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

"read(byte[],int,int)" should be overridden

Analyze your code

Code Smell Minor performance

When directly subclassing `java.io.InputStream` or `java.io.FilterInputStream`, the only requirement is that you implement the method `read()`. However most uses for such streams don't read a single byte at a time and the default implementation for `read(byte[],int,int)` will call `read(int)` for every single byte in the array which can create a lot of overhead and is utterly inefficient. It is therefore strongly recommended that subclasses provide an efficient implementation of `read(byte[],int,int)`.

This rule raises an issue when a direct subclass of `java.io.InputStream` or `java.io.FilterInputStream` doesn't provide an override of `read(byte[],int,int)`.

Noncompliant Code Example

```
public class MyInputStream extends java.io.InputStream {
  private FileInputStream fin;

  public MyInputStream(File file) throws IOException {
    fin = new FileInputStream(file);
  }

  @Override
  public int read() throws IOException {
    return fin.read();
  }
}
```

Compliant Solution

```
public class MyInputStream extends java.io.InputStream {
  private FileInputStream fin;

  public MyInputStream(File file) throws IOException {
    fin = new FileInputStream(file);
  }

  @Override
  public int read() throws IOException {
    return fin.read();
  }

  @Override
  public int read(byte[] b, int off, int len) throws IOExcep
    return fin.read(b, off, len);
  }
}
```

Exceptions

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

This rule doesn't raise an issue when the class is declared `abstract`.

Available In:

sonarlint sonarcloud sonarqube