

Refactoring Álvaro

Index:

Introduction	3
Bad smells	4
Define constant instead of duplicating literal	4
This block of commented out lines should be removed	4
Remove this unused import	5
Replace the type specification in the constructor call with the literal (<>)	5
Use the primitive expression boolean here	6
Remove the literal "true" boolean value	6
Remove this useless local variable	7
Remove this unused private field	7
Either remove or fill this block of code	7
Remove this "public" modifier	8
Add at least one assertion to this use case	8
Remove the declaration of thrown exception "DataAccessException" that is a runtime exception	9
Complete the task associated with this TODO comment	9
Immediately return this expression instead of assigning it to the temporary variable	9
Code changed	10
Result	11

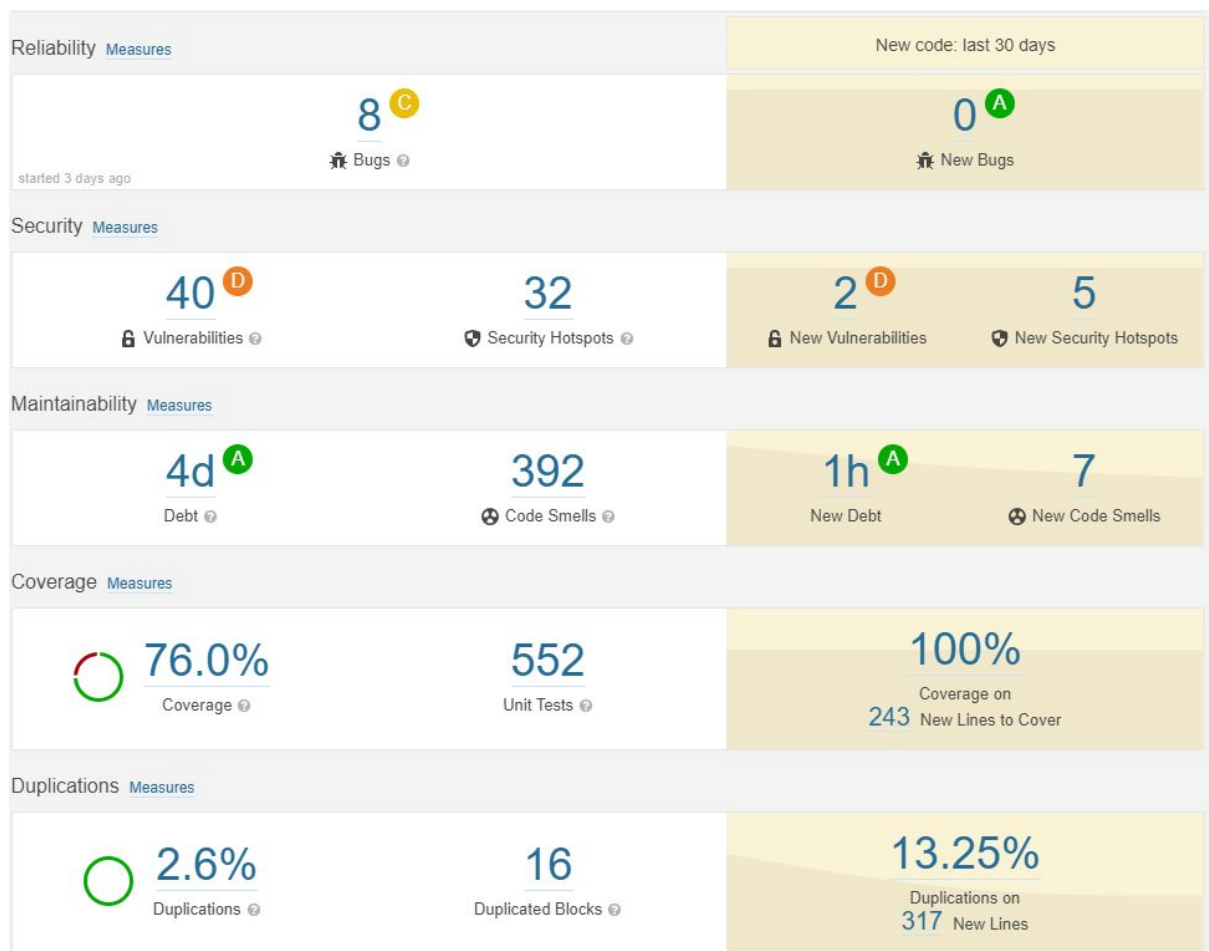
Introduction

The goal of this refactoring was intended to be the refactorization of the classes related with a selected US, but instead, after looking the amount of bad smells our code has been stacking, the refactorization has focus every code I have done so far.

That been, the five US I have work into, the tests, repositories, models, services, validators and some classes that were already in the source code provided, but contained some bad smells as well.

In this document are described every bad smell that has been encountered, the number of them presence in the code, the reason why It should be avoid following the documentation of SonarCloud about good practices and how they were solved (Or the reason we could not solve them).

