




 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

An abstract class should have both abstract and concrete methods

Analyze your code

Code Smell Minor ? convention

The purpose of an abstract class is to provide some heritable behaviors while also defining methods which must be implemented by sub-classes.

A class with no abstract methods that was made abstract purely to prevent instantiation should be converted to a concrete class (i.e. remove the abstract keyword) with a private constructor.

A class with only abstract methods and no inheritable behavior should be converted to an interface.

Noncompliant Code Example

```
public abstract class Animal { // Noncompliant; should be a
    abstract void move();
    abstract void feed();
}

public abstract class Color { // Noncompliant; should be co
    private int red = 0;
    private int green = 0;
    private int blue = 0;

    public int getRed() {
        return red;
    }
}
```

Compliant Solution

```
public interface Animal {
    void move();
    void feed();
}





public class Color {
    private int red = 0;
    private int green = 0;
    private int blue = 0;

    private Color () {}

    public int getRed() {
        return red;
    }
}

public abstract class Lamp {

    private boolean switchLamp=false;
```

<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>

```
public abstract void glow();

public void flipSwitch() {
    switchLamp = !switchLamp;
    if (switchLamp) {
        glow();
    }
}
```

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](#)