




 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... 🔍

 Vulnerability


Calls to methods should not trigger an `IllegalArgumentException`

 Bug


Unsupported methods should not be called on some collection implementations

 Bug


Cast operations should not trigger a `ClassCastException`

 Bug


Members ignored during record serialization should not be used

 Bug


Map "`computeIfAbsent()`" and "`computeIfPresent()`" should not be used to add "null" values.

 Bug


Regex lookahead assertions should not be contradictory

 Bug


Back references in regular expressions should only refer to capturing groups that are matched before the reference

 Bug


Regex boundaries should not be used in a way that can never be matched

 Bug

Regex patterns following a possessive quantifier should not always fail


 Bug


Regular expressions should be syntactically valid


 Bug

JUnit test cases should call super methods

Analyze your code

 Code Smell

 Blocker

 junit tests

Overriding a parent class method prevents that method from being called unless an explicit `super` call is made in the overriding method. In some cases not calling the `super` method is acceptable, but not with `setUp` and `tearDown` in a JUnit 3 `TestCase`.

Noncompliant Code Example

```
public class MyClassTest extends MyAbstractTestCase {

    private MyClass myClass;

    @Override
    protected void setUp() throws Exception { // Noncompliant
        myClass = new MyClass();
    }
}
```




Compliant Solution

```
public class MyClassTest extends MyAbstractTestCase {

    private MyClass myClass;

    @Override
    protected void setUp() throws Exception {
        super.setUp();
        myClass = new MyClass();
    }
}
```





Available In:

 |  | 

© 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-2188

1/2

<p>Assertions comparing incompatible types should not be made</p> <p> Bug</p>
<p>JUnit5 inner test classes should be annotated with @Nested</p> <p> Bug</p>
<p>Only one method invocation is expected when testing checked exceptions</p> <p> Bug</p>
<p>Assertion methods should not be used within the try block of a try-catch catching an Error</p> <p> Bug</p>