




 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... 🔍

Using hardcoded IP addresses is security-sensitive

Security Hotspot

'serialVersionUID' field should not be set to '0L' in records

Code Smell

Permitted types of a sealed class should be omitted if they are declared in the same file

Code Smell

Switch arrow labels should not use redundant keywords

Code Smell

Text blocks should not be used in complex expressions

Code Smell

Pattern Matching for "instanceof" operator should be used instead of simple "instanceof" + cast

Code Smell

Call to Mockito method "verify", "when" or "given" should be simplified

Code Smell

Character classes should be preferred over reluctant quantifiers in regular expressions

Code Smell

Consecutive AssertJ "assertThat" statements should be chained

Code Smell

Chained AssertJ assertions should be simplified to the corresponding dedicated assertion

Code Smell

Exception testing via JUnit @Test annotation should be avoided

### Non-serializable classes should not be written

Analyze your code

Bug

Major

serialization

Nothing in a non-serializable class will be written out to file, and attempting to serialize such a class will result in an exception being thrown. Only a class that implements `Serializable` or one that extends such a class can successfully be serialized (or de-serialized).

#### Noncompliant Code Example

```
public class Vegetable { // neither implements Serializable
//...
}

public class Menu {
    public void meal() throws IOException {
        Vegetable veg;
//...
        FileOutputStream fout = new FileOutputStream(veg.getName
ObjectOutputStream oos = new ObjectOutputStream(fout);
        oos.writeObject(veg); // Noncompliant. Nothing will be
    }
}
```

#### Compliant Solution

```
public class Vegetable implements Serializable { // can now
//...
}

public class Menu {
    public void meal() throws IOException {
        Vegetable veg;
//...
        FileOutputStream fout = new FileOutputStream(veg.getName
ObjectOutputStream oos = new ObjectOutputStream(fout);
        oos.writeObject(veg);
    }
}
```





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](#)

https://rules.sonarsource.com/java/RSPEC-2118

1/2

 Code Smell
<b>Escape sequences should not be used in text blocks</b>  Code Smell
<b>Simple string literal should be used for single line strings</b>  Code Smell
<b>Boxed "Boolean" should be avoided in boolean expressions</b>  Code Smell
<b>Type parameters should not shadow other type parameters</b>