Java static code analysis

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

Hard-coded credentials are security-sensitive
Security Hotspot

Methods returns should not be invariant
Code Smell

"ThreadGroup" should not be used
Code Smell

"clone" should not be overridden
Code Smell

Assertions should be complete
Code Smell

Tests should include assertions
Code Smell

Silly bit operations should not be performed
Code Smell

Child class fields should not shadow parent class fields
Code Smell

JUnit test cases should call super methods
Code Smell

TestCases should contain tests
Code Smell

Short-circuit logic should be used in boolean contexts
Code Smell

Methods and field names should not be the same or differ only by capitalization
Code Smell

"@SpringBootApplication" and "@ComponentScan" should not be used in the default package

Analyze your code

Bug Blocker spring

`@ComponentScan` is used to determine which Spring Beans are available in the application context. The packages to scan can be configured thanks to the `basePackageClasses` or `basePackages` (or its alias `value`) parameters. If neither parameter is configured, `@ComponentScan` will consider only the package of the class annotated with it. When `@ComponentScan` is used on a class belonging to the default package, the entire classpath will be scanned.

This will slow-down the start-up of the application and it is likely the application will fail to start with an `BeanDefinitionStoreException` because you ended up scanning the Spring Framework package itself.

This rule raises an issue when:

- `@ComponentScan`, `@SpringBootApplication` and `@ServletComponentScan` are used on a class belonging to the default package
- `@ComponentScan` is explicitly configured with the default package

Noncompliant Code Example

```
import org.springframework.boot.SpringApplication;

@SpringBootApplication // Noncompliant; RootBootApp is decla
public class RootBootApp {
...
}
```

```
@ComponentScan("")
public class Application {
...
}
```

Compliant Solution

```
package hello;

import org.springframework.boot.SpringApplication;

@SpringBootApplication // Compliant; RootBootApp belongs to
public class RootBootApp {
...
}
```

Available In:
sonarlint sonarcloud sonarqube

Switch cases should end with an unconditional "break" statement

Code Smell

"switch" statements should not contain non-case labels

Code Smell

Future keywords should not be used as names

Code Smell

Thread suspensions should not be vulnerable to Denial of Service attacks

Vulnerability