




 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... 🔍

 Code Smell


Chained AssertJ assertions should be simplified to the corresponding dedicated assertion




Exception testing via JUnit @Test annotation should be avoided




Escape sequences should not be used in text blocks




Simple string literal should be used for single line strings




Boxed "Boolean" should be avoided in boolean expressions




Type parameters should not shadow other type parameters




"read(byte[],int,int)" should be overridden




An iteration on a Collection should be performed on the type handled by the Collection



"StandardCharsets" constants should be preferred





"@CheckForNull" or "@Nullable" should not be used on primitive types



Composed "@RequestMapping" variants should be preferred

Invalid "Date" values should not be used

Analyze your code

 Bug  Major ?

Whether the valid value ranges for Date fields start with 0 or 1 varies by field. For instance, month starts at 0, and day of month starts at 1. Enter a date value that goes past the end of the valid range, and the date will roll without error or exception. For instance, enter 12 for month, and you'll get January of the following year.

This rule checks for bad values used in conjunction with java.util.Date, java.sql.Date, and java.util.Calendar. Specifically, values outside of the valid ranges:

Field	Valid
month	0-11
date (day)	0-31
hour	0-23
minute	0-60
second	0-61

Note that this rule does not check for invalid leap years, leap seconds (second = 61), or invalid uses of the 31st day of the month.

Noncompliant Code Example




```
Date d = new Date();
d.setDate(25);
d.setYear(2014);
d.setMonth(12); // Noncompliant; rolls d into the next year

Calendar c = new GregorianCalendar(2014, 12, 25); // Noncom
if (c.get(Calendar.MONTH) == 12) { // Noncompliant; invalid
    // ...
}
```

Compliant Solution





```
Date d = new Date();
d.setDate(25);
d.setYear(2014);
d.setMonth(11);

Calendar c = new GregorianCalendar(2014, 11, 25);
if (c.get(Calendar.MONTH) == 11) {
    // ...
}
```

Available In:
  

https://rules.sonarsource.com/java/RSPEC-2110

1/2

 Code Smell	<div>© 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</div>
"write(byte[],int,int)" should be overridden	
 Code Smell	
Functional Interfaces should be as specialised as possible	
 Code Smell	
Null checks should not be used with "instanceof"	
 Code Smell	
"close()" calls should not be redundant	