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...

- Lines should not be too long — Code Smell
- JEE applications should not "getClassLoader" — Bug
- Math should not be performed on floats — Bug
- "equals" methods should be symmetric and work for subclasses — Bug
- Literal suffixes should be upper case — Code Smell
- Unicode-aware versions of character classes should be preferred — Code Smell
- Use Java 12 "switch" expression — Code Smell
- "serialVersionUID" should not be declared blindly — Code Smell
- "Stream.collect()" calls should not be redundant — Code Smell
- Local constants should follow naming conventions for constants — Code Smell
- Unit tests should throw exceptions — Code Smell
- Test methods should comply with a naming convention — Code Smell

Primitives should not be boxed just for "String" conversion

Analyze your code

Code Smell Major performance

"Boxing" is the process of putting a primitive value into a primitive-wrapper object. When that's done purely to use the wrapper class' `toString` method, it's a waste of memory and cycles because those methods are `static`, and can therefore be used without a class instance. Similarly, using the `static` method `valueOf` in the primitive-wrapper classes with a non-`String` argument should be avoided.

Noncompliant Code Example

```
int myInt = 4;
String myIntString = (new Integer(myInt)).toString(); // Non
myIntString = Integer.valueOf(myInt).toString(); // Noncompl
```

Compliant Solution

```
int myInt = 4;
String myIntString = Integer.toString(myInt);
```

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

Value-based objects should not be serialized

Code Smell

Default annotation parameter values should not be passed as arguments

Code Smell

Method parameters should be declared with base types

Code Smell

Fields should not be initialized to default values

Code Smell