mjkf
0.0.1-SNAPSHOT

Demo project for Spring Boot

java:Sonar way

xml:Sonar way

2022-02-12

目录

1. mjkf

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mjkf，生成时间为2022-02-12，使用的质量配置为 java:Sonar way xml:Sonar way，共计 328条规则。

1.1. 概述

编码问题

Bug 12	可靠性修复工作 1h0min
漏洞 8	安全修复工作 1h20min
坏味道 170	技术债务 20h48min

190 问题	开启问题	190
	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	0
	已删除的问题	0
	阻断	3
	严重	21
	主要	78
	次要	88
	提示	0

静态分析

项目规模

2547 代码行数	行数	3466
	方法	385
	类	58
	文件	59
	目录	8
	重复行(%)	8.8

复杂度

409 复杂度	文件	7.1

注释(%)

8.6 注释(%)	注释行数	239

1.2. 问题分析

违反最多的规则TOP10	
Sections of code should not be commented out	34
Composed "@RequestMapping" variants should be preferred	27
Standard outputs should not be used directly to log anything	19
String literals should not be duplicated	13
Method names should comply with a naming convention	13
Local variables should not be declared and then immediately returned or thrown	10
Package names should comply with a naming convention	10
Classes should not be compared by name	9
Throwable.printStackTrace(...) should not be called	8
Cognitive Complexity of methods should not be too high	6

违规最多的文件TOP5	
GetDocument.java	36
AddDataImpl.java	22
UpdateTable.java	21
Rank.java	14
Keyword.java	11

复杂度最高的文件TOP5	
Rank.java	42
GetDocument.java	39
Paper.java	28
PaperVO.java	28
UpdateTable.java	25

重复行最多的文件TOP5	
Paper.java	140
PaperVO.java	140
AuthorPic.java	13
AuthorPicVO.java	13

1.3. 问题详情

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. See MISRA C:2004, 2.4 - Sections of code should not be "commented out". MISRA C++:2008, 2-7-2 - Sections of code shall not be "commented out" using C-style comments. MISRA C++:2008, 2-7-3 - Sections of code should not be "commented out" using C++ comments. MISRA C:2012, Dir. 4.4 - Sections of code should not be "commented out"

文件名称	违规行
AuthorPicController.java	25, 50
ConferPicController.java	34, 47, 49
RankController.java	24, 38
PortrayMapper.java	19

RankMapper.java	15, 20, 30, 36, 39, 54, 60, 63, 69
StatisticsMapper.java	20
RankInter.java	12
AddDataImpl.java	89
GetDocument.java	64, 223, 297
Portray.java	41
Rank.java	49, 54, 60, 97, 102, 108, 114, 178, 230
Statistics.java	32

规则	Composed "@RequestMapping" variants should be preferred

规则描述	Spring framework 4.3 introduced variants of the @RequestMapping annotation to better represent the semantics of the annotated methods.
	The use of @GetMapping , @PostMapping , @PutMapping , @PatchMapping and @DeleteMapping should be preferred to the use of the raw @RequestMapping(method = RequestMethod.XYZ) .
	Noncompliant Code Example
	
	@RequestMapping(path = "/greeting", method = RequestMethod.GET) // Noncompliant
	public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {
	...
	}
	
	Compliant Solution
	
	@GetMapping(path = "/greeting") // Compliant
	public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {
	...
	}

文件名称	违规行
AddDataController.java	21
AuthorPicController.java	20, 29, 37, 45
ConferPicController.java	30, 44, 53
LoginController.java	24
PaperController.java	33, 63, 70, 97, 103, 110
PortrayController.java	26, 33, 40, 47, 54
RankController.java	21, 45, 67, 76
StatisticsController.java	21, 27, 33

规则	Standard outputs should not be used directly to log anything

规则描述	When logging a message there are several important requirements which must be fulfilled:
	
