IMU-Android
1.0

java:Sonar way

2022-04-30

目录

1. IMU-Android

报告提供了项目指标的概要，显示了与项目质量相关的最重要的指标。如果需要获取更详细的信息，请登陆网站进一步查询。

报告的项目为IMU-Android，生成时间为2022-04-30，使用的质量配置为 java:Sonar way，共计452条规则。

1.1. 概述

编码问题

Bug 8	可靠性修复工作 1h0min
漏洞 0	安全修复工作 0min
坏味道 161	技术债务 13h33min

169 问题	开启问题	169
	重开问题	0
	确认问题	0
	误判问题	0
	不修复的问题	0
	已解决的问题	139
	已删除的问题	0
	阻断	1
	严重	12
	主要	36
	次要	119
	提示	1

静态分析

项目规模

1356 代码行数	行数	1643
	方法	75
	类	9
	文件	8
	目录	N/A
	重复行(%)	0.0

复杂度

233 复杂度	文件	29.1

注释(%)

6.9 注释(%)	注释行数	101

1.2. 问题分析

违反最多的规则TOP10	
Unnecessary imports should be removed	28
Sections of code should not be commented out	13
Field names should comply with a naming convention	12
Method names should comply with a naming convention	11
Public constants and fields initialized at declaration should be "static final" rather than merely "final"	11
Package names should comply with a naming convention	7
Private fields only used as local variables in methods should become local variables	7
Unused "private" fields should be removed	7
"private" methods called only by inner classes should be moved to those classes	6

Local variable and method parameter names should comply with a naming convention	5

违规最多的文件TOP5	
MainActivity.java	73
MyService.java	59
KalManFilter.java	11
DataProcess.java	7
GestureOpAdapter.java	6

复杂度最高的文件TOP5	
MainActivity.java	109
MyService.java	93
GestureOpAdapter.java	8
IMUBytes.java	8
Operation.java	7

重复行最多的文件TOP5
No duplications

1.3. 问题详情

规则	Unnecessary imports should be removed

规则描述	The imports part of a file should be handled by the Integrated Development Environment (IDE), not manually by the developer.
	Unused and useless imports should not occur if that is the case.
	Leaving them in reduces the code's readability, since their presence can be confusing.
	Noncompliant Code Example
	
	package my.company;
	
	import java.lang.String; // Noncompliant; java.lang classes are always implicitly imported
	import my.company.SomeClass; // Noncompliant; same-package files are always implicitly imported
	import java.io.File; // Noncompliant; File is not used
	
	import my.company2.SomeType;
	import my.company2.SomeType; // Noncompliant; 'SomeType' is already imported
	
	class ExampleClass {
	
	public String someString;
	public SomeType something;
	
	}
	
	Exceptions
	Imports for types mentioned in comments, such as Javadocs, are ignored.

文件名称	违规行
IMUBytes.java	3
MainActivity.java	51, 58, 65, 70, 73
GestureOpAdapter.java	5, 12, 15
SpinAdapter.java	5, 8, 9
Operation.java	3
MainActivity.java	6, 26, 37, 44, 57, 68, 75, 76, 77, 80, 84
MyService.java	13, 18, 26, 36

规则	Sections of code should not be commented out
规则描述	Programmers should not comment out code as it bloats programs and reduces readability. Unused code should be deleted and can be retrieved from source control history if required.

文件名称	违规行
MainActivity.java	203, 219, 436, 598
MyService.java	86, 175, 179, 190, 195, 229, 240, 250, 288

规则	Field names should comply with a naming convention

规则描述	Sharing some naming conventions is a key point to make it possible for a team to efficiently collaborate. This rule allows to check that field
	names match a provided regular expression.
	Noncompliant Code Example
	With the default regular expression ^[a-z][a-zA-Z0-9]*$:
	
	class MyClass {
	private int my_field;
	}
	
	Compliant Solution
	
	class MyClass {
	private int myField;
	}

文件名称	违规行
MainActivity.java	87, 88, 89, 90, 95
MyService.java	46, 47, 48, 49, 50, 54
KalManFilter.java	8

规则	Public constants and fields initialized at declaration should be "static final" rather than merely "final"

规则描述	Making a public constant just final as opposed to static final leads to duplicating its value for every
	instance of the class, uselessly increasing the amount of memory required to execute the application.
	Further, when a non- public , final field isn't also static , it implies that different instances can have
	different values. However, initializing a non- static final field in its declaration forces every instance to have the same value. So such fields should either be made static or initialized in the constructor.
	Noncompliant Code Example
	
	public class Myclass {
	public final int THRESHOLD = 3;
	}
	
	Compliant Solution
	
	public class Myclass {
	public static final int THRESHOLD = 3; // Compliant
	}
	
	Exceptions
	No issues are reported on final fields of inner classes whose type is not a primitive or a String. Indeed according to the Java specification:
	
	An inner class is a nested class that is not explicitly or implicitly declared static. Inner classes may not declare static initializers (§8.7)
	or member interfaces. Inner classes may not declare static members, unless they are compile-time constant fields (§15.28).

文件名称	违规行
MainActivity.java	87, 88, 89, 90, 91
MyService.java	48, 49, 50, 51, 52
KalManFilter.java	8

规则	Method names should comply with a naming convention

规则描述	Shared naming conventions allow teams to collaborate efficiently. This rule checks that all method names match a provided regular expression. Noncompliant Code Example With default provided regular expression ^[a-z][a-zA-Z0-9]*$: public int DoSomething(){...} Compliant Solution public int doSomething(){...} Exceptions Overriding methods are excluded. @Override public int Do_Something(){...}

文件名称	违规行
DataProcess.java	9, 11, 13, 15
MainActivity.java	488, 726
MyService.java	378, 458, 496, 499, 502

规则	Private fields only used as local variables in methods should become local variables

规则描述	When the value of a private field is always assigned to in a class' methods before being read, then it is not being used to store class information. Therefore, it should become a local variable in the relevant methods to prevent any misunderstanding.
	Noncompliant Code Example
	
	public class Foo {
	  private int a;
	  private int b;
	
	  public void doSomething(int y) {
	  a = y + 5;
	  ...
	  if(a == 0) {
	  ...
	  }
	  ...
	  }
	
	  public void doSomethingElse(int y) {
	  b = y + 3;
	  ...
	  }
	}
	
	Compliant Solution
	
	public class Foo {
	
	  public void doSomething(int y) {
	  int a = y + 5;
	  ...
	  if(a == 0) {
	  ...
	  }
	  }
	
	  public void doSomethingElse(int y) {
	  int b = y + 3;
	  ...
	  }
	}
	
	Exceptions
	This rule doesn't raise any issue on annotated field.

文件名称	违规行
MainActivity.java	102, 111, 112, 113, 116, 134, 137

规则	Unused "private" fields should be removed

规则描述	If a private field is declared but not used in the program, it can be considered dead code and should therefore be removed. This will improve maintainability because developers will not wonder what the variable is used for.
	Note that this rule does not take reflection into account, which means that issues will be raised on private fields that are only accessed using the reflection API.
	Noncompliant Code Example
	
	public class MyClass {
	private int foo = 42;
	
	public int compute(int a) {
	return a * 42;
	}
	
	}
	
	Compliant Solution
	
	public class MyClass {
	public int compute(int a) {
	return a * 42;
	}
	}
	
	Exceptions
	The Java serialization runtime associates with each serializable class a version number, called serialVersionUID , which is used during deserialization to verify that the sender and receiver of a serialized object have loaded classes for that object that are compatible with respect to serialization.
	A serializable class can declare its own serialVersionUID explicitly by declaring a field named serialVersionUID that must be static, final, and of type long. By definition those serialVersionUID fields should not be reported by this rule:
	
	public class MyClass implements java.io.Serializable {
	private static final long serialVersionUID = 42L;
	}
	
	Moreover, this rule doesn't raise any issue on annotated fields.

