




 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... 🔍

Instance methods should not write to "static" fields

Code Smell

"indexOf" checks should not be for positive numbers

Code Smell

Method overrides should not change contracts

Code Smell

Whitespace and control characters in literals should be explicit

Code Smell

Null should not be returned from a "Boolean" method

Code Smell

Classes should not access their own subclasses during initialization

Code Smell

"Object.wait(...)" and "Condition.await(...)" should be called inside a "while" loop

Code Smell

IllegalMonitorStateException should not be caught

Code Smell

JUnit assertions should not be used in "run" methods

Code Smell

Class names should not shadow interfaces or superclasses

Code Smell

"Cloneables" should implement "clone"

Code Smell

Try-with-resources should be used

Code Smell

Cast operations should not trigger a ClassCastException

Analyze your code

Bug

Critical

A cast operation makes it possible to convert an object from one type to another one. The compiler raises an error when it can determine that the target type is incompatible with the declared type of the object. Other cases are accepted by the compiler. However, depending on the actual runtime type of the object, a cast operation may fail at runtime with a `ClassCastException` which can crash the program. This kind of failure happens when a piece of code makes an invalid assumption about the runtime type of an object.

The appropriate fix for those cast operations highly depends on the context: adding a condition before the cast or changing the runtime type of the object are some of the possible solutions.

Noncompliant Code Example

```
List<String> list = new LinkedList<>();
if (someCondition) {
    list = new ArrayList<>();
}
((LinkedList<String>) list).addLast("abc"); // Noncompliant,
```

Compliant Solution


```
List<String> list = new LinkedList<>();
if (someCondition) {
    list = new ArrayList<>();
}
if (list instanceof LinkedList) {
    ((LinkedList<String>) list).addLast("abc");
}

or

List<String> list = new LinkedList<>();
if (someCondition) {
    list = new ArrayList<>();
}
list.add("abc");






or

LinkedList<String> list = new LinkedList<>();
list.addLast("abc");
```

Available In:
sonarcloud 

https://rules.sonarsource.com/java/RSPEC-6320

1/2

 Code Smell
"readResolve" methods should be inheritable  Code Smell
"for" loop increment clauses should modify the loops' counters  Code Smell
Fields in a "Serializable" class should either be transient or serializable  Code Smell
Package declaration should match source file directory  Code Smell

© 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](#)