a value is saved for a key or index and then unconditionally overwritten. Such replacements are likely errors.

Noncompliant Code Example

```
letters.put("a", "Apple");
letters.put("a", "Boy");  // Noncompliant

towns[i] = "London";
towns[i] = "Chicago";  // Noncompliant
```

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

"String" conversion

Code Smell

Constructors should not be used to instantiate "String", "BigInteger", "BigDecimal" and primitive-wrapper classes

Code Smell

"URL.hashCode" and "URL.equals" should be avoided

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell