sonar RULES

Products

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

Classes that override "clone" should be "Cloneable" and call "super.clone()"

Analyze your code

Code Smell Minor ? convention cwe cert

`Cloneable` is the marker `Interface` that indicates that `clone()` may be called on an object. Overriding `clone()` without implementing `Cloneable` can be useful if you want to control how subclasses clone themselves, but otherwise, it's probably a mistake.

The usual convention for `Object.clone()` according to Oracle's Javadoc is:

1. `x.clone() != x`
2. `x.clone().getClass() == x.getClass()`
3. `x.clone().equals(x)`

Obtaining the object that will be returned by calling `super.clone()` helps to satisfy those invariants:

1. `super.clone()` returns a new object instance
2. `super.clone()` returns an object of the same type as the one `clone()` was called on
3. `Object.clone()` performs a shallow copy of the object's state

Noncompliant Code Example

```
class BaseClass {  // Noncompliant; should implement Cloneab
  @Override
  public Object clone() throws CloneNotSupportedException {
    return new BaseClass();
  }
}

class DerivedClass extends BaseClass implements Cloneable {
  /* Does not override clone() */

  public void sayHello() {
    System.out.println("Hello, world!");
  }
}

class Application {
  public static void main(String[] args) throws Exception {
    DerivedClass instance = new DerivedClass();
    ((DerivedClass) instance.clone()).sayHello();
  }
}
```

Compliant Solution

```
class BaseClass implements Cloneable {
  @Override
  public Object clone() throws CloneNotSupportedException {
    return super.clone();
  }
```

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

```
}

class DerivedClass extends BaseClass implements Cloneable {
  /* Does not override clone() */

  public void sayHello() {
    System.out.println("Hello, world!");
  }
}

class Application {
  public static void main(String[] args) throws Exception {
    DerivedClass instance = new DerivedClass();
    ((DerivedClass) instance.clone()).sayHello();
  }
}
```

See

- MITRE, CWE-580 - clone() Method Without super.clone()
- CERT, MET53-J. - Ensure that the clone() method calls super.clone()

Available In:

sonarlint | sonarcloud | sonarqube