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...

Loggers should be named for their enclosing classes
Code Smell

Methods should not return constants
Code Smell

"private" methods called only by inner classes should be moved to those classes
Code Smell

"enum" fields should not be publicly mutable
Code Smell

Abstract methods should not be redundant
Code Smell

Arrays should not be copied using loops
Code Smell

Static non-final field names should comply with a naming convention
Code Smell

JUnit rules should be used
Code Smell

Nested "enum"s should not be declared static
Code Smell

"catch" clauses should do more than rethrow
Code Smell

Mutable fields should not be "public static"
Code Smell

Methods should not be named "tostring", "hashcode" or "equal"

Analyze your code

Bug Major pitfall

Naming a method `tostring`, `hashcode` or `equal` is either:

- A bug in the form of a typo. Overriding `toString`, `Object.hashCode()` (note the camelCasing) or `Object.equals` (note the 's' on the end) was meant, and the application does not behave as expected.
- Done on purpose. The name however will confuse every other developer, who may not notice the naming difference, or who will think it is a bug.

In both cases, the method should be renamed.

Noncompliant Code Example

```
public int hashcode() { /* ... */ }  // Noncompliant

public String tostring() { /* ... */ } // Noncompliant

public boolean equal(Object obj) { /* ... */ }  // Noncompli
```

Compliant Solution

```
@Override
public int hashCode() { /* ... */ }

@Override
public String toString() { /* ... */ }

@Override
public boolean equals(Object obj) { /* ... */ }
```

Available In:

sonarlint sonarcloud sonarqube

The diamond operator ("<>") should be used

Code Smell

"finalize" should not set fields to "null"

Code Smell

Subclasses that add fields should override "equals"

Code Smell

Catches should be combined

Code Smell

Methods of "Random" that return floating point values should not be