**智能抄表业务系统代码测试报告**

**程序版本v1.0.6**

**2018年12月**

|  |  |  |  |
| --- | --- | --- | --- |
| **文档编号** |  | **保 密 等 级** |  |
| **作 者** |  | **最后修改日期** |  |
| **审 核 人** |  | **最后审批日期** |  |
| **批 准 人** |  | **最后批准日期** |  |

**修订记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本** | **修订日期** | **原因与修改情况描述** | **位置（页/段落/章节号）** | **修订人** | **审核人** |
| V1.0 | 2018.12.13 | 新建 | 全文 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目 录

[目 录 4](#_Toc532927291)

[1. 引言 5](#_Toc532927292)

[1.1. 编写目的 5](#_Toc532927293)

[2. 背景 6](#_Toc532927294)

[3. 测试明细 7](#_Toc532927295)

[3.1. 启动SpotBugs工具 7](#_Toc532927296)

[3.2. 选择项目编译后Jar包及源文件目录 7](#_Toc532927297)

[3.3. 分析代码，并查看分析结果 8](#_Toc532927298)

[3.4. 分析后代码中缺陷的种类 9](#_Toc532927299)

[3.4.1. EC\_UNRELATED\_TYPES 9](#_Toc532927300)

[3.4.2. NP\_ALWAYS\_NULL 10](#_Toc532927301)

[3.4.3. INT\_BAD\_COMPARISON\_WITH\_NONNEGATIVE\_VALUE 10](#_Toc532927302)

[3.4.4. DM\_DEFAULT\_ENCODING 11](#_Toc532927303)

[3.4.5. UC\_USELESS\_CONDITION 13](#_Toc532927304)

[4. 修改SpotBugs提示的各项缺陷 14](#_Toc532927305)

[4.1. 使用SpotBugs重新分析修改后的版本 14](#_Toc532927306)

[5. 结论 15](#_Toc532927307)

# 引言

## 编写目的

本报告为《智能抄表业务系统V1.0.6》的代码测试报告，目的在于提升代码的可读性、可维护性，建立团队共同的编码风格，提升业务系统的代码质量。

本报告的代码测试使用SpotBugs（代码静态分析工具）进行代码静态检查。

# 背景

软件名称：智能抄表系统V1.0.6

测试类别：□集成测试 □系统测试 □集成测试+系统测试

■其他 代码静态测试

承担测试任务部门：\*\*测试组

测试人员： 蒋桂齐

项目经理：

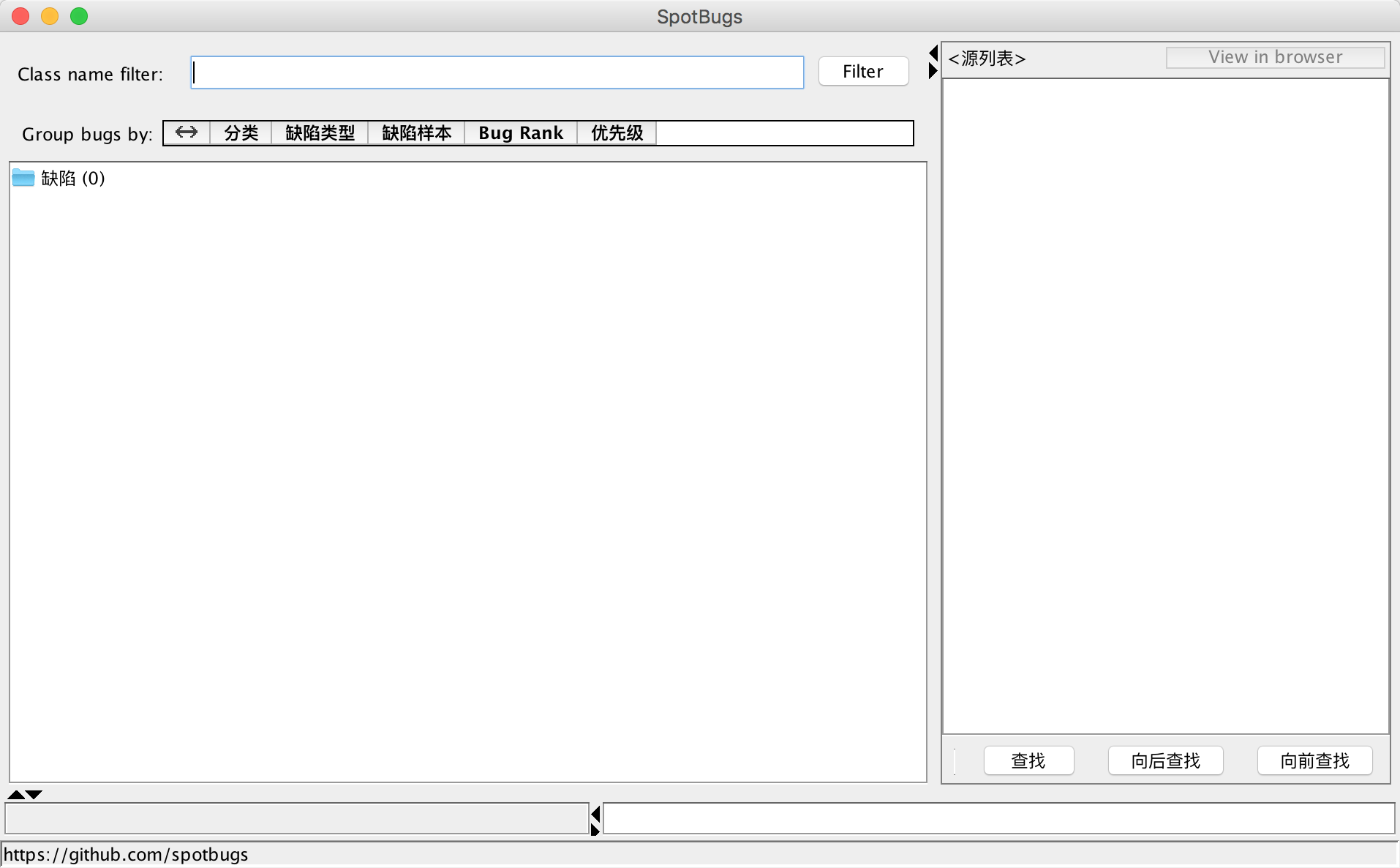
测试经理：

支持人员：

# 测试明细

## 启动SpotBugs工具

代码测试统一使用图形化SpotBugs工具进行检测，可以通过命令行直接启动，启动后界面如下图：



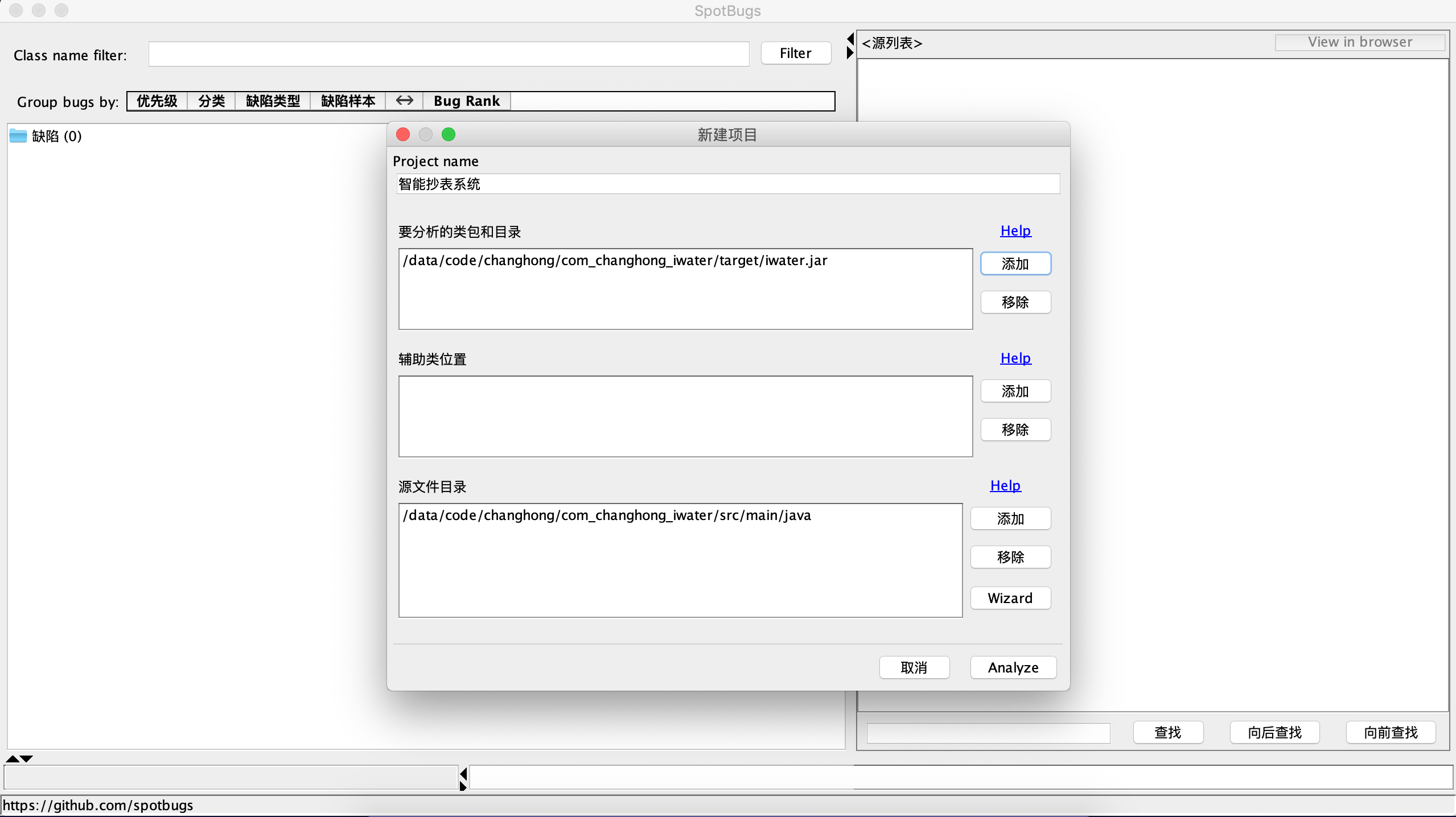
（图一）

## 选择项目编译后Jar包及源文件目录

SpotBugs工具启动后，选择“项目->新建”来启动一个新的代码测试，如[图二](#图二)所示。

其中，我们需要填入项目名称，选择要检查的类Jar包或者包含class文件的目录，以及相应的源文件根目录。

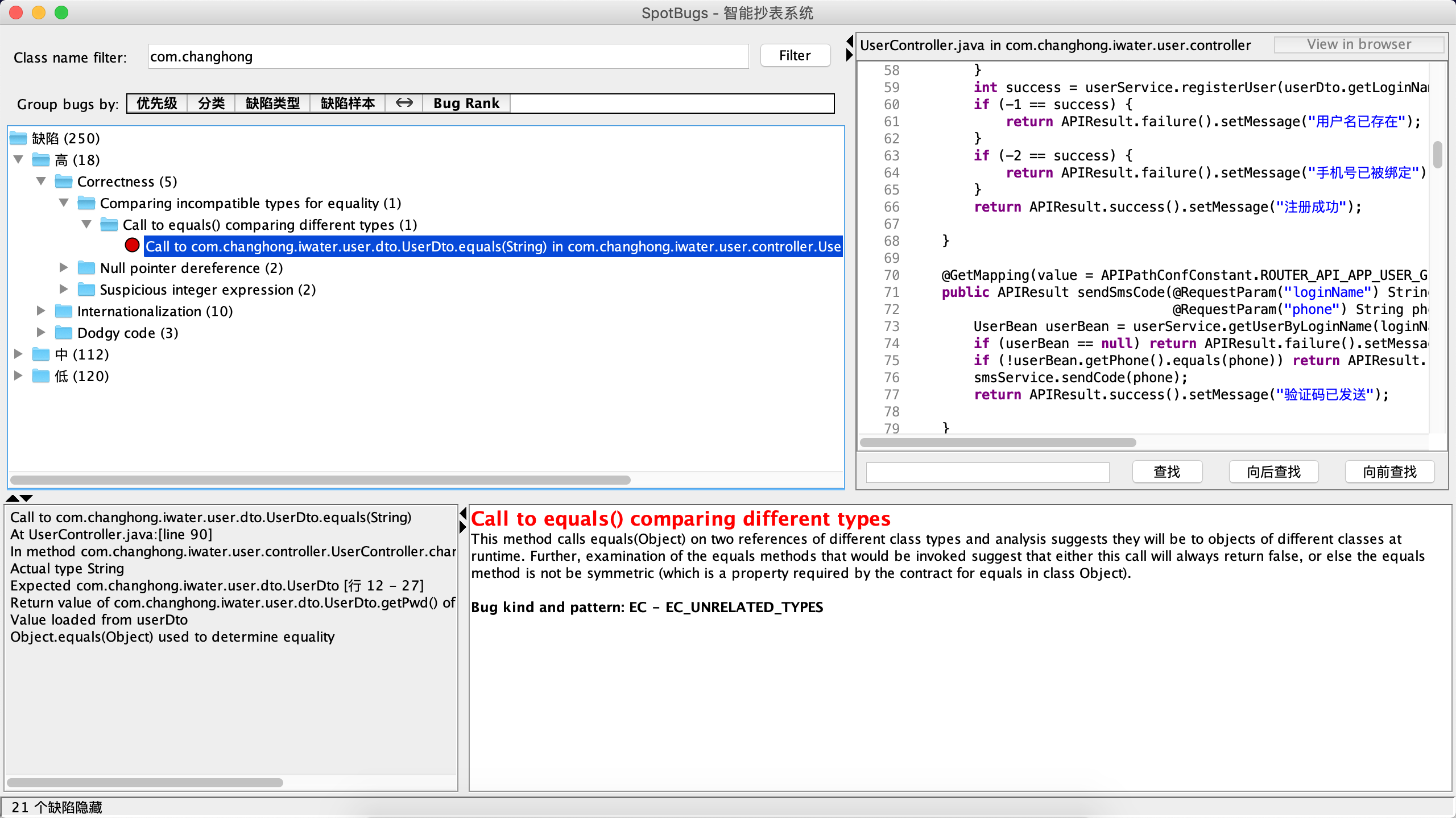
此外，如果我们的代码依赖一些第三方的jar包，我们也可以通过添加辅助类的形式进行增加。



（图二）

## 分析代码，并查看分析结果

项目配置完毕后，选择“分析”按钮，SpotBugs就会自动分析我们的代码：



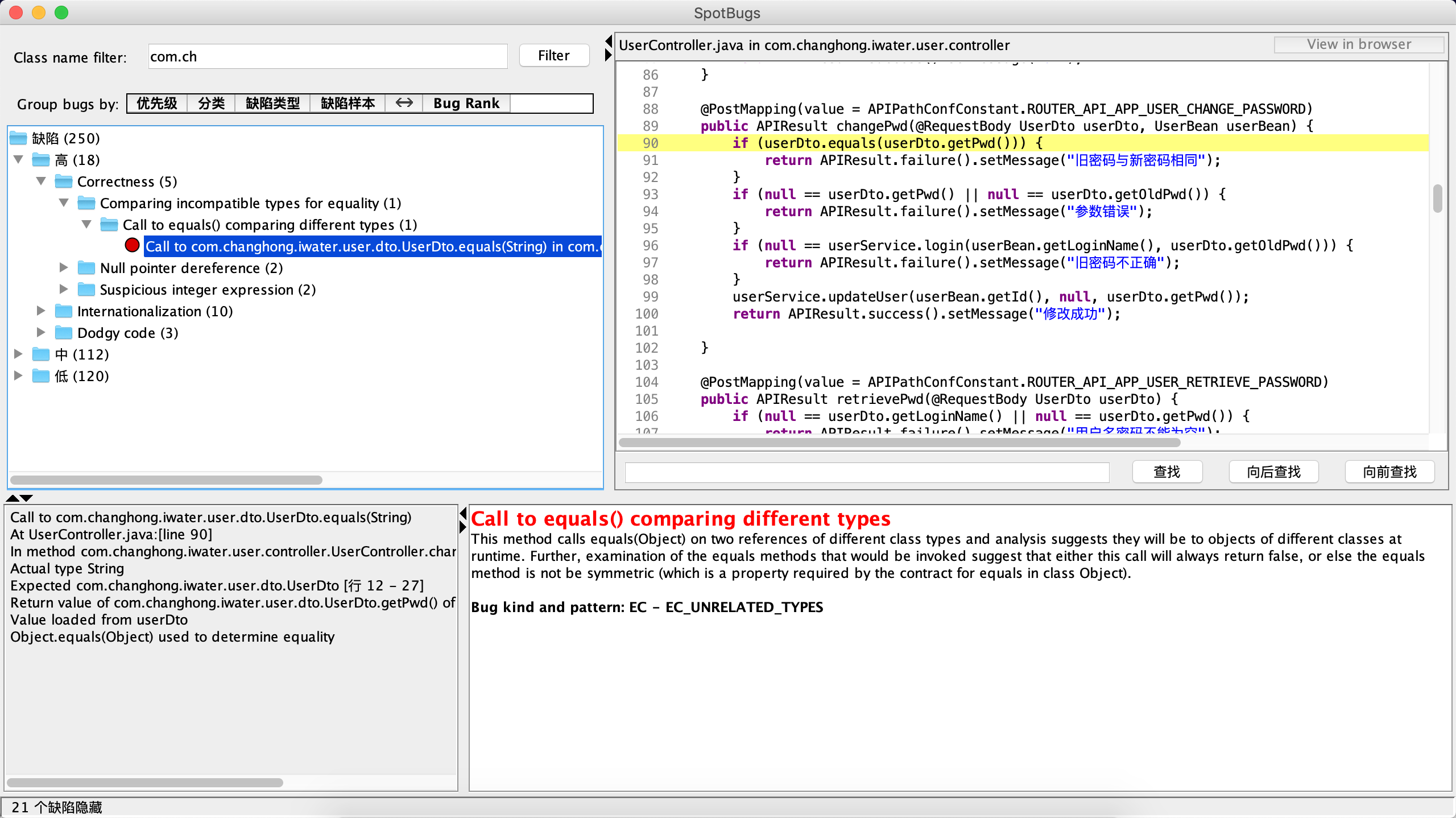
从上图可以发现，智能抄表系统共发现风险：

|  |  |  |
| --- | --- | --- |
| **风险等级** | **数量** | **是否需要解决** |
| 高 | 18 | 是 |
| 中 | 112 | 是 |
| 低 | 120 | 否 |

其中低风险等级的缺陷不需要解决的原因为代码冗余、选择结构未提供默认的选项、未使用有返回值的函数等可忽略的风险。

# 代码中高等级缺陷分类

## EC\_UNRELATED\_TYPES



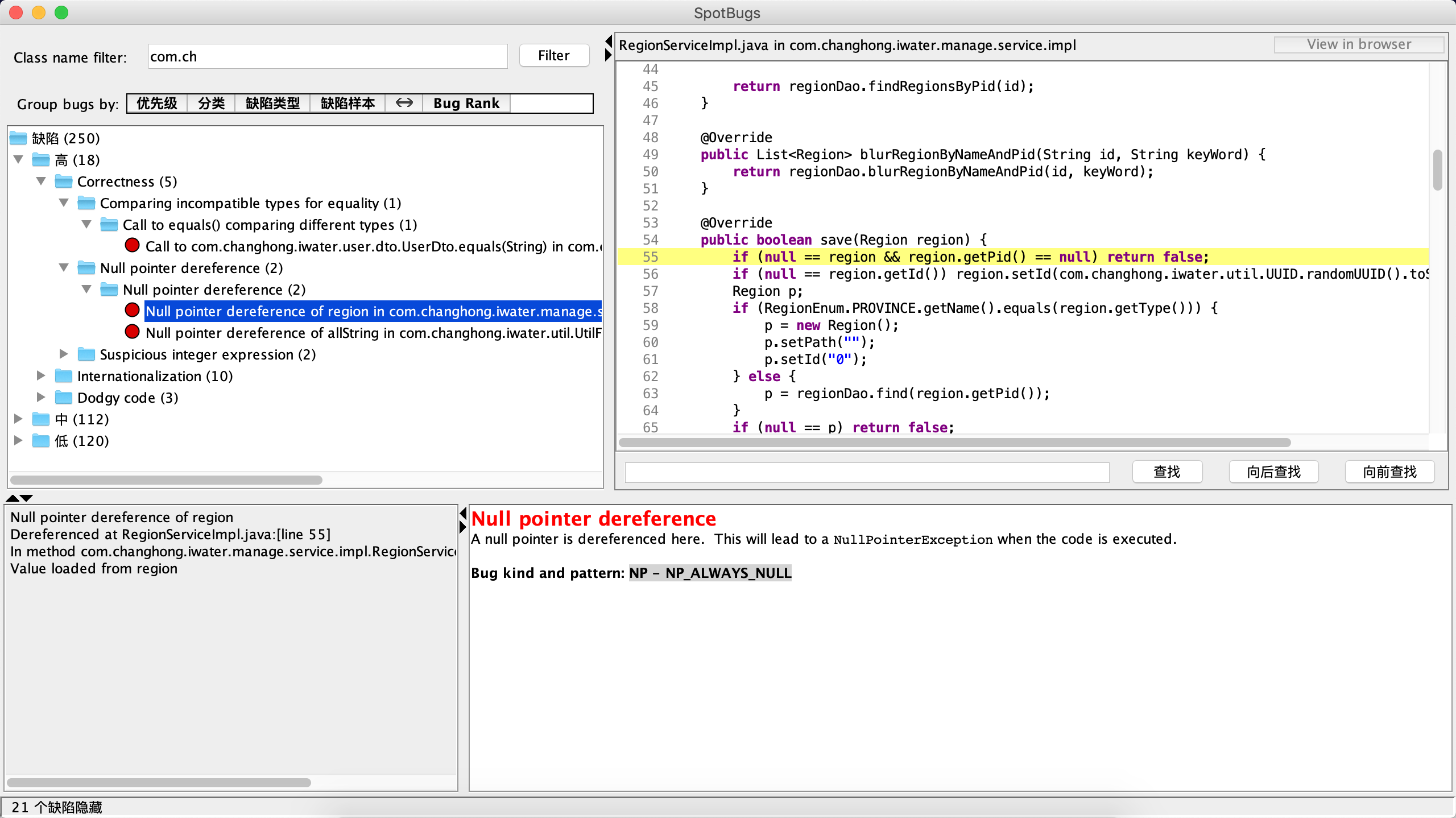
缺陷信息:

Call to equals() comparing different types。

缺陷及解决方法描述：

This method calls equals(Object) on two references of different class types and analysis suggests they will be to objects of different classes at runtime. Further, examination of the equals methods that would be invoked suggest that either this call will always return false, or else the equals method is not be symmetric (which is a property required by the contract for equals in class Object).

## NP\_ALWAYS\_NULL



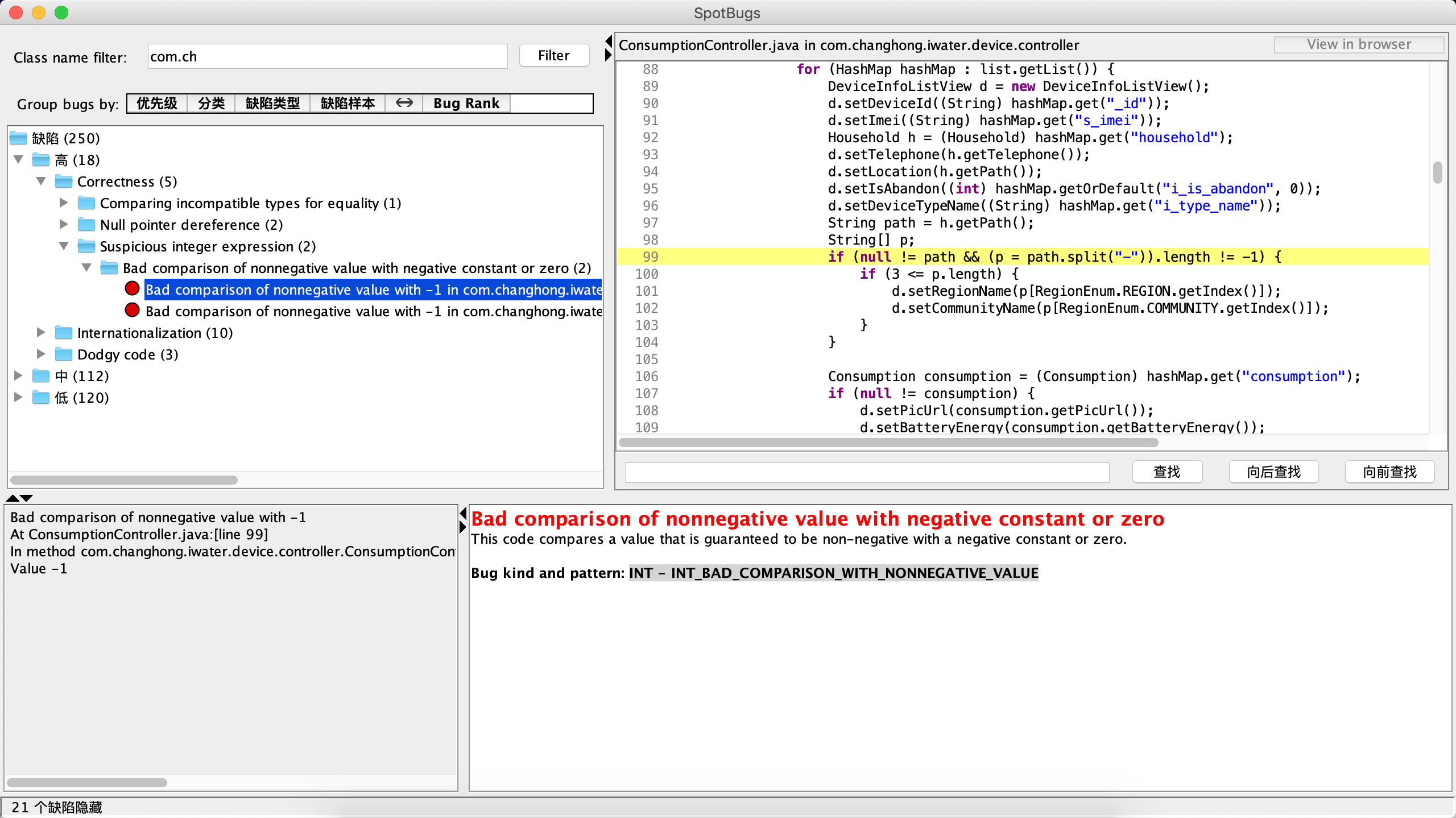
缺陷信息：

Null pointer dereference。

缺陷及解决方法描述：

A null pointer is dereferenced here.  This will lead to a NullPointerException when the code is executed.

## INT\_BAD\_COMPARISON\_WITH\_NONNEGATIVE\_VALUE



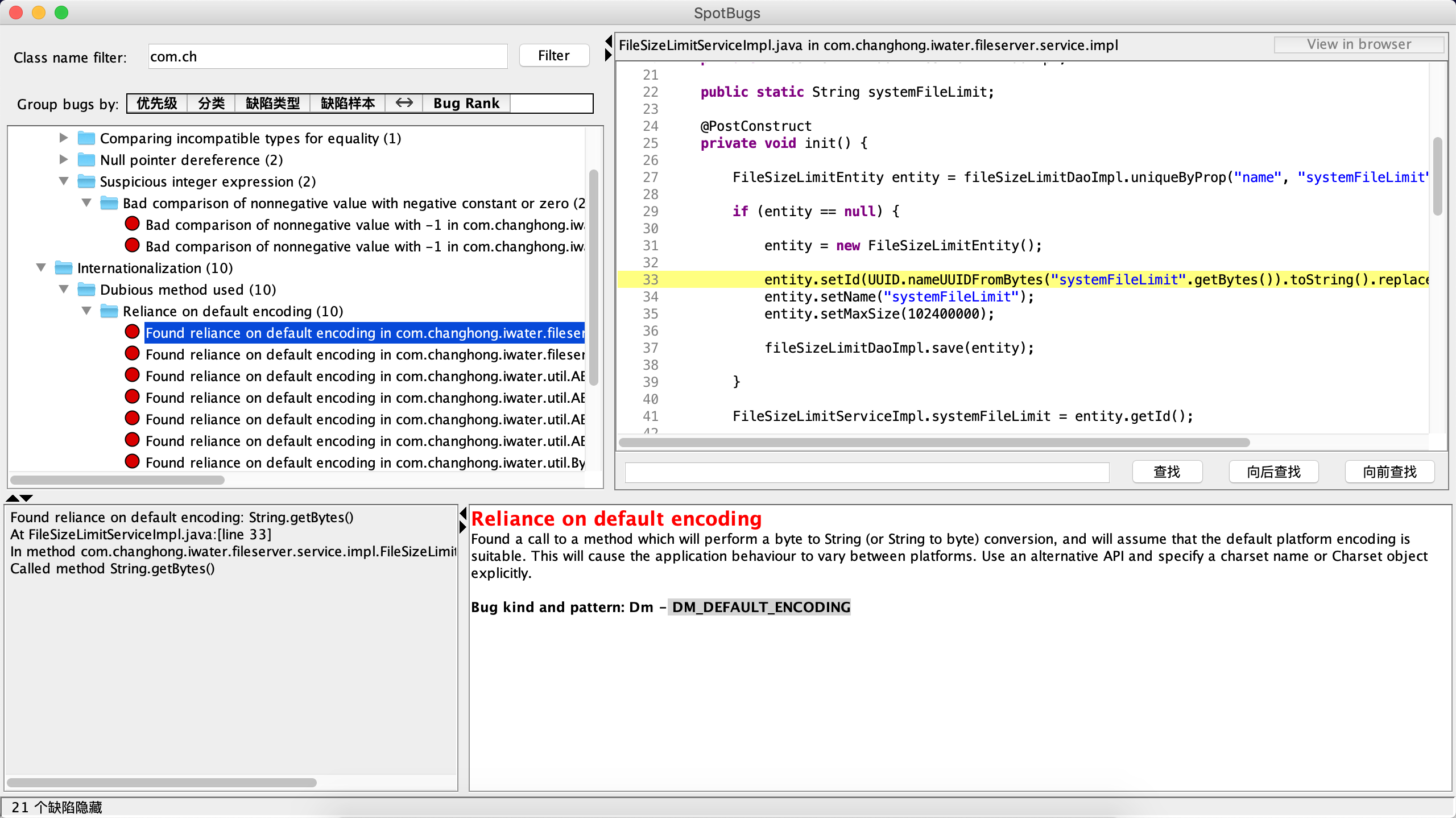
缺陷信息：

Bad comparison of nonnegative value with negative constant or zero。

缺陷及解决方法描述：

This code compares a value that is guaranteed to be non-negative with a negative constant or zero.

## DM\_DEFAULT\_ENCODING



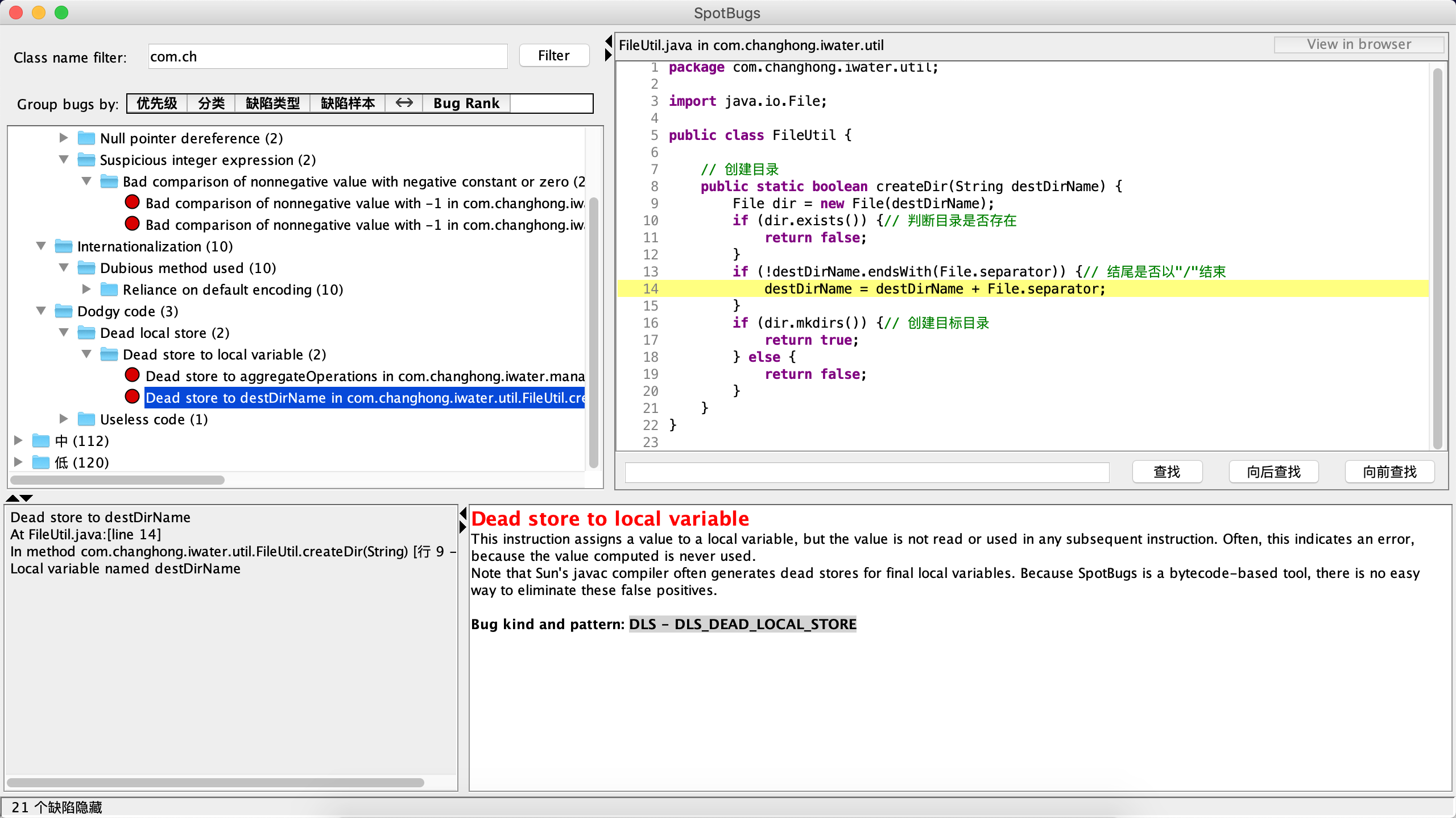
缺陷信息：

Reliance on default encoding。

缺陷及解决方法描述：

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

* 1. DLS - DLS\_DEAD\_LOCAL\_STORE



缺陷信息：

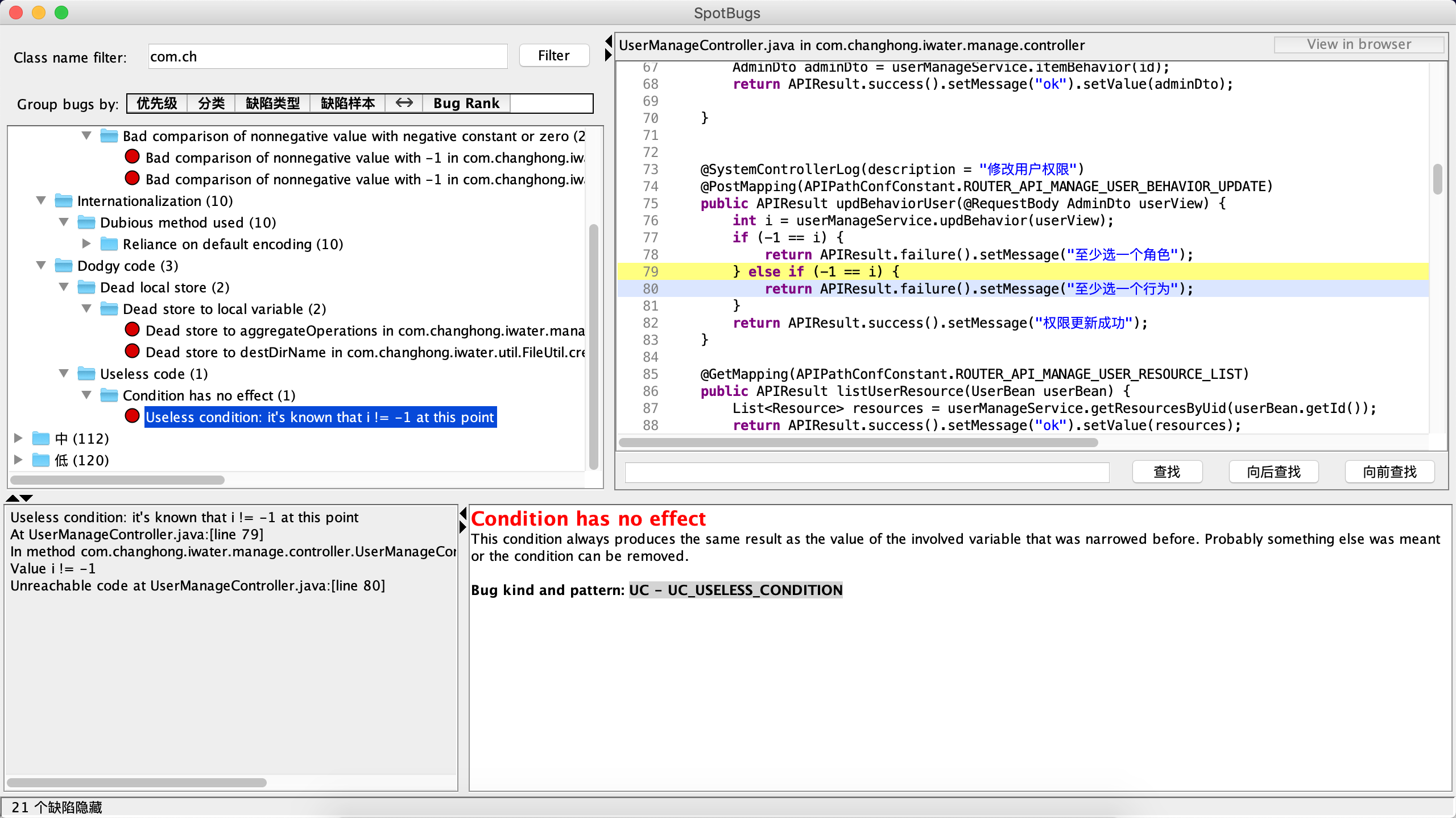
Dead store to local variable

缺陷及解决方法描述：

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

Note that Sun's javac compiler often generates dead stores for final local variables. Because SpotBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.

