
































-  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... 🔍

Specialised as possible

 Code Smell

Null checks should not be used with "instanceof"

 Code Smell

"close()" calls should not be redundant

 Code Smell

"ThreadLocal.withInitial" should be preferred

 Code Smell

"Stream" call chains should be simplified when possible

 Code Smell

Packages containing only "package-info.java" should be removed

 Code Smell

Arrays should not be created for varargs parameters

 Code Smell

Jump statements should not be redundant

 Code Smell

Test classes should comply with a naming convention

 Code Smell

Loggers should be named for their enclosing classes

 Code Smell

Methods should not return constants

 Code Smell

"private" methods called only by inner classes should be moved to those classes

 Code Smell

"Iterator.hasNext()" should not call "Iterator.next()"

Analyze your code

 Bug  Major ?





Calling `Iterator.hasNext()` is not supposed to have any side effects, and therefore should not change the state of the iterator. `Iterator.next()` advances the iterator by one item. So calling it inside `Iterator.hasNext()`, breaks the `hasNext()` contract, and will lead to unexpected behavior in production.

Noncompliant Code Example

```
public class FibonacciIterator implements Iterator<Integer>{
    ...
    @Override
    public boolean hasNext() {
        if(next() != null) {
            return true;
        }
        return false;
    }
    ...
}
```

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](#)

<p><b>"enum" fields should not be publicly mutable</b></p> <p> Code Smell</p>
<p><b>Abstract methods should not be redundant</b></p> <p> Code Smell</p>
<p><b>Arrays should not be copied using loops</b></p> <p> Code Smell</p>
<p><b>Static non-final field names should comply with a naming convention</b></p> <p> Code Smell</p>