文件名称	违规行
MainActivity.java	92
MyService.java	74, 78, 81, 87, 88
KalManFilter.java	12

规则	Package names should comply with a naming convention

规则描述	Shared coding conventions allow teams to collaborate efficiently. This rule checks that all package names match a provided regular expression. Noncompliant Code Example With the default regular expression ^[a-z_]+(\.[a-z_][a-z0-9_]*)*$: package org.exAmple; // Noncompliant Compliant Solution package org.example;

文件名称	违规行
GestureOpAdapter.java	1
SpinAdapter.java	1
IMUBytes.java	1
Operation.java	1
DataProcess.java	1
MyService.java	1
KalManFilter.java	1

规则	"private" methods called only by inner classes should be moved to those classes

规则描述	When a private method is only invoked by an inner class, there's no reason not to move it into that class. It will still have the same access to the outer class' members, but the outer class will be clearer and less cluttered.
	Noncompliant Code Example
	
	public class Outie {
	private int i=0;
	
	private void increment() { // Noncompliant
	i++;
	}
	
	public class Innie {
	public void doTheThing() {
	Outie.this.increment();
	}
	}
	}
	
	Compliant Solution
	
	public class Outie {
	private int i=0;
	
	public class Innie {
	public void doTheThing() {
	increment();
	}
	
	private void increment() {
	Outie.this.i++;
	}
	}
	}

文件名称	违规行
MainActivity.java	688
MyService.java	310, 368, 378, 391, 403

规则	Local variable and method parameter names should comply with a naming convention

规则描述	Shared naming conventions allow teams to collaborate effectively. This rule raises an issue when a local variable or function parameter name does not match the provided regular expression.
	Noncompliant Code Example
	With the default regular expression ^[a-z][a-zA-Z0-9]*$:
	
	public void doSomething(int my_param) {
	int LOCAL;
	...
	}
	
	Compliant Solution
	
	public void doSomething(int myParam) {
	int local;
	...
	}
	
	Exceptions
	Loop counters are ignored by this rule.
	
	for (int i_1 = 0; i_1 < limit; i_1++) { // Compliant
	// ...
	}
	
	as well as one-character catch variables:
	
	try {
	//...
	} catch (Exception e) { // Compliant
	}

文件名称	违规行
IMUBytes.java	38, 38
DataProcess.java	11, 15
MyService.java	325

规则	Overriding methods should do more than simply call the same method in the super class

规则描述	Overriding a method just to call the same method from the super class without performing any other actions is useless and misleading. The only time this is justified is in final overriding methods, where the effect is to lock in the parent class behavior. This rule ignores such overrides of equals , hashCode and toString .
	Noncompliant Code Example
	
	public void doSomething() {
	super.doSomething();
	}
	
	@Override
	public boolean isLegal(Action action) {
	return super.isLegal(action);
	}
	
	Compliant Solution
	
	@Override
	public boolean isLegal(Action action) { // Compliant - not simply forwarding the call
	return super.isLegal(new Action(/* ... */));
	}
	
	@Id
	@Override
	public int getId() { // Compliant - there is annotation different from @Override
	return super.getId();
	}

文件名称	违规行
MyService.java	160, 166, 265, 301, 306

规则	Class variable fields should not have public accessibility

规则描述	

	Public class variable fields do not respect the encapsulation principle and has three main disadvantages:
	
	Additional behavior such as validation cannot be added.
	The internal representation is exposed, and cannot be changed afterwards.
	Member values are subject to change from anywhere in the code and may not meet the programmer's assumptions.
	
	By using private attributes and accessor methods (set and get), unauthorized modifications are prevented.
	Noncompliant Code Example
	
	public class MyClass {
	
	public static final int SOME_CONSTANT = 0; // Compliant - constants are not checked
	
	public String firstName; // Noncompliant
	
	}
	
	Compliant Solution
	
	public class MyClass {
	
	public static final int SOME_CONSTANT = 0; // Compliant - constants are not checked
	
	private String firstName; // Compliant
	
	public String getFirstName() {
	return firstName;
	}
	
	public void setFirstName(String firstName) {
	this.firstName = firstName;
	}
	
	}
	
	Exceptions
	Because they are not modifiable, this rule ignores public final fields. Also, annotated fields, whatever the annotation(s) will be ignored, as annotations are often used by injection frameworks, which in exchange require having public fields.
	See
	
	MITRE, CWE-493 - Critical Public Variable Without Final Modifier

文件名称	违规行
MainActivity.java	125, 126, 127
MyService.java	90

规则	"public static" fields should be constant

规则描述	There is no good reason to declare a field "public" and "static" without also declaring it "final". Most of the time this is a kludge to share a
	state among several objects. But with this approach, any object can do whatever it wants with the shared state, such as setting it to
	null .
	Noncompliant Code Example
	
	public class Greeter {
	public static Foo foo = new Foo();
	...
	}
	
	Compliant Solution
	
	public class Greeter {
	public static final Foo FOO = new Foo();
	...
	}
	
	See
	
	MITRE, CWE-500 - Public Static Field Not Marked Final
	CERT OBJ10-J. - Do not use public static nonfinal fields

文件名称	违规行
MainActivity.java	125, 126, 127
MyService.java	90

规则	Methods should not be empty

规则描述	There are several reasons for a method not to have a method body:
	
	It is an unintentional omission, and should be fixed to prevent an unexpected behavior in production.
	It is not yet, or never will be, supported. In this case an UnsupportedOperationException should be thrown.
	The method is an intentionally-blank override. In this case a nested comment should explain the reason for the blank override.
	
	Noncompliant Code Example
	
	public void doSomething() {
	}
	
	public void doSomethingElse() {
	}
	
	Compliant Solution
	
	@Override
	public void doSomething() {
	// Do nothing because of X and Y.
	}
	
	@Override
	public void doSomethingElse() {
	throw new UnsupportedOperationException();
	}
	
	Exceptions
	Default (no-argument) constructors are ignored when there are other constructors in the class, as are empty methods in abstract classes.
	
	public abstract class Animal {
	void speak() { // default implementation ignored
	}
	}

文件名称	违规行
GestureOpAdapter.java	76
MainActivity.java	152
MyService.java	279
KalManFilter.java	16

规则	Unused local variables should be removed

规则描述	If a local variable is declared but not used, it is dead code and should be removed. Doing so will improve maintainability because developers will
	not wonder what the variable is used for.
	Noncompliant Code Example
	
	public int numberOfMinutes(int hours) {
	int seconds = 0; // seconds is never used
	return hours * 60;
	}
	
	Compliant Solution
	
	public int numberOfMinutes(int hours) {
	return hours * 60;
	}

文件名称	违规行
MainActivity.java	611
KalManFilter.java	28, 29, 30

规则	Unused assignments should be removed

规则描述	A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value
	only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it's not an error, it is at best a waste of resources.
	Therefore all calculated values should be used.
	Noncompliant Code Example
	
	i = a + b; // Noncompliant; calculation result not used before value is overwritten
	i = compute();
	
