




 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... 🔍


 Bug


Child class methods named for parent class methods should be overrides





 Inappropriate "Collection" calls should not be made





 Silly equality checks should not be made





 Dissimilar primitive wrappers should not be used with the ternary operator without explicit casting




 "InterruptedException" should not be ignored



 Classes extending java.lang.Thread should override the "run" method



 "Double.longBitsToDouble" should not be used for "int"



 Values should not be uselessly incremented



 Silly String operations should not be made



 Non-serializable classes should not be written





 "hashCode" and "toString" should not be called on array instances




### Method overrides should not change contracts

Analyze your code

 Code Smell

 Critical

 suspicious

Because a subclass instance may be cast to and treated as an instance of the superclass, overriding methods should uphold the aspects of the superclass contract that relate to the Liskov Substitution Principle. Specifically, if the parameters or return type of the superclass method are marked with any of the following: @Nullable, @CheckForNull, @NotNull, @NonNull, and @Nonnull, then subclass parameters are not allowed to tighten the contract, and return values are not allowed to loosen it.

#### Noncompliant Code Example

```
public class Fruit {  
  
    private Season ripe;  
    private String color;  
  
    public void setRipe(@Nullable Season ripe) {  
        this.ripe = ripe;  
    }  
  
    public @NotNull Integer getProtein() {  
        return 12;  
    }  
}  
  
public class Raspberry extends Fruit {  
  
    public void setRipe(@NotNull Season ripe) { // Noncompliant  
        this.ripe = ripe;  
    }  
  
    public @Nullable Integer getProtein() { // Noncompliant  
        return null;  
    }  
}
```

See

- [Wikipedia - Liskov substitution principle](#)

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](#)

https://rules.sonarsource.com/java/RSPEC-2638

1/2

 Bug
<p>Collections should not be passed as arguments to their own methods</p>  Bug
<p>"BigDecimal(double)" should not be used</p>  Bug
<p>Invalid "Date" values should not be used</p>  Bug
<p>Reflection should not be used to check non-runtime annotations</p>