	The user must be able to easily retrieve the logs
	The format of all logged message must be uniform to allow the user to easily read the log
	Logged data must actually be recorded
	Sensitive data must only be logged securely
	
	If a program directly writes to the standard outputs, there is absolutely no way to comply with those requirements. That's why defining and using a
	dedicated logger is highly recommended.
	Noncompliant Code Example
	
	System.out.println("My Message"); // Noncompliant
	
	Compliant Solution
	
	logger.log("My Message");
	
	See
	
	CERT, ERR02-J. - Prevent exceptions while logging data

文件名称	违规行
AddDataController.java	24
ConferPicController.java	56, 58
AddDataImpl.java	84
GetDocument.java	227, 235, 246
UpdateTable.java	26, 38, 51, 63, 74, 86, 99, 113, 128, 147, 163, 178

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
PaperController.java	118
Keyword.java	21, 22, 23, 24, 25, 26
AddDataImpl.java	193
GetDocument.java	30, 208
UpdateTable.java	33, 45, 120

规则	String literals should not be duplicated

规则描述	Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences.
	On the other hand, constants can be referenced from many places, but only need to be updated in a single place.
	Noncompliant Code Example
	With the default threshold of 3:
	
	public void run() {
	prepare("action1"); // Noncompliant - "action1" is duplicated 3 times
	execute("action1");
	release("action1");
	}
	
	@SuppressWarning("all") // Compliant - annotations are excluded
	private void method1() { /* ... */ }
	@SuppressWarning("all")
	private void method2() { /* ... */ }
	
	public String method3(String a) {
	System.out.println("'" + a + "'"); // Compliant - literal "'" has less than 5 characters and is excluded
	return ""; // Compliant - literal "" has less than 5 characters and is excluded
	}
	
	Compliant Solution
	
	private static final String ACTION_1 = "action1"; // Compliant
	
	public void run() {
	prepare(ACTION_1); // Compliant
	execute(ACTION_1);
	release(ACTION_1);
	}
	
	Exceptions
	To prevent generating some false-positives, literals having less than 5 characters are excluded.

文件名称	违规行
AuthorPicController.java	32
RankController.java	26, 27
GetDocument.java	35, 37, 73, 86, 99, 116, 133, 165, 178, 190

规则	Local variables should not be declared and then immediately returned or thrown

规则描述	Declaring a variable only to immediately return or throw it is a bad practice.
	Some developers argue that the practice improves code readability, because it enables them to explicitly name what is being returned. However, this
	variable is an internal implementation detail that is not exposed to the callers of the method. The method name should be sufficient for callers to
	know exactly what will be returned.
	Noncompliant Code Example
	
	public long computeDurationInMilliseconds() {
	long duration = (((hours * 60) + minutes) * 60 + seconds) * 1000 ;
	return duration;
	}
	
	public void doSomething() {
	RuntimeException myException = new RuntimeException();
	throw myException;
	}
	
	Compliant Solution
	
	public long computeDurationInMilliseconds() {
	return (((hours * 60) + minutes) * 60 + seconds) * 1000 ;
	}
	
	public void doSomething() {
	throw new RuntimeException();
	}

文件名称	违规行
AddDataController.java	30
ConferPicController.java	33, 48
Portray.java	28, 34, 40, 63
Search.java	89, 94, 99

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

AddDataImpl.java	1
AuthorPicImpl.java	1
GetConferPic.java	1
GetDocument.java	1
Login.java	1
Portray.java	1
Rank.java	1
Search.java	1
Statistics.java	1
UpdateTable.java	1

规则	Classes should not be compared by name

规则描述	

There is no requirement that class names be unique, only that they be unique within a package. Therefore trying to determine an object's type based
on its class name is an exercise fraught with danger. One of those dangers is that a malicious user will send objects of the same name as the trusted
class and thereby gain trusted access.

Instead, the instanceof operator or the Class.isAssignableFrom() method should be used to check the object's underlying
type.

