sonar RULES Products

- 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...

- Abstract class names should comply with a naming convention — Code Smell
- Strings literals should be placed on the left side when checking for equality — Code Smell
- Files should contain an empty newline at the end — Code Smell
- Source code should be indented consistently — Code Smell
- A close curly brace should be located at the beginning of a line — Code Smell
- Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines — Code Smell
- Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line — Code Smell
- An open curly brace should be located at the beginning of a line — Code Smell
- An open curly brace should be located at the end of a line — Code Smell
- Tabulation characters should not be used — Code Smell
- Functions should not be defined with a variable number of arguments — Code Smell

Constructor injection should be used instead of field injection

Analyze your code

Bug Major spring design jee pitfall

Field injection seems like a tidy way to get your classes what they need to do their jobs, but it's really a `NullPointerException` waiting to happen unless all your class constructors are `private`. That's because any class instances that are constructed by callers, rather than instantiated by a Dependency Injection framework compliant with the JSR-330 (Spring, Guice, ...), won't have the ability to perform the field injection.

Instead `@Inject` should be moved to the constructor and the fields required as constructor parameters.

This rule raises an issue when classes with non-`private` constructors (including the default constructor) use field injection.

Noncompliant Code Example

```
class MyComponent {  // Anyone can call the default construc

  @Inject MyCollaborator collaborator;  // Noncompliant

  public void myBusinessMethod() {
    collaborator.doSomething();  // this will fail in classe
  }
}
```

Compliant Solution

```
class MyComponent {

  private final MyCollaborator collaborator;

  @Inject
  public MyComponent(MyCollaborator collaborator) {
    Assert.notNull(collaborator, "MyCollaborator must not be
    this.collaborator = collaborator;
  }

  public void myBusinessMethod() {
    collaborator.doSomething();
  }
}
```

Available In:
sonarlint sonarcloud sonarqube

Local-Variable Type Inference should be used

Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

Code Smell

Track uses of disallowed classes

Code Smell

Track uses of "@SuppressWarnings" annotations

Code Smell