


-  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... 🔍

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

URIs should not be hardcoded

Analyze your code

Code Smell

Minor

android cert

Hard coding a URI makes it difficult to test a program: path literals are not always portable across operating systems, a given absolute path may not exist on a specific test environment, a specified Internet URL may not be available when executing the tests, production environment filesystems usually differ from the development environment, ...etc. For all those reasons, a URI should never be hard coded. Instead, it should be replaced by customizable parameter.

Further even if the elements of a URI are obtained dynamically, portability can still be limited if the path-delimiters are hard-coded.

This rule raises an issue when URI's or path delimiters are hard coded.

Noncompliant Code Example

```
public class Foo {
    public Collection<User> listUsers() {
        File userList = new File("/home/mylogin/Dev/users.txt");
        Collection<User> users = parse(userList);
        return users;
    }
}
```

Compliant Solution

```
public class Foo {
    // Configuration is a class that returns customizable prop
    private Configuration config;
    public Foo(Configuration myConfig) {
        this.config = myConfig;
    }
    public Collection<User> listUsers() {
        // Find here the way to get the correct folder, in this
        String listingFolder = config.getProperty("myApplication
        // and use this parameter instead of the hard coded path
        File userList = new File(listingFolder, "users.txt"); //
        Collection<User> users = parse(userList);
        return users;
    }
}
```

See

- [CERT, MSC03-J](#) - Never hard code sensitive information

Available In:





sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-1075

1/2

<div>Local-Variable Type Inference should be used</div> <div> Code Smell</div>
<div>Migrate your tests from JUnit4 to the new JUnit5 annotations</div> <div> Code Smell</div>
<div>Track uses of disallowed classes</div> <div> Code Smell</div>
<div>Track uses of "@SuppressWarnings" annotations</div> <div> Code Smell</div>

© 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](#)