Noncompliant Code Example

```
package computer;
class Pear extends Laptop { ... }

package food;
class Pear extends Fruit { ... }

class Store {

  public boolean hasSellByDate(Object item) {
    if ("Pear".equals(item.getClass().getSimpleName())) {  //
Noncompliant
      return true;  // Results in throwing away week-old computers
    }
    return false;
  }

  public boolean isList(Class<T> valueClass) {
    if (List.class.getName().equals(valueClass.getName())) {  //
Noncompliant
      return true;
    }
    return false;
  }
}
```

Compliant Solution

```
class Store {

  public boolean hasSellByDate(Object item) {
    if (item instanceof food.Pear) {
      return true;
    }
    return false;
  }

  public boolean isList(Class<T> valueClass) {
    if (valueClass.isAssignableFrom(List.class)) {
      return true;
    }
    return false;
  }
}
```

See

MITRE, CWE-486 - Comparison of Classes by Name
CERT, OBJ09-J. - Compare classes and not class names

文件名称	违规行
GetDocument.java	37, 75, 88, 101, 118, 135, 167, 180, 192

规则	Throwable.printStackTrace(...) should not be called

规则描述	Throwable.printStackTrace(...) prints a Throwable and its stack trace to some stream. By default that stream System.Err , which could inadvertently expose sensitive information.
	Loggers should be used instead to print Throwable s, as they have many advantages:
	
	Users are able to easily retrieve the logs.
	The format of log messages is uniform and allow users to browse the logs easily.
	
	This rule raises an issue when printStackTrace is used without arguments, i.e. when the stack trace is printed to the default stream.
	Noncompliant Code Example
	
	try {
	/* ... */
	} catch(Exception e) {
	e.printStackTrace(); // Noncompliant
	}
	
	Compliant Solution
	
	try {
	/* ... */
	} catch(Exception e) {
	LOGGER.log("context", e);
	}
	
	See
	
	MITRE, CWE-489 - Leftover Debug Code
	OWASP Top 10 2017 Category A3 - Sensitive Data Exposure

文件名称	违规行
AddDataImpl.java	145, 225, 227, 233
GetDocument.java	304
UpdateTable.java	201, 207, 214

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
AddDataImpl.java	77
GetDocument.java	30
Rank.java	21, 68, 148, 199

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
AddDataImpl.java	79
GetDocument.java	51, 52, 148, 292

规则	Source files should not have any duplicated blocks
规则描述	An issue is created on a file as soon as there is at least one block of duplicated code on this file

文件名称	违规行
AuthorPic.java	N/A
Paper.java	N/A
AuthorPicVO.java	N/A
PaperVO.java	N/A

规则	Parsing should be used to convert "Strings" to primitives

规则描述	Rather than creating a boxed primitive from a String to extract the primitive value, use the relevant parse method instead. It will be clearer and more efficient.
	Noncompliant Code Example
	
	String myNum = "12.2";
	
	float f = (new Float(myNum)).floatValue(); // Noncompliant; creates & discards a Float
	
	Compliant Solution
	
	String myNum = "12.2";
	
	float f = Float.parseFloat(myNum);

文件名称	违规行
Search.java	32, 33, 47, 48

规则	Printf-style format strings should be used correctly

规则描述	Because printf -style format strings are interpreted at runtime, rather than validated by the compiler, they can contain errors that result in the wrong strings being created. This rule statically validates the correlation of printf -style format strings to their arguments when calling the format(...) methods of java.util.Formatter , java.lang.String , java.io.PrintStream , MessageFormat , and java.io.PrintWriter classes and the printf(...) methods of java.io.PrintStream or java.io.PrintWriter classes.
	Noncompliant Code Example
	
