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...

Abstract class names should comply with a naming convention
Code Smell

Strings literals should be placed on the left side when checking for equality
Code Smell

Files should contain an empty newline at the end
Code Smell

Source code should be indented consistently
Code Smell

A close curly brace should be located at the beginning of a line
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line
Code Smell

An open curly brace should be located at the beginning of a line
Code Smell

An open curly brace should be located at the end of a line
Code Smell

Tabulation characters should not be used
Code Smell

Functions should not be defined with a variable number of arguments
Code Smell

String operations should not rely on the default system locale

Analyze your code

Code Smell Minor unpredictable cert

Failure to specify a locale when calling the methods `toLowerCase()`, `toUpperCase()` or `format()` on `String` objects means the system default encoding will be used, possibly creating problems with international characters or number representations. For instance with the Turkish language, when converting the small letter 'i' to upper case, the result is capital letter 'I' with a dot over it.

Case conversion without a locale may work fine in its "home" environment, but break in ways that are extremely difficult to diagnose for customers who use different encodings. Such bugs can be nearly, if not completely, impossible to reproduce when it's time to fix them. For locale-sensitive strings, the correct locale should always be used, but `Locale.ROOT` can be used for case-insensitive ones.

Noncompliant Code Example

```
myString.toLowerCase()
```

Compliant Solution

```
myString.toLowerCase(Locale.TR)
```

See

- CERT, STR02-J. - Specify an appropriate locale when comparing locale-dependent data

Available In:
sonarlint sonarcloud sonarqube

Local-Variable Type Inference should be used

Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

Code Smell

Track uses of disallowed classes

Code Smell

Track uses of "@SuppressWarnings" annotations

Code Smell