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

Math should not be performed on floats
Bug

"equals" methods should be symmetric and work for subclasses
Bug

Literal suffixes should be upper case
Code Smell

Unicode-aware versions of character classes should be preferred
Code Smell

Use Java 12 "switch" expression
Code Smell

"serialVersionUID" should not be declared blindly
Code Smell

"Stream.collect()" calls should not be redundant
Code Smell

Local constants should follow naming conventions for constants
Code Smell

Unit tests should throw exceptions
Code Smell

Test methods should comply with a naming convention
Code Smell

Value-based objects should not be serialized
Code Smell

Default annotation parameter values should not be passed as arguments

Constructors should not be used to instantiate "String", "BigInteger", "BigDecimal" and primitive-wrapper classes

Analyze your code

Code Smell | Major | performance

Constructors for `String`, `BigInteger`, `BigDecimal` and the objects used to wrap primitives should never be used. Doing so is less clear and uses more memory than simply using the desired value in the case of strings, and using `valueOf` for everything else.

Noncompliant Code Example

```
String empty = new String(); // Noncompliant; yields essenti
String nonempty = new String("Hello world"); // Noncompliant
Double myDouble = new Double(1.1); // Noncompliant; use valu
Integer integer = new Integer(1); // Noncompliant
Boolean bool = new Boolean(true); // Noncompliant
BigInteger bigInteger1 = new BigInteger("3"); // Noncomplian
BigInteger bigInteger2 = new BigInteger("9223372036854775807
BigInteger bigInteger3 = new BigInteger("111222333444555666
```

Compliant Solution

```
String empty = "";
String nonempty = "Hello world";
Double myDouble = Double.valueOf(1.1);
Integer integer = Integer.valueOf(1);
Boolean bool = Boolean.valueOf(true);
BigInteger bigInteger1 = BigInteger.valueOf(3);
BigInteger bigInteger2 = BigInteger.valueOf(9223372036854775
BigInteger bigInteger3 = new BigInteger("111222333444555666
```

Exceptions

`BigDecimal` constructor with `double` argument is ignored as using `valueOf` instead might change resulting value. See {rule:java:S2111} .

Available In:

sonarlint | sonarcloud | sonarqube

Code Smell

Method parameters should be declared with base types

Code Smell

Fields should not be initialized to default values

Code Smell

Multiple loops over the same set should be combined

Code Smell

Classes without "public" constructors should be "final"