	String.format("First {0} and then {1}", "foo", "bar");
	//Noncompliant. Looks like there is a confusion with the use of {{java.text.MessageFormat}}, parameters "foo" and "bar" will be simply ignored here
	String.format("Display %3$d and then %d", 1, 2, 3);
	//Noncompliant; the second argument '2' is unused
	String.format("Too many arguments %d and %d", 1, 2, 3);
	//Noncompliant; the third argument '3' is unused
	String.format("First Line\n"); //Noncompliant; %n should be used in place of \n to produce the platform-specific line separator
	String.format("Is myObject null ? %b", myObject);
	//Noncompliant; when a non-boolean argument is formatted with %b, it prints true for any nonnull value, and false for null. Even if intended, this is misleading. It's better to directly inject the boolean value (myObject == null in this case)
	String.format("value is " + value); // Noncompliant
	String s = String.format("string without arguments"); // Noncompliant
	
	MessageFormat.format("Result '{0}'.", value); // Noncompliant; String contains no format specifiers. (quote are discarding format specifiers)
	MessageFormat.format("Result {0}.", value, value); // Noncompliant; 2nd argument is not used
	MessageFormat.format("Result {0}.", myObject.toString()); // Noncompliant; no need to call toString() on objects
	
	java.util.Logger logger;
	logger.log(java.util.logging.Level.SEVERE, "Result {0}.", myObject.toString()); // Noncompliant; no need to call toString() on objects
	logger.log(java.util.logging.Level.SEVERE, "Result.", new Exception()); // compliant, parameter is an exception
	logger.log(java.util.logging.Level.SEVERE, "Result '{0}'", 14); // Noncompliant {{String contains no format specifiers.}}
	
	org.slf4j.Logger slf4jLog;
	org.slf4j.Marker marker;
	
	slf4jLog.debug(marker, "message {}");
	slf4jLog.debug(marker, "message ", 1); // Noncompliant {{String contains no format specifiers.}}
	
	Compliant Solution
	
	String.format("First %s and then %s", "foo", "bar");
	String.format("Display %2$d and then %d", 1, 3);
	String.format("Too many arguments %d %d", 1, 2);
	String.format("First Line%n");
	String.format("Is myObject null ? %b", myObject == null);

```
String.format("value is %d", value);
String s = "string without arguments";

MessageFormat.format("Result {0}.", value);
MessageFormat.format("Result '{0}' = {0}", value);
MessageFormat.format("Result {0}.", myObject);

java.util.Logger logger;
logger.log(java.util.logging.Level.SEVERE, "Result {0}.", myObject);
logger.log(java.util.logging.Level.SEVERE, "Result {0}'", 14);


org.slf4j.Logger slf4jLog;
org.slf4j.Marker marker;

slf4jLog.debug(marker, "message {}");
slf4jLog.debug(marker, "message {}", 1);
```

See

CERT, FIO47-C. - Use valid format strings

文件名称	违规行
AddDataImpl.java	163, 169
UpdateTable.java	208

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 it's 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
- 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
AddDataImpl.java	81, 200
UpdateTable.java	194

1.4. 质量配置

质量配置	java:Sonar way Bug:100 漏洞:27 坏味道:194	
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
"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
Locks should be released	Bug	严重
Dependencies should not have "system" scope	Bug	严重
The signature of "finalize()" should match that of "Object.finalize()"	Bug	严重
"runFinalizersOnExit" should not be called	Bug	严重
"ScheduledThreadPoolExecutor" should not have 0 core threads	Bug	严重
"super.finalize()" should be called at the end of "Object.finalize()" implementations	Bug	严重
Zero should not be a possible denominator	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
Conditionally executed blocks 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
"BigDecimal(double)" should not be used	Bug	主要
Jump statements should not occur in "finally" blocks	Bug	主要
Non-public methods should not be "@Transactional"	Bug	主要
Invalid "Date" values should not be used	Bug	主要
Non-serializable classes should not be written	Bug	主要