	Compliant Solution
	
	i = a + b;
	i += compute();
	
	Exceptions
	This rule ignores initializations to -1, 0, 1, null , true , false and "" .
	See
	
	MITRE, CWE-563 - Assignment to Variable without Use ('Unused Variable')
	CERT, MSC13-C. - Detect and remove unused values
	CERT, MSC56-J. - Detect and remove superfluous code and values

文件名称	违规行
MainActivity.java	611
KalManFilter.java	28, 29, 30

规则	Collection.isEmpty() should be used to test for emptiness

规则描述	Using Collection.size() to test for emptiness works, but using Collection.isEmpty() makes the code more readable and can be more performant. The time complexity of any isEmpty() method implementation should be O(1) whereas some implementations of size() can be O(n) .
	Noncompliant Code Example
	
	if (myCollection.size() == 0) { // Noncompliant
	/* ... */
	}
	
	Compliant Solution
	
	if (myCollection.isEmpty()) {
	/* ... */
	}

文件名称	违规行
MainActivity.java	661
MyService.java	369, 380

规则	"switch" statements should have "default" clauses

规则描述	The requirement for a final default clause is defensive programming. The clause should either take appropriate action, or contain a
	suitable comment as to why no action is taken.
	Noncompliant Code Example
	
	switch (param) { //missing default clause
	  case 0:
	  doSomething();
	  break;
	  case 1:
	  doSomethingElse();
	  break;
	}
	
	switch (param) {
	  default: // default clause should be the last one
	  error();
	  break;
	  case 0:
	  doSomething();
	  break;
	  case 1:
	  doSomethingElse();
	  break;
	}
	
	Compliant Solution
	
	switch (param) {
	  case 0:
	  doSomething();
	  break;
	  case 1:
	  doSomethingElse();
	  break;
	  default:
	  error();
	  break;
	}
	
	Exceptions
	If the switch parameter is an Enum and if all the constants of this enum are used in the case statements,
	then no default clause is expected.
	Example:
	
	public enum Day {
	  SUNDAY, MONDAY
	}
	...
	switch(day) {
	  case SUNDAY:
	  doSomething();
	  break;
	  case MONDAY:
	  doSomethingElse();
	  break;
	}
	
	See

	MITRE, CWE-478 - Missing Default Case in Switch Statement CERT, MSC01-C. - Strive for logical completeness

文件名称	违规行
MainActivity.java	326, 744
MyService.java	208

规则	Instance methods should not write to "static" fields

规则描述	Correctly updating a static field from a non-static method is tricky to get right and could easily lead to bugs if there are multiple class instances and/or multiple threads in play. Ideally, static fields are only updated from synchronized static methods.
	This rule raises an issue each time a static field is updated from a non-static method.
	Noncompliant Code Example
	
	public class MyClass {
	
	private static int count = 0;
	
	public void doSomething() {
	//...
	count++; // Noncompliant
	}
	}

文件名称	违规行
MainActivity.java	161, 194
MyService.java	503

规则	"switch" statements should have at least 3 "case" clauses

规则描述	switch statements are useful when there are many different cases depending on the value of the same expression.
	For just one or two cases however, the code will be more readable with if statements.
	Noncompliant Code Example
	
	switch (variable) {
	  case 0:
	  doSomething();
	  break;
	  default:
	  doSomethingElse();
	  break;
	}
	
	Compliant Solution
	
	if (variable == 0) {
	  doSomething();
	} else {
	  doSomethingElse();
	}

文件名称	违规行
MainActivity.java	326, 744
MyService.java	208

规则	Unused "private" methods should be removed

规则描述	private methods that are never executed are dead code: unnecessary, inoperative code that should be removed. Cleaning out dead code decreases the size of the maintained codebase, making it easier to understand the program and preventing bugs from being introduced.
	Note that this rule does not take reflection into account, which means that issues will be raised on private methods that are only accessed using the reflection API.
	Noncompliant Code Example
	
	public class Foo implements Serializable
	{
	private Foo(){} //Compliant, private empty constructor intentionally used to prevent any direct instantiation of a class.
	public static void doSomething(){
	Foo foo = new Foo();
	...
	}
	private void unusedPrivateMethod(){...}
	private void writeObject(ObjectOutputStream s){...} //Compliant, relates to the java serialization mechanism
	private void readObject(ObjectInputStream in){...} //Compliant, relates to the java serialization mechanism
	}
	
	Compliant Solution
	
	public class Foo implements Serializable
	{
	private Foo(){} //Compliant, private empty constructor intentionally used to prevent any direct instantiation of a class.
	public static void doSomething(){
	Foo foo = new Foo();
	...
	}
	
	private void writeObject(ObjectOutputStream s){...} //Compliant, relates to the java serialization mechanism
	
	private void readObject(ObjectInputStream in){...} //Compliant, relates to the java serialization mechanism
	}
	
	Exceptions
	This rule doesn't raise any issue on annotated methods.

文件名称	违规行
MainActivity.java	420, 427, 726

规则	The diamond operator ("<>") should be used

规则描述	Java 7 introduced the diamond operator (<>) to reduce the verbosity of generics code. For instance, instead of having to declare
	a List 's type in both its declaration and its constructor, you can now simplify the constructor declaration with <> ,
	and the compiler will infer the type.
	Note that this rule is automatically disabled when the project's sonar.java.source is lower than 7 .
	Noncompliant Code Example
	
	List<String> strings = new ArrayList<String>(); // Noncompliant
	Map<String,List<Integer>> map = new HashMap<String,List<Integer>>(); // Noncompliant
	
	Compliant Solution
	
	List<String> strings = new ArrayList<>();
	Map<String,List<Integer>> map = new HashMap<>();

文件名称	违规行
MainActivity.java	273
MyService.java	287

规则	Return values should not be ignored when they contain the operation status code

规则描述	When the return value of a function call contain the operation status code, this value should be tested to make sure the operation completed successfully.
	This rule raises an issue when the return values of the following are ignored:
	
	java.io.File operations that return a status code (except mkdirs)
	Iterator.hasNext()
	Enumeration.hasMoreElements()
	Lock.tryLock()
	non-void Condition.await* methods
	CountDownLatch.await(long, TimeUnit)
	Semaphore.tryAcquire
	BlockingQueue : offer , remove
	
	Noncompliant Code Example
	
	public void doSomething(File file, Lock lock) {
	file.delete(); // Noncompliant
	// ...
	lock.tryLock(); // Noncompliant
	}
	
	Compliant Solution
	
	public void doSomething(File file, Lock lock) {
	if (!lock.tryLock()) {
	// lock failed; take appropriate action
	}
	if (!file.delete()) {
	// file delete failed; take appropriate action
	}
	}
	
	See
	
	CERT, EXP00-J. - Do not ignore values returned by methods
	CERT, FIO02-J. - Detect and handle file-related errors
	MITRE, CWE-754 - Improper Check for Unusual Exceptional Conditions

文件名称	违规行
MainActivity.java	673, 674

规则	Anonymous inner classes containing only one method should become lambdas

规则描述	Before Java 8, the only way to partially support closures in Java was by using anonymous inner classes. But the syntax of anonymous classes may seem unwieldy and unclear.
	With Java 8, most uses of anonymous inner classes should be replaced by lambdas to highly increase the readability of the source code.
	Note that this rule is automatically disabled when the project's sonar.java.source is lower than 8 .
	Noncompliant Code Example
	
	myCollection.stream().map(new Mapper<String,String>() {
	public String map(String input) {
	return new StringBuilder(input).reverse().toString();
	}
	});
	
