




 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

thrown
Code Smell
Public methods should throw at most one checked exception
Code Smell
"switch case" clauses should not have too many lines of code
Code Smell
Methods should not have too many return statements
Code Smell
Magic numbers should not be used
Code Smell
Files should not have too many lines of code
Code Smell
Lines should not be too long
Code Smell
JEE applications should not "getClassLoader"
Bug
Math should not be performed on floats
Bug
"equals" methods should be symmetric and work for subclasses
Bug
Literal suffixes should be upper case
Code Smell
Unicode-aware versions of character classes should be preferred
Code Smell
Use Java 12 "switch" expression

Classes named like "Exception" should extend "Exception" or a subclass

Analyze your code

- Code Smell
- Major ?
- convention error-handling pitfall

Clear, communicative naming is important in code. It helps maintainers and API users understand the intentions for and uses of a unit of code. Using "exception" in the name of a class that does not extend `Exception` or one of its subclasses is a clear violation of the expectation that a class' name will indicate what it is and/or does.

Noncompliant Code Example

```
public class FruitException { // Noncompliant; this has not
    private Fruit expected;
    private String unusualCharacteristics;
    private boolean appropriateForCommercialExploitation;
    // ...
}

public class CarException { // Noncompliant; the extends cl
    public CarException(String message, Throwable cause) {
        // ...
    }
}
```





Compliant Solution

```
public class FruitSport {
    private Fruit expected;
    private String unusualCharacteristics;
    private boolean appropriateForCommercialExploitation;
    // ...
}

public class CarException extends Exception {
    public CarException(String message, Throwable cause) {
        // ...
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

<div>Use Java 12 switch expression</div> <div> Code Smell</div>
<div>"serialVersionUID" should not be declared blindly</div> <div> Code Smell</div>
<div>"Stream.collect()" calls should not be redundant</div> <div> Code Smell</div>
<div>Local constants should follow naming conventions for constants</div> <div> Code Smell</div>
<div>Unit tests should throw exceptions</div>