Optional value should only be accessed after calling isPresent()	Bug	主要
Blocks should be synchronized on "private final" fields	Bug	主要
"notifyAll" should be used	Bug	主要
".equals()" should not be used to test the values of "Atomic" classes	Bug	主要
Return values from functions without side effects should not be ignored	Bug	主要
Non-serializable objects should not be stored in "HttpSession" objects	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
Values should not be uselessly incremented	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
The Object.finalize() method should not be called	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
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
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
"StringBuilder" and "StringBuffer" should not be instantiated with a character	Bug	主要
Week Year ("YYYY") should not be used for date formatting	Bug	主要
"equals" method overrides should accept "Object" parameters	Bug	主要
Exception should not be created without being thrown	Bug	主要
Collection sizes and array length comparisons should make sense	Bug	主要
Synchronization should not be based on Strings or boxed primitives	Bug	主要
Related "if/else if" statements should not have the same condition	Bug	主要
All branches in a conditional structure should not have exactly the same implementation	Bug	主要
"Iterator.hasNext()" should not call "Iterator.next()"	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
Map values should not be replaced unconditionally	Bug	主要
"getClass" should not be used for synchronization	Bug	主要
"compareTo" results should not be checked for specific values	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
"equals(Object obj)" should test argument type	Bug	次要
Neither DES (Data Encryption Standard) nor DESede (3DES) should be used	漏洞	阻断
Cryptographic keys should not be too short	漏洞	阻断
LDAP deserialization should be disabled	漏洞	阻断
"HostnameVerifier.verify" should not always return true	漏洞	阻断
Credentials should not be hard-coded	漏洞	阻断
Default EJB interceptors should be declared in "ejb-jar.xml"	漏洞	阻断
Persistent entities should not be used as arguments of "@RequestMapping" methods	漏洞	严重
Defined filters should be used	漏洞	严重
Cryptographic RSA algorithms should always incorporate OAEP (Optimal Asymmetric Encryption Padding)	漏洞	严重
XML transformers should be secured	漏洞	严重
"HttpServletRequest.getRequestedSessionId()" should not be used	漏洞	严重
LDAP connections should be authenticated	漏洞	严重
AES encryption algorithm should be used with secured mode	漏洞	严重
"File.createTempFile" should not be used to create a directory	漏洞	严重
Web applications should not have a "main" method	漏洞	严重
SMTP SSL connection should check server identity	漏洞	严重
SQL binding mechanisms should be used	漏洞	严重
"SecureRandom" seeds should not be predictable	漏洞	严重
TrustManagers should not blindly accept any certificates	漏洞	主要
Weak SSL protocols should not be used	漏洞	主要
Throwable.printStackTrace(...) should not be called	漏洞	次要
Mutable fields should not be "public static"	漏洞	次要
"public static" fields should be constant	漏洞	次要
Exceptions should not be thrown from servlet methods	漏洞	次要

Class variable fields should not have public accessibility	漏洞	次要
"enum" fields should not be publicly mutable	漏洞	次要
Return values should not be ignored when they contain the operation status code	漏洞	次要
Child class fields should not shadow parent class fields	坏味道	阻断
JUnit framework methods should be declared properly	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
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
Constant names should comply with a naming convention	坏味道	严重
"default" clauses should be last	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
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
Instance methods should not write to "static" fields	坏味道	严重
String offset-based methods should be preferred for finding substrings from offsets	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
"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 parameters	坏味道	严重
Execution of the Garbage Collector should be triggered only by the JVM	坏味道	严重
The Object.finalize() method should not be overriden	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
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
String literals should not be duplicated	坏味道	严重
Class names should not shadow interfaces or superclasses	坏味道	严重
Try-with-resources should be used	坏味道	严重
Boolean expressions should not be gratuitous	坏味道	主要
Track uses of "FIXME" tags	坏味道	主要
Parameters should be passed in the correct order	坏味道	主要
"ResultSet.isLast()" should not be used	坏味道	主要
Nested blocks of code should not be left empty	坏味道	主要
"URL.hashCode" and "URL.equals" should be avoided	坏味道	主要
Try-catch blocks should not be nested	坏味道	主要
Methods should not have too many parameters	坏味道	主要
Synchronized classes Vector, Hashtable, Stack and StringBuffer should not be used	坏味道	主要
Generic exceptions should never be thrown	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
Classes with only "static" methods should not be instantiated	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

