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...

Code Smell

Classes with only "static" methods should not be instantiated
Code Smell

"Threads" should not be used where "Runnables" are expected
Code Smell

Inner class calls to super class methods should be unambiguous
Code Smell

Unused type parameters should be removed
Code Smell

Parameters should be passed in the correct order
Code Smell

"ResultSet.isLast()" should not be used
Code Smell

"static" members should be accessed statically
Code Smell

Silly math should not be performed
Code Smell

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

Exceptions should be either logged or rethrown but not both
Code Smell

Objects should not be created only to "getClass"
Code Smell

Primitives should not be boxed just for

"iterator" should not return "this"

Analyze your code

Bug Major pitfall

There are two classes in the Java standard library that deal with iterations: `Iterable<T>` and `Iterator<T>`. An `Iterable<T>` represents a data structure that can be the target of the "for-each loop" statement, and an `Iterator<T>` represents the state of an ongoing traversal. An `Iterable<T>` is generally expected to support multiple traversals.

An `Iterator<T>` that also implements `Iterable<t>` by returning itself as its `iterator()` will not support multiple traversals since its state will be carried over.

This rule raises an issue when the `iterator()` method of a class implementing both `Iterable<T>` and `Iterator<t>` returns `this`.

Noncompliant Code Example

```
class FooIterator implements Iterator<Foo>, Iterable<Foo> {
  private Foo[] seq;
  private int idx = 0;

  public boolean hasNext() {
    return idx < seq.length;
  }

  public Foo next() {
    return seq[idx++];
  }

  public Iterator<Foo> iterator() {
    return this; // Noncompliant
  }
  // ...
}
```

Compliant Solution

```
class FooSequence implements Iterable<Foo> {
  private Foo[] seq;

  public Iterator<Foo> iterator() {
    return new Iterator<Foo>() {
      private int idx = 0;

      public boolean hasNext() {
        return idx < seq.length;
      }

      public Foo next() {
        return seq[idx++];
      }
    };
  }
```

"String" conversion

Code Smell

Constructors should not be used to instantiate "String", "BigInteger", "BigDecimal" and primitive-wrapper classes

Code Smell

"URL.hashCode" and "URL.equals" should be avoided

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

```
    // ...
  }
```

Available In:

sonarlint sonarcloud sonarqube