




 **sonar** RULES


 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

overridable methods
 Code Smell
Methods should not be too complex
 Code Smell
Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply
 Code Smell
"if ... else if" constructs should end with "else" clauses
 Code Smell
Control structures should use curly braces
 Code Smell
Expressions should not be too complex
 Code Smell
Mockito argument matchers should be used on all parameters
 Bug
Spring "@Controller" classes should not use "@Scope"
 Bug
Constructor injection should be used instead of field injection
 Bug
Classes that don't define "hashCode()" should not be used in hashes
 Bug
Floating point numbers should not be tested for equality
 Bug
Increment (++) and decrement (--) operators should not be used in a method call or mixed with other

"java.nio.Files#delete" should be preferred

Analyze your code

 Code Smell  Major  error-handling api-design

When `java.io.File#delete` fails, this boolean method simply returns `false` with no indication of the cause. On the other hand, when `java.nio.file.Files#delete` fails, this void method returns one of a series of exception types to better indicate the cause of the failure. And since more information is generally better in a debugging situation, `java.nio.file.Files#delete` is the preferred option.

Noncompliant Code Example





```
public void cleanUp(Path path) {
    File file = new File(path);
    if (!file.delete()) { // Noncompliant
        //...
    }
}
```

Compliant Solution

```
public void cleanUp(Path path) throws NoSuchFileException, D
    Files.delete(path);
}
```

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](#)

operators in an expression  Code Smell
Limited dependence should be placed on operator precedence  Code Smell
Custom getter method should not be used to override record's getter behavior  Code Smell
Tests should use fixed data instead of randomized data  Code Smell