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...

"switch" statements should have at least 3 "case" clauses
Code Smell

A "while" loop should be used instead of a "for" loop
Code Smell

The default unnamed package should not be used
Code Smell

"equals(Object obj)" should be overridden along with the "compareTo(T obj)" method
Code Smell

Package names should comply with a naming convention
Code Smell

Nested code blocks should not be used
Code Smell

Array designators "[]" should be on the type, not the variable
Code Smell

Array designators "[]" should be located after the type in method signatures
Code Smell

Type parameter names should comply with a naming convention
Code Smell

Overriding methods should do more than simply call the same method in the super class
Code Smell

Classes that override "clone" should be "Cloneable" and call "super.clone()"
Code Smell

"Stream.toList()" method should be used instead of "collectors" when unmodifiable list needed

Analyze your code

Code Smell | Major | java16

In Java 8 `Streams` were introduced to support chaining of operations over collections in a functional style. The most common way to save a result of such chains is to save them to some collection (usually `List`). To do so there is a terminal method `collect` that can be used with a library of `Collectors`. The key problem is that `.collect(Collectors.toList())` actually returns a mutable kind of `List` while in the majority of cases unmodifiable lists are preferred. In Java 10 a new collector appeared to return an unmodifiable list: `toUnmodifiableList()`. This does the trick but results in verbose code. Since Java 16 there is now a better variant to produce an unmodifiable list directly from a stream: `Stream.toList()`.

This rule raises an issue when "collect" is used to create a list from a stream.

Noncompliant Code Example

```
List<String> list1 = Stream.of("A", "B", "C")
                           .collect(Collectors.toList()); //

List<String> list2 = Stream.of("A", "B", "C")
                           .collect(Collectors.toUnmodifiabl
```

Compliant Solution

```
List<String> list1 = Stream.of("A", "B", "C").toList(); // C

List<String> list2 = Stream.of("A", "B", "C")
                           .collect(Collectors.toList()); //

list2.add("X");
```

Available In:
sonarlint sonarcloud sonarqube

Public constants and fields initialized at declaration should be "static final" rather than merely "final"

Code Smell

Local variable and method parameter names should comply with a naming convention

Code Smell

Exception classes should be immutable

Code Smell

Field names should comply with a naming convention