Here is the result of the analysis of SonarCloud after this refactorization:



As It is observable, if we do not start working in cleaning the code, in futures iterations it will become a serious problem difficult to solve.

Bad smells

Define constant instead of duplicating literal

Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences.

src/.../samples/petclinic/web/InterventionController.java

Define a constant instead of duplicating this literal "pets/createOrUpdateInterventionForm" 4 times. Why is this an issue?

2 months ago ▾ L69 🔗 ⚙️ ▾

🔗 Code Smell 🚨 Critical 🔵 Open Not assigned 10min effort 🛠️ design

Define a constant instead of duplicating this literal "intervention" 3 times. Why is this an issue?

2 months ago ▾ L76 🔗 ⚙️ ▾

🔗 Code Smell 🚨 Critical 🔵 Open Not assigned 8min effort 🛠️ design

On the other hand, constants can be referenced from many places, but only need to be updated in a single place.

There were found 10 of this smells in the code analyzed. The solution for this problem is simple:

We define a private final static String that will substitute the literals:

```
@Controller
public class InterventionController {

    private final PetService petService;

    private static final String CREATE_FORM = "pets/createOrUpdateInterventionForm";
    private static final String INTERVENTION = "intervention";
```

This block of commented out lines should be removed

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.

This block of commented-out lines of code should be removed. Why is this an issue?

2 months ago ▾ L106 🔗 ⚙️ ▾

🔗 Code Smell 🚨 Major 🔵 Open Not assigned 5min effort 🛠️ unused

This block of commented-out lines of code should be removed. Why is this an issue?

2 months ago ▾ L108 🔗 ⚙️ ▾

🔗 Code Smell 🚨 Major 🔵 Open Not assigned 5min effort 🛠️ unused

This block of commented-out lines of code should be removed. Why is this an issue?

2 months ago ▾ L120 🔗 ⚙️ ▾

🔗 Code Smell 🚨 Major 🔵 Open Not assigned 5min effort 🛠️ unused

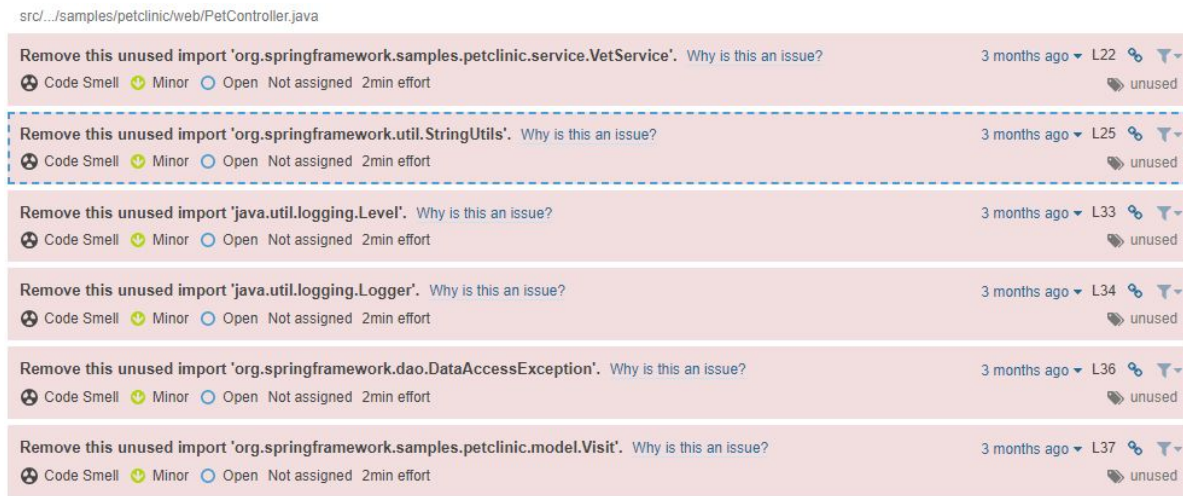
There were found 8 of this smells in the code analyzed. To solve this bad smell we simply deleted every line of code, been them unnecessary for a clean code.

Remove this unused import

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.

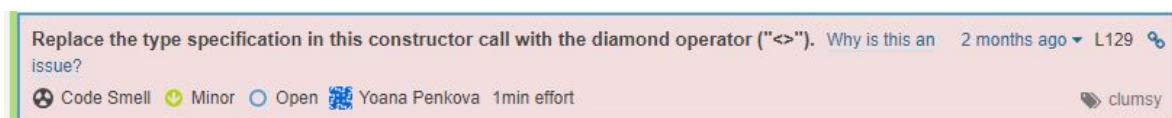
Leaving them in reduces the code's readability, since their presence can be confusing.



This was smells was the most abundant, in total having 26 warnings. It was solved by deleting the unused imports.

Replace the type specification in the constructor call with the literal (<>)

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.



This error could not be solved. The methods having this issue were already defined by spring and because of the lack of knowledge about the source code we were not confidence to start changing some methods that would affect every aspect of the application. In total, there were 7 inside the code refactored.