	Predicate<String> isEmpty = new Predicate<String> {
	boolean test(String myString) {
	return myString.isEmpty();
	}
	}
	
	Compliant Solution
	
	myCollection.stream().map(input -> new StringBuilder(input).reverse().toString());
	
	Predicate<String> isEmpty = myString -> myString.isEmpty();

文件名称	违规行
MainActivity.java	221, 751

规则	Method parameters, caught exceptions and foreach variables' initial values should not be ignored

规则描述	While it is technically correct to assign to parameters from within method bodies, doing so before the parameter value is read is likely a bug.
	Instead, initial values of parameters, caught exceptions, and foreach parameters should be, if not treated as final , then at least read
	before reassignment.
	Noncompliant Code Example
	
	public void doTheThing(String str, int i, List<String> strings) {
	str = Integer.toString(i); // Noncompliant
	
	for (String s : strings) {
	s = "hello world"; // Noncompliant
	}
	}

文件名称	违规行
GestureOpAdapter.java	58
SpinAdapter.java	51

规则	Static non-final field names should comply with a naming convention

规则描述	Shared naming conventions allow teams to collaborate efficiently. This rule checks that static non-final field names match a provided regular expression.
	Noncompliant Code Example
	With the default regular expression ^[a-z][a-zA-Z0-9]*$:
	
	public final class MyClass {
	private static String foo_bar;
	}
	
	Compliant Solution
	
	class MyClass {
	private static String fooBar;
	}

文件名称	违规行
MainActivity.java	126, 127

规则	Collapsible "if" statements should be merged

规则描述	Merging collapsible if statements increases the code's readability.
	Noncompliant Code Example
	
	if (file != null) {
	if (file.isFile() || file.isDirectory()) {
	/* ... */
	}
	}
	
	Compliant Solution
	
	if (file != null && isFileOrDirectory(file)) {
	/* ... */
	}
	
	private static boolean isFileOrDirectory(File file) {
	return file.isFile() || file.isDirectory();
	}

文件名称	违规行
MainActivity.java	192, 314

规则	Raw byte values should not be used in bitwise operations in combination with shifts

规则描述	When reading bytes in order to build other primitive values such as int s or long s, the byte values are automatically promoted, but that promotion can have unexpected results.
	For instance, the binary representation of the integer 640 is 0b0000_0010_1000_0000 , which can also be written with the array of (unsigned) bytes [2, 128] . However, since Java uses two's complement, the representation of the integer in signed bytes will be [2, -128] (because the byte 0b1000_0000 is promoted to the int 0b1111_1111_1111_1111_1111_1111_1000_0000). Consequently, trying to reconstruct the initial integer by shifting and adding the values of the bytes without taking care of the sign will not produce the expected result.
	To prevent such accidental value conversion, use bitwise and (&) to combine the byte value with 0xff (255) and turn all the higher bits back off.
	This rule raises an issue any time a byte value is used as an operand without & 0xff , when combined with shifts.
	Noncompliant Code Example
	
	int intFromBuffer() {
	int result = 0;
	for (int i = 0; i < 4; i++) {
	result = (result << 8) | readByte(); // Noncompliant
	}
	return result;
	}
	
	Compliant Solution
	
	int intFromBuffer() {
	int result = 0;
	for (int i = 0; i < 4; i++) {
	result = (result << 8) | (readByte() & 0xff);
	}
	return result;
	}
	
	See
	
	CERT, NUM52-J. - Be aware of numeric promotion behavior

文件名称	违规行
IMUBytes.java	43, 47

规则	Cognitive Complexity of methods should not be too high
规则描述	Cognitive Complexity is a measure of how hard the control flow of a method is to understand. Methods with high Cognitive Complexity will be difficult to maintain. See Cognitive Complexity

文件名称	违规行
MainActivity.java	488, 743

规则	Track uses of "TODO" tags

规则描述	TODO tags are commonly used to mark places where some more code is required, but which the developer wants to implement later.
	Sometimes the developer will not have the time or will simply forget to get back to that tag.
	This rule is meant to track those tags and to ensure that they do not go unnoticed.
	Noncompliant Code Example
	
	void doSomething() {
	// TODO
	}
	
	See
	
	MITRE, CWE-546 - Suspicious Comment

文件名称	违规行
MainActivity.java	342

规则	Resources should be closed

规则描述

Connections, streams, files, and other classes that implement the Closeable interface or its super-interface,
AutoCloseable , needs to be closed after use. Further, that close call must be made in a finally block otherwise
an exception could keep the call from being made. Preferably, when class implements AutoCloseable , resource should be created using
"try-with-resources" pattern and will be closed automatically.
Failure to properly close resources will result in a resource leak which could bring first the application and then perhaps the box the application
is on to their knees.
Noncompliant Code Example

```
private void readTheFile() throws IOException {
  Path path = Paths.get(this.fileName);
  BufferedReader reader = Files.newBufferedReader(path,
this.charset);
  // ...
  reader.close();  // Noncompliant
  // ...
  Files.lines("input.txt").forEach(System.out::println); //
Noncompliant: The stream needs to be closed
}

private void doSomething() {
  OutputStream stream = null;
  try {
    for (String property : propertyList) {
      stream = new FileOutputStream("myfile.txt");  // Noncompliant
      // ...
    }
  } catch (Exception e) {
    // ...
  } finally {
    stream.close();  // Multiple streams were opened. Only the last is
closed.
  }
}
```

Compliant Solution

```
private void readTheFile(String fileName) throws IOException {
    Path path = Paths.get(fileName);
    try (BufferedReader reader = Files.newBufferedReader(path,
StandardCharsets.UTF_8)) {
      reader.readLine();
      // ...
    }
    // ..
    try (Stream<String> input = Files.lines("input.txt"))  {
      input.forEach(System.out::println);
    }
}

private void doSomething() {
  OutputStream stream = null;
  try {
    stream = new FileOutputStream("myfile.txt");
    for (String property : propertyList) {
      // ...
```

```
    }
  } catch (Exception e) {
    // ...
  } finally {
    stream.close();
  }
}
```

Exceptions
Instances of the following classes are ignored by this rule because close has no effect:

- java.io.ByteArrayOutputStream
- java.io.ByteArrayInputStream
- java.io.CharArrayReader
- java.io.CharArrayWriter
- java.io.StringReader
- java.io.StringWriter

Java 7 introduced the try-with-resources statement, which implicitly closes Closeables . All resources opened in a try-with-resources
statement are ignored by this rule.

```
try (BufferedReader br = new BufferedReader(new
FileReader(fileName))) {
  //...
}
catch ( ... ) {
  //...
}
```

See

- MITRE, CWE-459 - Incomplete Cleanup
- MITRE, CWE-772 - Missing Release of Resource after Effective Lifetime
- CERT, FIO04-J. - Release resources when they are no longer needed
- CERT, FIO42-C. - Close files when they are no longer needed
- Try With Resources

文件名称	违规行
MyService.java	351

规则	Strings and Boxed types should be compared using "equals()"

规则描述	It's almost always a mistake to compare two instances of java.lang.String or boxed types like java.lang.Integer using reference equality == or != , because it is not comparing actual value but locations in memory.
	Noncompliant Code Example
	
	String firstName = getFirstName(); // String overrides equals
	String lastName = getLastName();
	
	if (firstName == lastName) { ... }; // Non-compliant; false even if the strings have the same value
	
	Compliant Solution
	
	String firstName = getFirstName();
	String lastName = getLastName();
	
	if (firstName != null && firstName.equals(lastName)) { ... };
	
	See
	
	MITRE, CWE-595 - Comparison of Object References Instead of Object Contents
	MITRE, CWE-597 - Use of Wrong Operator in String Comparison
	CERT, EXP03-J. - Do not use the equality operators when comparing values of boxed primitives
	CERT, EXP50-J. - Do not confuse abstract object equality with reference equality

文件名称	违规行
MyService.java	436

规则	Jump statements should not be redundant

规则描述	Jump statements such as return and continue let you change the default flow of program execution, but jump statements that direct the control flow to the original direction are just a waste of keystrokes.
	Noncompliant Code Example
	
	public void foo() {
	while (condition1) {
	if (condition2) {
	continue; // Noncompliant
	} else {
	doTheThing();
	}
	}
	return; // Noncompliant; this is a void method
	}
	
	Compliant Solution
	
	public void foo() {
	while (condition1) {
	if (!condition2) {
	doTheThing();
	}
	}
	}

文件名称	违规行
MyService.java	135

规则	Nested blocks of code should not be left empty
规则描述	Most of the time a block of code is empty when a piece of code is really missing. So such empty block must be either filled or removed. Noncompliant Code Example for (int i = 0; i < 42; i++){} // Empty on purpose or missing piece of code ? Exceptions When a block contains a comment, this block is not considered to be empty unless it is a synchronized block. synchronized blocks are still considered empty even with comments because they can still affect program flow.

文件名称	违规行
MainActivity.java	331

规则	Local variables should not shadow class fields

规则描述	Overriding or shadowing a variable declared in an outer scope can strongly impact the readability, and therefore the maintainability, of a piece of code. Further, it could lead maintainers to introduce bugs because they think they're using one variable but are really using another.
	Noncompliant Code Example
	
	class Foo {
	public int myField;
	
	public void doSomething() {
	int myField = 0;
	...
	}
	}
	
	See
	
	CERT, DCL01-C. - Do not reuse variable names in subscopes
	CERT, DCL51-J. - Do not shadow or obscure identifiers in subscopes

文件名称	违规行
MyService.java	322

1.4. 质量配置

质量配置	java:Sonar way Bug:140 漏洞:27 坏味道:256	
规则	类型	违规级别
Methods should not call same-class methods with incompatible "@Transactional" values	Bug	阻断
Methods "wait(...)", "notify()" and "notifyAll()" should not be called on Thread instances	Bug	阻断
Files opened in append mode should not be used with ObjectOutputStream	Bug	阻断
"PreparedStatement" and "ResultSet" methods should be called with valid indices	Bug	阻断
"wait(...)" should be used instead of "Thread.sleep(...)" when a lock is held	Bug	阻断
Printf-style format strings should not lead to unexpected behavior at runtime	Bug	阻断
"@SpringBootApplication" and "@ComponentScan" should not be used in the default package	Bug	阻断
"@Controller" classes that use "@SessionAttributes" must call "setComplete" on their "SessionStatus" objects	Bug	阻断
Loops should not be infinite	Bug	阻断
"wait" should not be called when multiple locks are held	Bug	阻断
Double-checked locking should not be used	Bug	阻断

Resources should be closed	Bug	阻断
Regular expressions should be syntactically valid	Bug	严重
Locks should be released	Bug	严重
Jump statements should not occur in "finally" blocks	Bug	严重
"Random" objects should be reused	Bug	严重
"super.finalize()" should be called at the end of "Object.finalize()" implementations	Bug	严重
Dependencies should not have "system" scope	Bug	严重
Assertions comparing incompatible types should not be made	Bug	严重
Assertion methods should not be used within the try block of a try-catch catching an Error	Bug	严重
The signature of "finalize()" should match that of "Object.finalize()"	Bug	严重
Only one method invocation is expected when testing checked exceptions	Bug	严重
"runFinalizersOnExit" should not be called	Bug	严重
Regex boundaries should not be used in a way that can never be matched	Bug	严重
"ScheduledThreadPoolExecutor" should not have 0 core threads	Bug	严重
Regex patterns following a possessive quantifier should not always fail	Bug	严重
Hibernate should not update database schemas	Bug	严重
Zero should not be a possible denominator	Bug	严重
Back references in regular expressions should only refer to capturing groups that are matched before the reference	Bug	严重
Regex lookahead assertions should not be contradictory	Bug	严重
JUnit5 inner test classes should be annotated with @Nested	Bug	严重
Map "computeIfAbsent()" and "computeIfPresent()" should not be used to add "null" values.	Bug	严重
Getters and setters should access the expected fields	Bug	严重
"toString()" and "clone()" methods should not return null	Bug	主要
Servlets should not have mutable instance fields	Bug	主要
Value-based classes should not be used for locking	Bug	主要
Regex alternatives should not be redundant	Bug	主要
Alternatives in regular expressions should be grouped when used with anchors	Bug	主要
Overrides should match their parent class methods in synchronization	Bug	主要
Conditionally executed code should be reachable	Bug	主要
"DefaultMessageListenerContainer" instances should not drop messages during restarts	Bug	主要

Reflection should not be used to check non-runtime annotations	Bug	主要
"SingleConnectionFactory" instances should be set to "reconnectOnException"	Bug	主要
"hashCode" and "toString" should not be called on array instances	Bug	主要
Collections should not be passed as arguments to their own methods	Bug	主要
Case insensitive Unicode regular expressions should enable the “UNICODE_CASE” flag	Bug	主要
"BigDecimal(double)" should not be used	Bug	主要
Assertions should not compare an object to itself	Bug	主要
Non-public methods should not be "@Transactional"	Bug	主要
Invalid "Date" values should not be used	Bug	主要
Unicode Grapheme Clusters should be avoided inside regex character classes	Bug	主要
Non-serializable classes should not be written	Bug	主要
Blocks should be synchronized on "private final" fields	Bug	主要
Optional value should only be accessed after calling isPresent()	Bug	主要
AssertJ configuration should be applied	Bug	主要
"notifyAll" should be used	Bug	主要
Return values from functions without side effects should not be ignored	Bug	主要
".equals()" should not be used to test the values of "Atomic" classes	Bug	主要
AssertJ methods setting the assertion context should come before an assertion	Bug	主要
The Object.finalize() method should not be called	Bug	主要
Non-serializable objects should not be stored in "HttpSession" objects	Bug	主要
Assertions should not be used in production code	Bug	主要
Tests method should not be annotated with competing annotations	Bug	主要
InputSteam.read() implementation should not return a signed byte	Bug	主要
"InterruptedException" should not be ignored	Bug	主要
Silly equality checks should not be made	Bug	主要
Dissimilar primitive wrappers should not be used with the ternary operator without explicit casting	Bug	主要
"wait", "notify" and "notifyAll" should only be called when a lock is obviously held on an object	Bug	主要
"Double.longBitsToDouble" should not be used for "int"	Bug	主要
Regular expressions should not overflow the stack	Bug	主要
Values should not be uselessly incremented	Bug	主要
Silly String operations should not be made	Bug	主要

Null pointers should not be dereferenced	Bug	主要
Expressions used in "assert" should not produce side effects	Bug	主要
Classes extending java.lang.Thread should override the "run" method	Bug	主要
Loop conditions should be true at least once	Bug	主要
A "for" loop update clause should move the counter in the right direction	Bug	主要
Intermediate Stream methods should not be left unused	Bug	主要
Consumed Stream pipelines should not be reused	Bug	主要
Variables should not be self-assigned	Bug	主要
Inappropriate regular expressions should not be used	Bug	主要
"=+" should not be used instead of "+="	Bug	主要
Loops with at most one iteration should be refactored	Bug	主要
Classes should not be compared by name	Bug	主要
Identical expressions should not be used on both sides of a binary operator	Bug	主要
JUnit5 test classes and methods should not be silently ignored	Bug	主要
"Thread.run()" should not be called directly	Bug	主要
"null" should not be used with "Optional"	Bug	主要
"read" and "readLine" return values should be used	Bug	主要
Strings and Boxed types should be compared using "equals()"	Bug	主要
Methods should not be named "tostring", "hashcode" or "equal"	Bug	主要
Non-thread-safe fields should not be static	Bug	主要
Getters and setters should be synchronized in pairs	Bug	主要
Unary prefix operators should not be repeated	Bug	主要
DateTimeFormatters should not use mismatched year and week numbers	Bug	主要
"StringBuilder" and "StringBuffer" should not be instantiated with a character	Bug	主要
"equals" method overrides should accept "Object" parameters	Bug	主要
Exceptions should not be created without being thrown	Bug	主要
Week Year ("YYYY") should not be used for date formatting	Bug	主要
Collection sizes and array length comparisons should make sense	Bug	主要
"ThreadLocal" variables should be cleaned up when no longer used	Bug	主要
Related "if/else if" statements should not have the same condition	Bug	主要

Synchronization should not be done on instances of value-based classes	Bug	主要
All branches in a conditional structure should not have exactly the same implementation	Bug	主要
The regex escape sequence \cX should only be used with characters in the @-_ range	Bug	主要
"Iterator.hasNext()" should not call "Iterator.next()"	Bug	主要
"String" calls should not go beyond their bounds	Bug	主要
Raw byte values should not be used in bitwise operations in combination with shifts	Bug	主要
Custom serialization method signatures should meet requirements	Bug	主要
"Externalizable" classes should have no-arguments constructors	Bug	主要
"iterator" should not return "this"	Bug	主要
Child class methods named for parent class methods should be overrides	Bug	主要
Inappropriate "Collection" calls should not be made	Bug	主要
"compareTo" should not be overloaded	Bug	主要
AssertJ assertions with "Consumer" arguments should contain assertion inside consumers	Bug	主要
"volatile" variables should not be used with compound operators	Bug	主要
Map values should not be replaced unconditionally	Bug	主要
"getClass" should not be used for synchronization	Bug	主要
Assignment of lazy-initialized members should be the last step with double-checked locking	Bug	主要
Min and max used in combination should not always return the same value	Bug	主要
"compareTo" results should not be checked for specific values	Bug	次要
Regex repetition pattern's body should not match the empty String	Bug	次要
AssertJ assertions "allMatch" and "doesNotContains" should also test for emptiness	Bug	次要
Double Brace Initialization should not be used	Bug	次要
Boxing and unboxing should not be immediately reversed	Bug	次要
"Iterator.next()" methods should throw "NoSuchElementException"	Bug	次要
"@NonNull" values should not be set to null	Bug	次要
Neither "Math.abs" nor negation should be used on numbers that could be "MIN_VALUE"	Bug	次要
The value returned from a stream read should be checked	Bug	次要
Method parameters, caught exceptions and foreach variables' initial values should not be ignored	Bug	次要

"equals(Object obj)" and "hashCode()" should be overridden in pairs	Bug	次要
"Serializable" inner classes of non-serializable classes should be "static"	Bug	次要
Math operands should be cast before assignment	Bug	次要
Ints and longs should not be shifted by zero or more than their number of bits-1	Bug	次要
"compareTo" should not return "Integer.MIN_VALUE"	Bug	次要
The non-serializable super class of a "Serializable" class should have a no-argument constructor	Bug	次要
"toArray" should be passed an array of the proper type	Bug	次要
Non-primitive fields should not be "volatile"	Bug	次要
"equals(Object obj)" should test argument type	Bug	次要
Return values should not be ignored when they contain the operation status code	Bug	次要
A secure password should be used when connecting to a database	漏洞	阻断
XML parsers should not be vulnerable to XXE attacks	漏洞	阻断
Default EJB interceptors should be declared in "ejb-jar.xml"	漏洞	阻断
Struts validation forms should have unique names	漏洞	阻断
Cipher Block Chaining IV's should be unpredictable	漏洞	严重
Defined filters should be used	漏洞	严重
Persistent entities should not be used as arguments of "@RequestMapping" methods	漏洞	严重
JWT should be signed and verified with strong cipher algorithms	漏洞	严重
Cipher algorithms should be robust	漏洞	严重
Encryption algorithms should be used with secure mode and padding scheme	漏洞	严重
A new session should be created during user authentication	漏洞	严重
Weak SSL/TLS protocols should not be used	漏洞	严重
Cryptographic keys should be robust	漏洞	严重
"HttpServletRequest.getRequestedSessionId()" should not be used	漏洞	严重
LDAP connections should be authenticated	漏洞	严重
Server hostnames should be verified during SSL/TLS connections	漏洞	严重
"HttpSecurity" URL patterns should be correctly ordered	漏洞	严重
Basic authentication should not be used	漏洞	严重
Server certificates should be verified during SSL/TLS connections	漏洞	严重
Passwords should not be stored in plain-text or with a fast hashing algorithm	漏洞	严重

"SecureRandom" seeds should not be predictable	漏洞	严重
Insecure temporary file creation methods should not be used	漏洞	严重
Hashes should include an unpredictable salt	漏洞	严重
Authorizations should be based on strong decisions	漏洞	主要
OpenSAML2 should be configured to prevent authentication bypass	漏洞	主要
"ActiveMQConnectionFactory" should not be vulnerable to malicious code deserialization	漏洞	次要
Exceptions should not be thrown from servlet methods	漏洞	次要
Tests should include assertions	坏味道	阻断
Child class fields should not shadow parent class fields	坏味道	阻断
Assertions should be complete	坏味道	阻断
"clone" should not be overridden	坏味道	阻断
"switch" statements should not contain non-case labels	坏味道	阻断
Methods returns should not be invariant	坏味道	阻断
Silly bit operations should not be performed	坏味道	阻断
Switch cases should end with an unconditional "break" statement	坏味道	阻断
Methods and field names should not be the same or differ only by capitalization	坏味道	阻断
JUnit test cases should call super methods	坏味道	阻断
TestCases should contain tests	坏味道	阻断
"ThreadGroup" should not be used	坏味道	阻断
Future keywords should not be used as names	坏味道	阻断
Short-circuit logic should be used in boolean contexts	坏味道	阻断
"default" clauses should be last	坏味道	严重
Whitespace and control characters in literals should be explicit	坏味道	严重
IllegalMonitorStateException should not be caught	坏味道	严重
Cognitive Complexity of methods should not be too high	坏味道	严重
The Object.finalize() method should not be overridden	坏味道	严重
Package declaration should match source file directory	坏味道	严重
Null should not be returned from a "Boolean" method	坏味道	严重
String offset-based methods should be preferred for finding substrings from offsets	坏味道	严重
Instance methods should not write to "static" fields	坏味道	严重
"indexOf" checks should not be for positive numbers	坏味道	严重

