


-  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... 🔍

Exception types should not be tested using "instanceof" in catch blocks

Code Smell

Classes from "sun.*" packages should not be used

Code Smell

Throwable and Error should not be caught

Code Smell

Unused method parameters should be removed

Code Smell

Only static class initializers should be used

Code Smell

Empty arrays and collections should be returned instead of null

Code Smell

"@Override" should be used on overriding and implementing methods

Code Smell

Enumeration should not be implemented

Code Smell

Synchronized classes Vector, Hashtable, Stack and StringBuffer should not be used

Code Smell

Unused "private" methods should be removed

Code Smell

Try-catch blocks should not be nested

Code Smell

Track uses of "FIXME" tags

Min and max used in combination should not always return the same value

Analyze your code

Bug

Major

When using `Math.min()` and `Math.max()` together for bounds checking, it's important to feed the right operands to each method. `Math.min()` should be used with the **upper** end of the range being checked, and `Math.max()` should be used with the **lower** end of the range. Get it backwards, and the result will always be the same end of the range.

Noncompliant Code Example

```
private static final int UPPER = 20;
private static final int LOWER = 0;

public int doRangeCheck(int num) {    // Let's say num = 1
    int result = Math.min(LOWER, num); // result = 0
    return Math.max(UPPER, result);   // Noncompliant; res
}
```

Compliant Solution

Swapping method `min()` and `max()` invocations without changing parameters.

```
private static final int UPPER = 20;
private static final int LOWER = 0;

public int doRangeCheck(int num) {    // Let's say num = 1
    int result = Math.max(LOWER, num); // result = 12
    return Math.min(UPPER, result);   // Compliant; result
}
```

or swapping bounds `UPPER` and `LOWER` used as parameters without changing the invoked methods.

```
private static final int UPPER = 20;
private static final int LOWER = 0;

public int doRangeCheck(int num) {    // Let's say num = 1
    int result = Math.min(UPPER, num); // result = 12
    return Math.max(LOWER, result);   // Compliant; result
}
```

Available In:

sonarlint





 |

sonarcloud

 |

sonarqube

Java static code analysis: Min and max used in combination should not always return the same value

 Code Smell
Deprecated elements should have both the annotation and the Javadoc tag
 Code Smell
Assignments should not be made from within sub-expressions
 Code Smell
Generic exceptions should never be thrown
 Code Smell
Labels should not be used

© 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](#)