




 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

"equals" methods should be symmetric and work for subclasses

Analyze your code

BugMinorcert

A key facet of the equals contract is that if `a.equals(b)` then `b.equals(a)`, i.e. that the relationship is symmetric.

Using `instanceof` breaks the contract when there are subclasses, because while the child is an `instanceof` the parent, the parent is not an `instanceof` the child. For instance, assume that `Raspberry` extends `Fruit` and adds some fields (requiring a new implementation of `equals`):

```
Fruit fruit = new Fruit();
Raspberry raspberry = new Raspberry();

if (raspberry instanceof Fruit) { ... } // true
if (fruit instanceof Raspberry) { ... } // false
```

If similar `instanceof` checks were used in the classes' `equals` methods, the symmetry principle would be broken:

```
raspberry.equals(fruit); // false
fruit.equals(raspberry); //true
```

Additionally, non `final` classes shouldn't use a hardcoded class name in the `equals` method because doing so breaks the method for subclasses. Instead, make the comparison dynamic.

Further, comparing to an unrelated class type breaks the contract for that unrelated type, because while `thisClass.equals(unrelatedClass)` can return true, `unrelatedClass.equals(thisClass)` will not.





Noncompliant Code Example

```
public class Fruit extends Food {
    private Season ripe;

    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (Fruit.class == obj.getClass()) { // Noncompliant; br
            return ripe.equals(((Fruit)obj).getRipe());
        }
        if (obj instanceof Fruit ) { // Noncompliant; broken fo
            return ripe.equals(((Fruit)obj).getRipe());
        }
        else if (obj instanceof Season) { // Noncompliant; symme
            // ...
        }
    }
}
```

https://rules.sonarsource.com/java/RSPEC-2162

1/2

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

```
}  
//...
```

Compliant Solution

```
public class Fruit extends Food {  
    private Season ripe;  
  
    public boolean equals(Object obj) {  
        if (obj == this) {  
            return true;  
        }  
        if (obj == null) {  
            return false;  
        }  
        if (this.getClass() == obj.getClass()) {  
            return ripe.equals(((Fruit)obj).getRipe());  
        }  
        return false;  
    }  
}
```

See

- [CERT, MET08-J](#) - Preserve the equality contract when overriding the equals() method

Available In:

