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...

Bug

"BigDecimal(double)" should not be used
Bug

Invalid "Date" values should not be used
Bug

Reflection should not be used to check non-runtime annotations
Bug

Custom serialization method signatures should meet requirements
Bug

"Externalizable" classes should have no-arguments constructors
Bug

Classes should not be compared by name
Bug

Related "if/else if" statements should not have the same condition
Bug

Synchronization should not be done on instances of value-based classes
Bug

"Iterator.hasNext()" should not call "Iterator.next()"
Bug

Identical expressions should not be used on both sides of a binary operator
Bug

Loops with at most one iteration should be refactored
Bug

Class names should not shadow interfaces or superclasses

Analyze your code

Code Smell | Critical | pitfall

While it's perfectly legal to give a class the same simple name as a class in another package that it extends or interface it implements, it's confusing and could cause problems in the future.

Noncompliant Code Example

```
package my.mypackage;

public class Foo implements a.b.Foo { // Noncompliant
```

Compliant Solution

```
package my.mypackage;

public class FooJr implements a.b.Foo {
```

Available In:
sonarlint | sonarcloud | sonarqube

Bug

Variables should not be self-assigned

Bug

"StringBuilder" and "StringBuffer" should not be instantiated with a character

Bug

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

Bug

"Thread.run()" should not be called directly