




 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... 🔍


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


"java.time" classes should be used for dates and times

 Code Smell


The names of methods with boolean return values should start with "is" or "has"

 Code Smell

Files should contain only one top-level class or interface each


 Code Smell


Classes should not have too many fields


 Code Smell

"DateUtils.truncate" from Apache Commons Lang library should not be used

Analyze your code

 Code Smell

 Major ?

 performance

java8

The use of the `ZonedDateTime` class introduced in Java 8 to truncate a date can be significantly faster than the `DateUtils` class from Commons Lang.

Note that this rule is automatically disabled when the project's `sonar.java.source` is lower than 8.




Noncompliant Code Example

```
public Date trunc(Date date) {
    return DateUtils.truncate(date, Calendar.SECOND); // Nonc
}
```

Compliant Solution

```
public Date trunc(Date date) {
    Instant instant = date.toInstant();
    ZonedDateTime zonedDateTime = instant.atZone(ZoneId.system
    ZonedDateTime truncatedZonedDateTime = zonedDateTime.trunc
    Instant truncatedInstant = truncatedZonedDateTime.toInstant
    return Date.from(truncatedInstant);
}
```





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-2718

1/2

<div>The ternary operator should not be used</div> <div> Code Smell</div>
<div>Standard functional interfaces should not be redefined</div> <div> Code Smell</div>
<div>"NullPointerException" should not be caught</div> <div> Code Smell</div>
<div>"NullPointerException" should not be explicitly thrown</div> <div> Code Smell</div>