




 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

### "throws" declarations should not be superfluous

Analyze your code

Code Smell

Minor

Quick Fix

error-handling unused redundant clumsy

An exception in a throws declaration in Java is superfluous if it is:

- listed multiple times
- a subclass of another listed exception
- completely unnecessary because the declared exception type cannot actually be thrown

#### Noncompliant Code Example

```
void foo() throws MyException, MyException {} // Noncompliant
void bar() throws Throwable, Exception {} // Noncompliant;
```

#### Compliant Solution

```
void foo() throws MyException {}
void bar() throws Throwable {}
```

#### Exceptions

The rule will not raise any issue for exceptions that cannot be thrown from the method body:

- in overriding and implementation methods
- in interface default methods
- in non-private methods that only throw, have empty bodies, or a single return statement.
- in overridable methods (non-final, or not member of a final class, non-static, non-private), if the exception is documented with a proper JavaDoc

Also, the rule won't raise issues on RuntimeException, or one of its descendants, because explicating runtime exceptions which could be thrown can ultimately help the method's users, and can even be considered as good practice.

```
class A extends B {
    @Override
    void doSomething() throws IOException {
        compute(a);
    }

    public void foo() throws IOException {}





    public void qix() throws MyRuntimeException {}

    protected void bar() throws IOException {
        throw new UnsupportedOperationException("This method sho
    }

    Object foobar(String s) throws IOException {
```

https://rules.sonarsource.com/java/RSPEC-1130

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

```
        return null;
    }

    /**
     * @throws IOException Overriding classes may throw this e
     */
    protected void print() throws IOException { // no issue, m
        System.out.println("foo");
    }
}
```

Available In:

 |  | 