

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

## Classes and methods that rely on the default system encoding should not be used

Analyze your code

Code Smell

Minor

unpredictable cert i18n





Using classes and methods that rely on the default system encoding can result in code that works fine in its "home" environment. But that code may break for customers who use different encodings in ways that are extremely difficult to diagnose and nearly, if not completely, impossible to reproduce when it's time to fix them.

This rule detects uses of the following classes and methods:

- `FileReader`
- `FileWriter`
- String constructors with a `byte[]` argument but no `Charset` argument
  - `String(byte[] bytes)`
  - `String(byte[] bytes, int offset, int length)`
- `String.getBytes()`
- `String.getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)`
- `InputStreamReader(InputStream in)`
- `OutputStreamWriter(OutputStream out)`
- `ByteArrayOutputStream.toString()`
- Some `Formatter` constructors
  - `Formatter(String fileName)`
  - `Formatter(File file)`
  - `Formatter(OutputStream os)`
- Some `Scanner` constructors
  - `Scanner(File source)`
  - `Scanner(Path source)`
  - `Scanner(InputStream source)`
- Some `PrintStream` constructors
  - `PrintStream(File file)`
  - `PrintStream(OutputStream out)`
  - `PrintStream(OutputStream out, boolean autoFlush)`
  - `PrintStream(String fileName)`
- Some `PrintWriter` constructors
  - `PrintWriter(File file)`
  - `PrintWriter(OutputStream out)`
  - `PrintWriter(OutputStream out, boolean autoFlush)`
  - `PrintWriter(String fileName)`
- methods from Apache commons-io library which accept an encoding argument when that argument is null, and overloads of those methods that omit the encoding argument
  - `IOUtils.copy(InputStream, Writer)`
  - `IOUtils.copy(Reader, OutputStream)`
  - `IOUtils.readLines(InputStream)`
  - `IOUtils.toByteArray(Reader)`
  - `IOUtils.toByteArray(String)`
  - `IOUtils.toCharArray(InputStream)`
  - `IOUtils.toInputStream(TypeCriteria.subtypeOf(CharSequence))`
  - `IOUtils.toString(byte[])`
  - `IOUtils.toString(URI)`
  - `IOUtils.toString(URL)`
  - `IOUtils.write(char[], OutputStream)`
  - `IOUtils.write(CharSequence, OutputStream)`

https://rules.sonarsource.com/java/RSPEC-1943

1/2



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

- `IOUtils.writeLines(Collection, String, OutputStream)`
- `FileUtils.readFileToString(File)`
- `FileUtils.readLines(File)`
- `FileUtils.write(File, CharSequence)`
- `FileUtils.write(File, CharSequence, boolean)`
- `FileUtils.writeStringToFile(File, String)`

**See**

- [CERT, STR04-J](#) - Use compatible character encodings when communicating string data between JVMs
- [CERT, STR50-J](#) - Use the appropriate method for counting characters in a string

Available In:

 |  | 