Factory method injection should be used in "@Configuration" classes	坏味道	严重
Empty lines should not be tested with regex MULTILINE flag	坏味道	严重
Mocking all non-private methods of a class should be avoided	坏味道	严重
"Object.finalize()" should remain protected (versus public) when overriding	坏味道	严重
"Cloneables" should implement "clone"	坏味道	严重
"Object.wait(...)" and "Condition.await(...)" should be called inside a "while" loop	坏味道	严重
Methods should not be empty	坏味道	严重
"equals" method parameters should not be marked "@Nonnull"	坏味道	严重
Classes should not access their own subclasses during initialization	坏味道	严重
Exceptions should not be thrown in finally blocks	坏味道	严重
Method overrides should not change contracts	坏味道	严重
"for" loop increment clauses should modify the loops' counters	坏味道	严重
Constants should not be defined in interfaces	坏味道	严重
Generic wildcard types should not be used in return types	坏味道	严重
Execution of the Garbage Collector should be triggered only by the JVM	坏味道	严重
Derived exceptions should not hide their parents' catch blocks	坏味道	严重
Methods setUp() and tearDown() should be correctly annotated starting with JUnit4	坏味道	严重
Conditionals should start on new lines	坏味道	严重
A conditionally executed single line should be denoted by indentation	坏味道	严重
Class members annotated with "@VisibleForTesting" should not be accessed from production code	坏味道	严重
Fields in a "Serializable" class should either be transient or serializable	坏味道	严重
"switch" statements should have "default" clauses	坏味道	严重
JUnit assertions should not be used in "run" methods	坏味道	严重
"readResolve" methods should be inheritable	坏味道	严重
Constant names should comply with a naming convention	坏味道	严重
"static" base class members should not be accessed via derived types	坏味道	严重
String literals should not be duplicated	坏味道	严重
Class names should not shadow interfaces or superclasses	坏味道	严重
"String#replace" should be preferred to "String#replaceAll"	坏味道	严重
Try-with-resources should be used	坏味道	严重

Source files should not have any duplicated blocks	坏味道	主要
Regexes containing characters subject to normalization should use the CANON_EQ flag	坏味道	主要
Boolean expressions should not be gratuitous	坏味道	主要
Track uses of "FIXME" tags	坏味道	主要
Similar tests should be grouped in a single Parameterized test	坏味道	主要
Tests should be stable	坏味道	主要
"@Deprecated" code marked for removal should never be used	坏味道	主要
Parameters should be passed in the correct order	坏味道	主要
Unused "private" methods should be removed	坏味道	主要
"ResultSet.isLast()" should not be used	坏味道	主要
"URL.hashCode" and "URL.equals" should be avoided	坏味道	主要
Names of regular expressions named groups should be used	坏味道	主要
Try-catch blocks should not be nested	坏味道	主要
Character classes in regular expressions should not contain the same character twice	坏味道	主要
Synchronized classes Vector, Hashtable, Stack and StringBuffer should not be used	坏味道	主要
Redundant pairs of parentheses should be removed	坏味道	主要
Classes with only "static" methods should not be instantiated	坏味道	主要
"Lock" objects should not be "synchronized"	坏味道	主要
Multiline blocks should be enclosed in curly braces	坏味道	主要
Labels should not be used	坏味道	主要
"static" members should be accessed statically	坏味道	主要
Utility classes should not have public constructors	坏味道	主要
Assertion arguments should be passed in the correct order	坏味道	主要
Local variables should not shadow class fields	坏味道	主要
Unused type parameters should be removed	坏味道	主要
AssertJ "assertThatThrownBy" should not be used alone	坏味道	主要
"switch" statements should not have too many "case" clauses	坏味道	主要
Regular expressions should not be too complicated	坏味道	主要
Deprecated elements should have both the annotation and the Javadoc tag	坏味道	主要
Assignments should not be made from within sub-expressions	坏味道	主要
Test methods should not contain too many assertions	坏味道	主要
Ternary operators should not be nested	坏味道	主要

'List.remove()' should not be used in ascending 'for' loops	坏味道	主要
Exception testing via JUnit ExpectedException rule should not be mixed with other assertions	坏味道	主要
Inner class calls to super class methods should be unambiguous	坏味道	主要
Only one method invocation is expected when testing runtime exceptions	坏味道	主要
Nullness of parameters should be guaranteed	坏味道	主要
Unused method parameters should be removed	坏味道	主要
Only static class initializers should be used	坏味道	主要
Vararg method arguments should not be confusing	坏味道	主要
Unused "private" fields should be removed	坏味道	主要
Collapsible "if" statements should be merged	坏味道	主要
Unused labels should be removed	坏味道	主要
JUnit assertTrue/assertFalse should be simplified to the corresponding dedicated assertion	坏味道	主要
Whitespace for text block indent should be consistent	坏味道	主要
Throwable and Error should not be caught	坏味道	主要
Printf-style format strings should be used correctly	坏味道	主要
"Integer.toHexString" should not be used to build hexadecimal strings	坏味道	主要
Constructors of an "abstract" class should not be declared "public"	坏味道	主要
Constructors should not be used to instantiate "String", "BigInteger", "BigDecimal" and primitive-wrapper classes	坏味道	主要
Enumeration should not be implemented	坏味道	主要
Empty arrays and collections should be returned instead of null	坏味道	主要
Objects should not be created only to "getClass"	坏味道	主要
Primitives should not be boxed just for "String" conversion	坏味道	主要
Exceptions should be either logged or rethrown but not both	坏味道	主要
"@Override" should be used on overriding and implementing methods	坏味道	主要
"Preconditions" and logging arguments should not require evaluation	坏味道	主要
"entrySet()" should be iterated when both the key and value are needed	坏味道	主要
"Class.forName()" should not load JDBC 4.0+ drivers	坏味道	主要
Two branches in a conditional structure should not have exactly the same implementation	坏味道	主要
"Map.get" and value test should be replaced with single method call	坏味道	主要

"Arrays.stream" should be used for primitive arrays	坏味道	主要
"@RequestMapping" methods should be "public"	坏味道	主要
Non-constructor methods should not have the same name as the enclosing class	坏味道	主要
"readObject" should not be "synchronized"	坏味道	主要
"Threads" should not be used where "Runnables" are expected	坏味道	主要
Java features should be preferred to Guava	坏味道	主要
"Stream.peek" should be used with caution	坏味道	主要
Unused "private" classes should be removed	坏味道	主要
Raw types should not be used	坏味道	主要
A field should not duplicate the name of its containing class	坏味道	主要
Single-character alternations in regular expressions should be replaced with character classes	坏味道	主要
String multiline concatenation should be replaced with Text Blocks	坏味道	主要
Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string	坏味道	主要
Unused assignments should be removed	坏味道	主要
"DateUtils.truncate" from Apache Commons Lang library should not be used	坏味道	主要
"Thread.sleep" should not be used in tests	坏味道	主要
Sections of code should not be commented out	坏味道	主要
"for" loop stop conditions should be invariant	坏味道	主要
Anonymous inner classes containing only one method should become lambdas	坏味道	主要
JUnit4 @Ignored and JUnit5 @Disabled annotations should be used to disable tests and should provide a rationale	坏味道	主要
"Object.wait(...)" should never be called on objects that implement "java.util.concurrent.locks.Condition"	坏味道	主要
Inheritance tree of classes should not be too deep	坏味道	主要
Generic exceptions should never be thrown	坏味道	主要
Silly math should not be performed	坏味道	主要
Standard outputs should not be used directly to log anything	坏味道	主要
Methods should not have too many parameters	坏味道	主要
Nested blocks of code should not be left empty	坏味道	主要
"writeObject" should not be the only "synchronized" code in a class	坏味道	主要
Classes named like "Exception" should extend "Exception" or a subclass	坏味道	主要
Reflection should not be used to increase accessibility of classes, methods, or fields	坏味道	主要

