




 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... 

Java 8's "Files.exists" should not be used
 Code Smell
"Optional" should not be used for parameters
 Code Smell
Tests should be kept in a dedicated source directory
 Code Smell
"this" should not be exposed from constructors
 Code Smell
Classes should not have too many "static" imports
 Code Smell
Escaped Unicode characters should not be used
 Code Smell
Inner classes should not have too many lines of code
 Code Smell
Inner classes which do not reference their owning classes should be "static"
 Code Smell
"deleteOnExit" should not be used
 Code Smell
Public methods should not contain selector arguments
 Code Smell
Java parser failure
 Code Smell
Track uses of disallowed methods
 Code Smell

Reflection should not be used to increase accessibility of classes, methods, or fields

Analyze your code

 Code Smell

 Major 

 cert

This rule raises an issue when reflection is used to change the visibility of a class, method or field, and when it is used to directly update a field value. Altering or bypassing the accessibility of classes, methods, or fields violates the encapsulation principle and could lead to run-time errors.





Noncompliant Code Example

```
public void makeItPublic(String methodName) throws NoSuchMet  
  
    this.getClass().getMethod(methodName).setAccessible(true);  
}  
  
public void setItAnyWay(String fieldName, int value) {  
    this.getClass().getDeclaredField(fieldName).setInt(this, v  
}
```

- See
- [CERT, SEC05-J](#). - Do not use reflection to increase accessibility of classes, methods, or fields

Available In:

 |  | 

<div>Types should be used in lambdas</div> <div> Code Smell</div>
<div>"java.time" classes should be used for dates and times</div> <div> Code Smell</div>
<div>The names of methods with boolean return values should start with "is" or "has"</div> <div> Code Smell</div>
<div>Files should contain only one top-level class or interface each</div> <div> Code Smell</div>