




 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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags ▾

Search by name... 🔍

"Optional" should not be used for parameters

Code Smell

Tests should be kept in a dedicated source directory

Code Smell

"this" should not be exposed from constructors

Code Smell

Classes should not have too many "static" imports

Code Smell

Escaped Unicode characters should not be used

Code Smell

Inner classes should not have too many lines of code

Code Smell

Inner classes which do not reference their owning classes should be "static"

Code Smell

"deleteOnExit" should not be used

Code Smell

Public methods should not contain selector arguments

Code Smell

Java parser failure

Code Smell

Track uses of disallowed methods

Code Smell

Types should be used in lambdas

Code Smell

Static fields should not be updated in constructors

Analyze your code

Code Smell

Major ?

Assigning a value to a `static` field in a constructor could cause unreliable behavior at runtime since it will change the value for all instances of the class.

Instead remove the field's `static` modifier, or initialize it statically.

Noncompliant Code Example

```
public class Person {
    static Date dateOfBirth;
    static int expectedFingers;

    public Person(date birthday) {
        dateOfBirth = birthday; // Noncompliant; now everyone h
        expectedFingers = 10; // Noncompliant
    }
}
```

Compliant Solution

```
public class Person {
    Date dateOfBirth;
    static int expectedFingers = 10;

    public Person(date birthday) {
        dateOfBirth = birthday;
    }
}
```

Available In:

sonarlint

sonarcloud





sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy

https://rules.sonarsource.com/java/RSPEC-3010

1/2

<p><b>"java.time" classes should be used for dates and times</b></p> <p> Code Smell</p>
<p><b>The names of methods with boolean return values should start with "is" or "has"</b></p> <p> Code Smell</p>
<p><b>Files should contain only one top-level class or interface each</b></p> <p> Code Smell</p>
<p><b>Classes should not have too many fields</b></p> <p></p>