Static fields should not be updated in constructors	坏味道	主要
Exception types should not be tested using "instanceof" in catch blocks	坏味道	主要
Classes from "sun.*" packages should not be used	坏味道	主要
"java.nio.Files#delete" should be preferred	坏味道	主要
Assignments should not be redundant	坏味道	主要
"else" statements should be clearly matched with an "if"	坏味道	主要
Operator "instanceof" should be used instead of "A.class.isInstance()"	坏味道	主要
Methods should not have identical implementations	坏味道	主要
Restricted Identifiers should not be used as Identifiers	坏味道	主要
Asserts should not be used to check the parameters of a public method	坏味道	主要
Consecutive AssertJ "assertThat" statements should be chained	坏味道	次要
"throws" declarations should not be superfluous	坏味道	次要
Character classes should be preferred over reluctant quantifiers in regular expressions	坏味道	次要
A "while" loop should be used instead of a "for" loop	坏味道	次要
"Collections.EMPTY_LIST", "EMPTY_MAP", and "EMPTY_SET" should not be used	坏味道	次要
Chained AssertJ assertions should be simplified to the corresponding dedicated assertion	坏味道	次要
Empty statements should be removed	坏味道	次要
Loggers should be named for their enclosing classes	坏味道	次要
Return of boolean expressions should not be wrapped into an "if-then-else" statement	坏味道	次要
Local variables should not be declared and then immediately returned or thrown	坏味道	次要
Boolean literals should not be redundant	坏味道	次要
Modifiers should be declared in the correct order	坏味道	次要
Deprecated "${pom}" properties should not be used	坏味道	次要
Unnecessary imports should be removed	坏味道	次要
Unused local variables should be removed	坏味道	次要
Exception testing via JUnit @Test annotation should be avoided	坏味道	次要
Catches should be combined	坏味道	次要
Mutable fields should not be "public static"	坏味道	次要
Null checks should not be used with "instanceof"	坏味道	次要
Boxed "Boolean" should be avoided in boolean expressions	坏味道	次要

Methods of "Random" that return floating point values should not be used in random integer generation	坏味道	次要
Public constants and fields initialized at declaration should be "static final" rather than merely "final"	坏味道	次要
"@CheckForNull" or "@Nullable" should not be used on primitive types	坏味道	次要
Simple string literal should be used for single line strings	坏味道	次要
Escape sequences should not be used in text blocks	坏味道	次要
Overriding methods should do more than simply call the same method in the super class	坏味道	次要
Static non-final field names should comply with a naming convention	坏味道	次要
Classes that override "clone" should be "Cloneable" and call "super.clone()"	坏味道	次要
Primitive wrappers should not be instantiated only for "toString" or "compareTo" calls	坏味道	次要
Case insensitive string comparisons should be made without intermediate upper or lower casing	坏味道	次要
Test classes should comply with a naming convention	坏味道	次要
Collection.isEmpty() should be used to test for emptiness	坏味道	次要
String.valueOf() should not be appended to a String	坏味道	次要
Exception classes should be immutable	坏味道	次要
Parsing should be used to convert "Strings" to primitives	坏味道	次要
Multiple variables should not be declared on the same line	坏味道	次要
"read(byte[],int,int)" should be overridden	坏味道	次要
"switch" statements should have at least 3 "case" clauses	坏味道	次要
"@Deprecated" code should not be used	坏味道	次要
Strings should not be concatenated using '+' in a loop	坏味道	次要
Maps with keys that are enum values should be replaced with EnumMap	坏味道	次要
"catch" clauses should do more than rethrow	坏味道	次要
Nested "enum"s should not be declared static	坏味道	次要
"equals(Object obj)" should be overridden along with the "compareTo(T obj)" method	坏味道	次要
Private fields only used as local variables in methods should become local variables	坏味道	次要
Arrays should not be created for varargs parameters	坏味道	次要
Class variable fields should not have public accessibility	坏味道	次要

Methods should not return constants	坏味道	次要
The default unnamed package should not be used	坏味道	次要
Type parameters should not shadow other type parameters	坏味道	次要
Declarations should use Java collection interfaces such as "List" rather than specific implementation classes such as "LinkedList"	坏味道	次要
"public static" fields should be constant	坏味道	次要
An iteration on a Collection should be performed on the type handled by the Collection	坏味道	次要
"StandardCharsets" constants should be preferred	坏味道	次要
Jump statements should not be redundant	坏味道	次要
Boolean checks should not be inverted	坏味道	次要
"close()" calls should not be redundant	坏味道	次要
"indexOf" checks should use a start position	坏味道	次要
Redundant casts should not be used	坏味道	次要
"ThreadLocal.withInitial" should be preferred	坏味道	次要
Abstract classes without fields should be converted to interfaces	坏味道	次要
"toString()" should never be called on a String object	坏味道	次要
Parentheses should be removed from a single lambda input parameter when its type is inferred	坏味道	次要
Lambdas should be replaced with method references	坏味道	次要
Call to Mockito method "verify", "when" or "given" should be simplified	坏味道	次要
JUnit rules should be used	坏味道	次要
Annotation repetitions should not be wrapped	坏味道	次要
Lambdas containing only one statement should not nest this statement in a block	坏味道	次要
Loops should not contain more than a single "break" or "continue" statement	坏味道	次要
Abstract methods should not be redundant	坏味道	次要
"private" methods called only by inner classes should be moved to those classes	坏味道	次要
Fields in non-serializable classes should not be "transient"	坏味道	次要
Composed "@RequestMapping" variants should be preferred	坏味道	次要
Package names should comply with a naming convention	坏味道	次要
Interface names should comply with a naming convention	坏味道	次要
Field names should comply with a naming convention	坏味道	次要
Local variable and method parameter names should comply with a naming convention	坏味道	次要

Type parameter names should comply with a naming convention	坏味道	次要
"write(byte[],int,int)" should be overridden	坏味道	次要
Nested code blocks should not be used	坏味道	次要
Array designators "[]" should be on the type, not the variable	坏味道	次要
URIs should not be hardcoded	坏味道	次要
"finalize" should not set fields to "null"	坏味道	次要
Arrays should not be copied using loops	坏味道	次要
Array designators "[]" should be located after the type in method signatures	坏味道	次要
Subclasses that add fields should override "equals"	坏味道	次要
Class names should comply with a naming convention	坏味道	次要
Method names should comply with a naming convention	坏味道	次要
The diamond operator ("<>") should be used	坏味道	次要
Switch arrow labels should not use redundant keywords	坏味道	次要
Text blocks should not be used in complex expressions	坏味道	次要
Functional Interfaces should be as specialised as possible	坏味道	次要
"enum" fields should not be publicly mutable	坏味道	次要
"Stream" call chains should be simplified when possible	坏味道	次要
Packages containing only "package-info.java" should be removed	坏味道	次要
Classes should not be empty	坏味道	次要
Track uses of "TODO" tags	坏味道	提示
Deprecated code should be removed	坏味道	提示
JUnit5 test classes and methods should have default package visibility	坏味道	提示
Comma-separated labels should be used in Switch with colon case	坏味道	提示
Local-Variable Type Inference should be used	坏味道	提示