Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

Getters and setters should be synchronized in pairs

Analyze your code

Bug Major multi-threading cert

When one part of a getter/setter pair is `synchronized` the other part should be too. Failure to synchronize both sides of a pair may result in inconsistent behavior at runtime as callers access an inconsistent method state.

This rule raises an issue when either the method or the contents of one method in a getter/setter pair are synchrnoized but the other is not.

Noncompliant Code Example

```
public class Person {
  String name;
  int age;

  public synchronized void setName(String name) {
    this.name = name;
  }

  public String getName() {  // Noncompliant
    return this.name;
  }

  public void setAge(int age) {  // Noncompliant
    this.age = age;
  }

  public int getAge() {
    synchronized (this) {
      return this.age;
    }
  }
}
```

Compliant Solution

```
public class Person {
  String name;
  int age;

  public synchronized void setName(String name) {
    this.name = name;
  }

  public synchronized String getName() {
    return this.name;
  }

  public void setAge(int age) {
    synchronized (this) {
      this.age = age;
```

Collapsible "if" statements should be merged

Code Smell

Unused labels should be removed

Code Smell

Standard outputs should not be used directly to log anything

Code Smell

OS commands should not be vulnerable to argument injection attacks

Vulnerability

```
    }
  }

  public int getAge() {
    synchronized (this) {
      return this.age;
    }
  }
}
```

See

- CERT, VNA01-J. - Ensure visibility of shared references to immutable objects

Available In:

sonarlint | sonarcloud | sonarqube