## UC\_USELESS\_CONDITION



缺陷信息：

Condition has no effect

缺陷及解决方法描述：

This condition always produces the same result as the value of the involved variable that was narrowed before. Probably something else was meant or the condition can be removed

## 所有缺陷明细

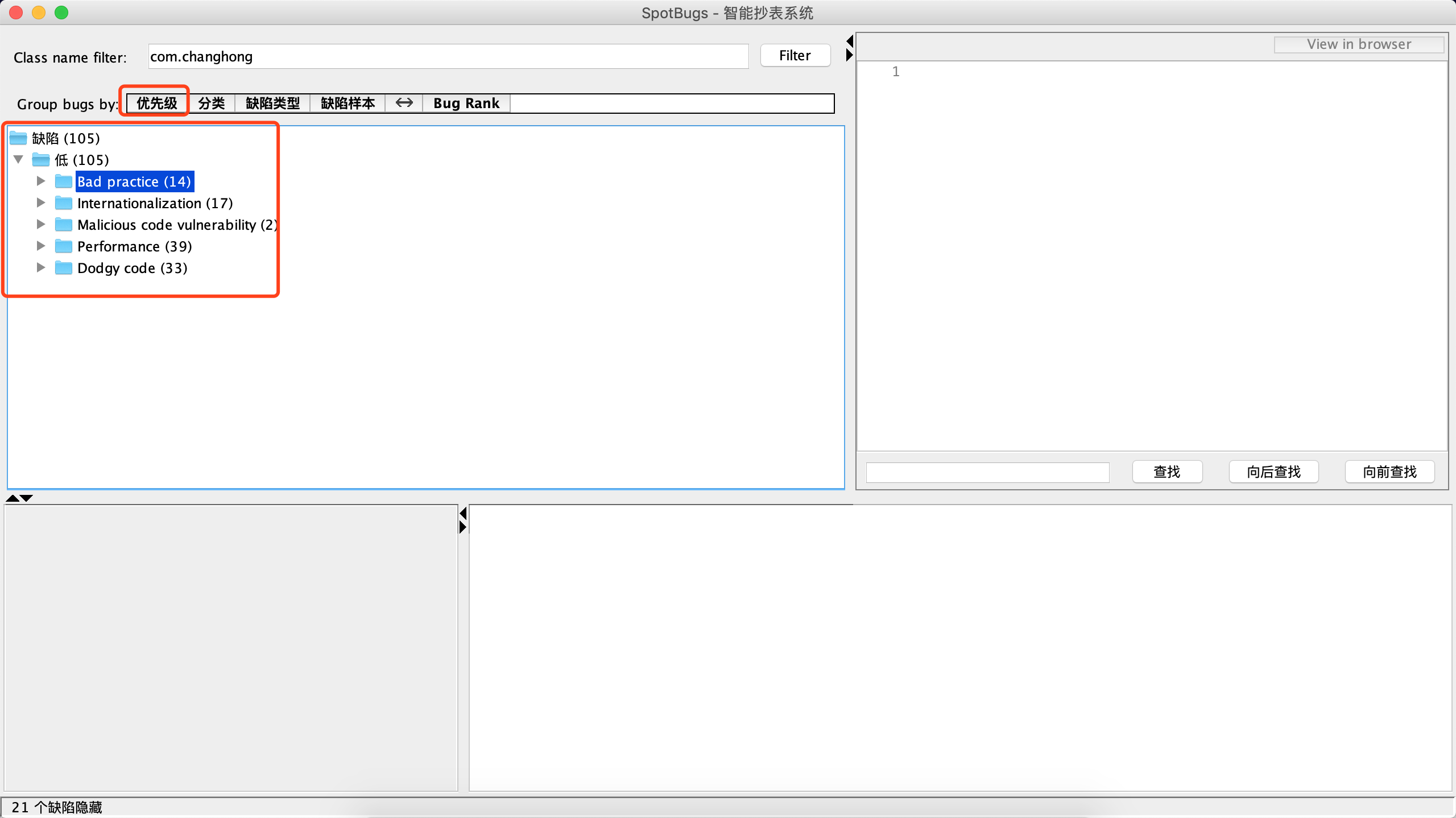
请查阅：[Iwater-bugs-old](bugs/iwater-bugs-old.html)

# 修改SpotBugs提示的各项缺陷

我们根据SpotBugs的提示，修改所有除第三方依赖包代码的缺陷，解决我们自身代码中全部的高风险和中风险级别代码缺陷。缺陷修改完毕后，代码功能测试正常。

## 使用SpotBugs重新分析修改后的版本

再次启动SpotBugs，并按照步骤1~3进行代码静态测试。



## 高、中缺陷处理后剩余缺陷明细

请查阅：[Iwater-bugs-new](bugs/iwater-bugs-new.html)

# 结论

使用SpotBugs（代码静态分析工具）发现了大量的代码缺陷，并根据SpotBugs的提示对这些缺陷进行了修复：

|  |  |  |  |
| --- | --- | --- | --- |
| **风险等级** | **数量** | **修复数量** | **剩余数量** |
| 高 | 18 | 18 | 0 |
| 中 | 112 | 112 | 0 |
| 低 | 120 | 0 | 120 |