




 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... 🔍

Deprecated code should be removed

Code Smell

Annotated Mockito objects should be initialized

Bug

Custom resources should be closed

Bug

Threads should not be started in constructors

Code Smell

"main" should not "throw" anything

Code Smell

Track lack of copyright and license headers

Code Smell

Octal values should not be used

Code Smell

Exit methods should not be called

Code Smell

HTTP response headers should not be vulnerable to injection attacks

Vulnerability

Members of Spring components should be injected

Vulnerability

Classes should not be loaded dynamically

Vulnerability

Equality operators should not be used in "for" loop termination conditions

Code Smell

"@Deprecated" code marked for removal should never be used

Analyze your code

Code Smell

Major

cwe obsolete cert

Java 9 introduced a flag for the @Deprecated annotation, which allows to explicitly say if the deprecated code is planned to be removed at some point or not. This is done using forRemoval=true as annotation parameter. The javadoc of the annotation explicitly mention the following:

If true, it means that this API element is earmarked for removal in a future release.

If false, the API element is deprecated, but there is currently no intention to remove it in a future release.

While usually deprecated classes, interfaces, and their deprecated members should be avoided rather than used, inherited or extended, those already marked for removal are much more sensitive to causing trouble in your code soon. Consequently, any usage of such deprecated code should be avoided or removed.

Noncompliant Code Example

```
/**
 * @deprecated As of release 1.3, replaced by {@link #Fee}.
 */
@Deprecated(forRemoval=true)
public class Foo { ... }

public class Bar {
    /**
     * @deprecated As of release 1.7, replaced by {@link #doT
     */
    @Deprecated(forRemoval=true)
    public void doTheThing() { ... }

    public void doTheThingBetter() { ... }

    /**
     * @deprecated As of release 1.14 due to poor performances
     */
    @Deprecated(forRemoval=false)
    public void doTheOtherThing() { ... }
}





public class Qix extends Bar {
    @Override
    public void doTheThing() { ... } // Noncompliant; don't ov
}

public class Bar extends Foo { // Noncompliant; Foo is depr

    public void myMethod() {
        Bar bar = new Bar(); // okay; the class isn't deprecate
        bar.doTheThing(); // Noncompliant; doTheThing method is
        bar.doTheOtherThing(); // Okay; deprecated, but not mark
```

https://rules.sonarsource.com/java/RSPEC-5738

1/2

"Bean Validation" (JSR 380) should be properly configured  Code Smell
Spring beans should be considered by "@ComponentScan"  Code Smell
Number patterns should be regular  Code Smell
Lazy initialization of "static" fields should be "synchronized"  Code Smell

```
}  
}
```

See

- [MITRE, CWE-477](#) - Use of Obsolete Functions
- [CERT, MET02-J](#) - Do not use deprecated or obsolete classes or methods
- RSPEC-1874 for standard deprecation use

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 

© 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](#)