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...

"Preconditions" and logging arguments should not require evaluation
Code Smell

Boolean expressions should not be gratuitous
Code Smell

"Lock" objects should not be "synchronized"
Code Smell

Classes with only "static" methods should not be instantiated
Code Smell

"Threads" should not be used where "Runnables" are expected
Code Smell

Inner class calls to super class methods should be unambiguous
Code Smell

Unused type parameters should be removed
Code Smell

Parameters should be passed in the correct order
Code Smell

"ResultSet.isLast()" should not be used
Code Smell

"static" members should be accessed statically
Code Smell

Silly math should not be performed
Code Smell

Classes named like "Exception" should extend "Exception" or a subclass

"compareTo" should not be overloaded

Analyze your code

Bug | Major | pitfall

When implementing the `Comparable<T>.compareTo` method, the parameter's type has to match the type used in the `Comparable` declaration. When a different type is used this creates an overload instead of an override, which is unlikely to be the intent.

This rule raises an issue when the parameter of the `compareTo` method of a class implementing `Comparable<T>` is not same as the one used in the `Comparable` declaration.

Noncompliant Code Example

```
public class Foo {
  static class Bar implements Comparable<Bar> {
    public int compareTo(Bar rhs) {
      return -1;
    }
  }

  static class FooBar extends Bar {
    public int compareTo(FooBar rhs) {  // Noncompliant: Par
      return 0;
    }
  }
}
```

Compliant Solution

```
public class Foo {
  static class Bar implements Comparable<Bar> {
    public int compareTo(Bar rhs) {
      return -1;
    }
  }

  static class FooBar extends Bar {
    public int compareTo(Bar rhs) {
      return 0;
    }
  }
}
```

Available In:
sonarlint | sonarcloud | sonarqube

extend "Exception" or a subclass

Code Smell

Exceptions should be either logged or rethrown but not both

Code Smell

Objects should not be created only to "getClass"

Code Smell

Primitives should not be boxed just for "String" conversion

Code Smell

Constructors should not be used to instantiate "String", "BigInteger",