Unused type parameters should be removed	坏味道	主要
"switch" statements should not have too many "case" clauses	坏味道	主要
Unused "private" methods should be removed	坏味道	主要
Redundant pairs of parentheses should be removed	坏味道	主要
Ternary operators should not be nested	坏味道	主要
Inner class calls to super class methods should be unambiguous	坏味道	主要
Nullness of parameters should be guaranteed	坏味道	主要
Only static class initializers should be used	坏味道	主要
Unused method parameters should be removed	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
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
Labels should not be used	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
"@Override" should be used on overriding and implementing methods	坏味道	主要
"entrySet()" should be iterated when both the key and value are needed	坏味道	主要
Assignments should not be made from within sub-expressions	坏味道	主要
"Preconditions" and logging arguments should not require evaluation	坏味道	主要
Java 8's "Files.exists" should not be used	坏味道	主要
Two branches in a conditional structure should not have exactly the same implementation	坏味道	主要
Sections of code should not be commented out	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
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
"for" loop stop conditions should be invariant	坏味道	主要
Inheritance tree of classes should not be too deep	坏味道	主要
Unused "private" classes should be removed	坏味道	主要
A field should not duplicate the name of its containing class	坏味道	主要
Dead stores should be removed	坏味道	主要
"DateUtils.truncate" from Apache Commons Lang library should not be used	坏味道	主要
Local variables should not shadow class fields	坏味道	主要
"Thread.sleep" should not be used in tests	坏味道	主要
Tests should not be ignored	坏味道	主要
Anonymous inner classes containing only one method should become lambdas	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
Deprecated elements should have both the annotation and the Javadoc tag	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
"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
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
String function use should be optimized for single characters	坏味道	主要
Assignments should not be redundant	坏味道	主要
"java.nio.Files#delete" should be preferred	坏味道	主要
Methods should not have identical implementations	坏味道	主要
Asserts should not be used to check the parameters of a public method	坏味道	主要
Source files should not have any duplicated blocks	坏味道	主要
Field 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
Type parameter 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

Package names should comply with a naming convention	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
Useless imports should be removed	坏味道	次要
Return of boolean expressions should not be wrapped into an "if-then-else" statement	坏味道	次要
Boolean literals should not be redundant	坏味道	次要
Local variables should not be declared and then immediately returned or thrown	坏味道	次要
Deprecated "${pom}" properties should not be used	坏味道	次要
Unused local variables should be removed	坏味道	次要
Catches should be combined	坏味道	次要
Null checks should not be used with "instanceof"	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
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
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
Method names should comply with a naming convention	坏味道	次要
Class names should comply with a naming convention	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
"switch" statements should have at least 3 "case" clauses	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
Methods should not return constants	坏味道	次要
The default unnamed package should not be used	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
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
"@Deprecated" code should not be used	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
Lambdas should be replaced with method references	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
JUnit rules should be used	坏味道	次要
Annotation repetitions should not be wrapped	坏味道	次要
Lamdbas 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
Composed "@RequestMapping" variants should be preferred	坏味道	次要
Fields in non-serializable classes should not be "transient"	坏味道	次要
Empty statements should be removed	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
"finalize" should not set fields to "null"	坏味道	次要

URIs should not be hardcoded	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
The diamond operator ("<>") should be used	坏味道	次要
"throws" declarations should not be superfluous	坏味道	次要
Modifiers should be declared in the correct order	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

质量配置	xml:Sonar way Bug:1	
规则	类型	违规级别
XML files containing a prolog header should start with "<?xml" characters	Bug	严重