Use the primitive expression boolean here

When boxed type `java.lang.Boolean` is used as an expression it will throw `NullPointerException` if the value is `null` as defined in [Java Language Specification §5.1.8 Unboxing Conversion](#).

It is safer to avoid such conversion altogether and handle the `null` value explicitly.

Use the primitive boolean expression here. Why is this an issue?

2 months ago ▾ L135 🔗

🚩 Code Smell 🟢 Minor 🔓 Open 👤 Yoana Penkova 5min effort 🐞 pitfall

```
Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
String username = ((UserDetails) principal).getUsername();
Optional<Vet> vet = this.petService.findVetByName(username);

if (vet.isPresent()) {
    intervention.setVet(vet.get());
}
this.petService.saveIntervention(intervention);
return "redirect:/owners/{ownerId}";
}
```

For some methods, like the one above, we cannot be sure when an optional value will return an object (in this case a vet) or a null value. This output is necessary for the input of other methods so we have to check them first. For this reason and also not knowing another way to solve it this bad smell has been untouched. In total we found 7 in the code refactored.

Remove the literal “true” boolean value

Redundant Boolean literals should be removed from expressions to improve readability.

Remove the literal "true" boolean value. Why is this an issue?

2 months ago ▾ L162 🔗

🚩 Code Smell 🟢 Minor 🔓 Open 👤 Yoana Penkova 5min effort 🐞 clumsy





It has been simply solved by checking with others methods the boolean value. In total we solved 7.

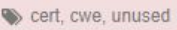
Remove this useless local variable

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.

Remove this useless assignment to local variable "view". Why is this an issue?

2 months ago ▾ L281 🔗

 Code Smell  Major  Open  Yoana Penkova 15min effort



Been an unused value the solution was to simply deleting it. In total there was just 1 in the code refactored.

Remove this unused private field

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.

Remove this unused "visitService" private field. Why is this an issue?

3 months ago ▾ L46 🔗 ▾

 Code Smell  Major  Open  Yoana Penkova 5min effort



Like before, the unused privates fields were deleted. In total, only 2 of this smell.

Either remove or fill this block of code

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.

Either remove or fill this block of code. Why is this an issue?

yesterday ▾ L174 🔗

 Code Smell  Major  Open  Yoana Penkova 5min effort



For this bad smell, it was not clear were the problem was at all. It is an indication that we should study the code because it could produce bugs but there were no clues about solving it. Only 1 class with this smell was found.

Remove this “public” modifier

JUnit5 is more tolerant regarding the visibilities of Test classes than JUnit4, which required everything to be `public`.

In this context, JUnit5 test classes can have any visibility but `private`, however, it is recommended to use the default package visibility, which improves readability of code.

Remove this 'public' modifier. [Why is this an issue?](#) last month ▾ L23 🔗

 Code Smell  Info  Open Not assigned 2min effort  junit, tests

Although test should be private, for all of our test it is required to have a public class, thus, we could not work with this bad smell. We got 6.

Add at least one assertion to this use case

A test case without assertions ensures only that no exceptions are thrown. Beyond basic runnability, it ensures nothing about the behavior of the code under test.

Add at least one assertion to this test case. [Why is this an issue?](#) last month ▾ L86 🔗 ▾

 Code Smell  Blocker  Open Not assigned 10min effort  junit, tests

Regarding this bad smells, for some of our test we had already implemented methods that performed “assertions”, so even SonarClouds sees it has a bad signed, it is not a real problem. 6 cases were found in the code refactored.

```
103     private boolean isElementPresent(final By by) {
104         try {
105             this.driver.findElement(by);
106             return true;
107         } catch (NoSuchElementException e) {
108             return false;
109         }
110     }
```


Remove the declaration of thrown exception “DataAccessException” that is a runtime exception

An exception in a `throws` declaration in Java is superfluous if it is:

- listed multiple times
- a subclass of another listed exception
- a `RuntimeException`, or one of its descendants
- completely unnecessary because the declared exception type cannot actually be thrown

The only solution I had knowledge of it was to build a try catch structure in the code to deal with the runtime exception, beed this solutions a worst option. This bad smell has not been solved. There were 12 found in the code refactored.

Complete the task associated with this TODO comment

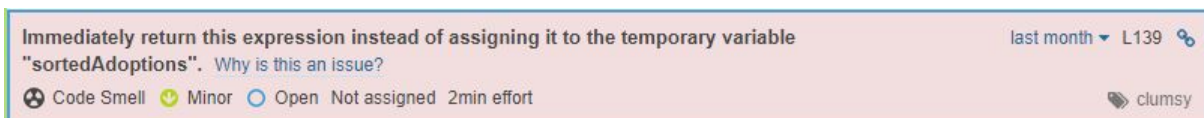
`TODO` tags are commonly used to mark places where some more code is required, but which the developer wants to implement later.



It was solved by deleted the commented line. “ found in the code refactored.

Immediately return this expression instead of assigning it to the temporary variable

Declaring a variable only to immediately return or throw it is a bad practice.



The solution was to delete the declaration of the variable that was the solution and instead give the solution in the return line. Only 1 found.

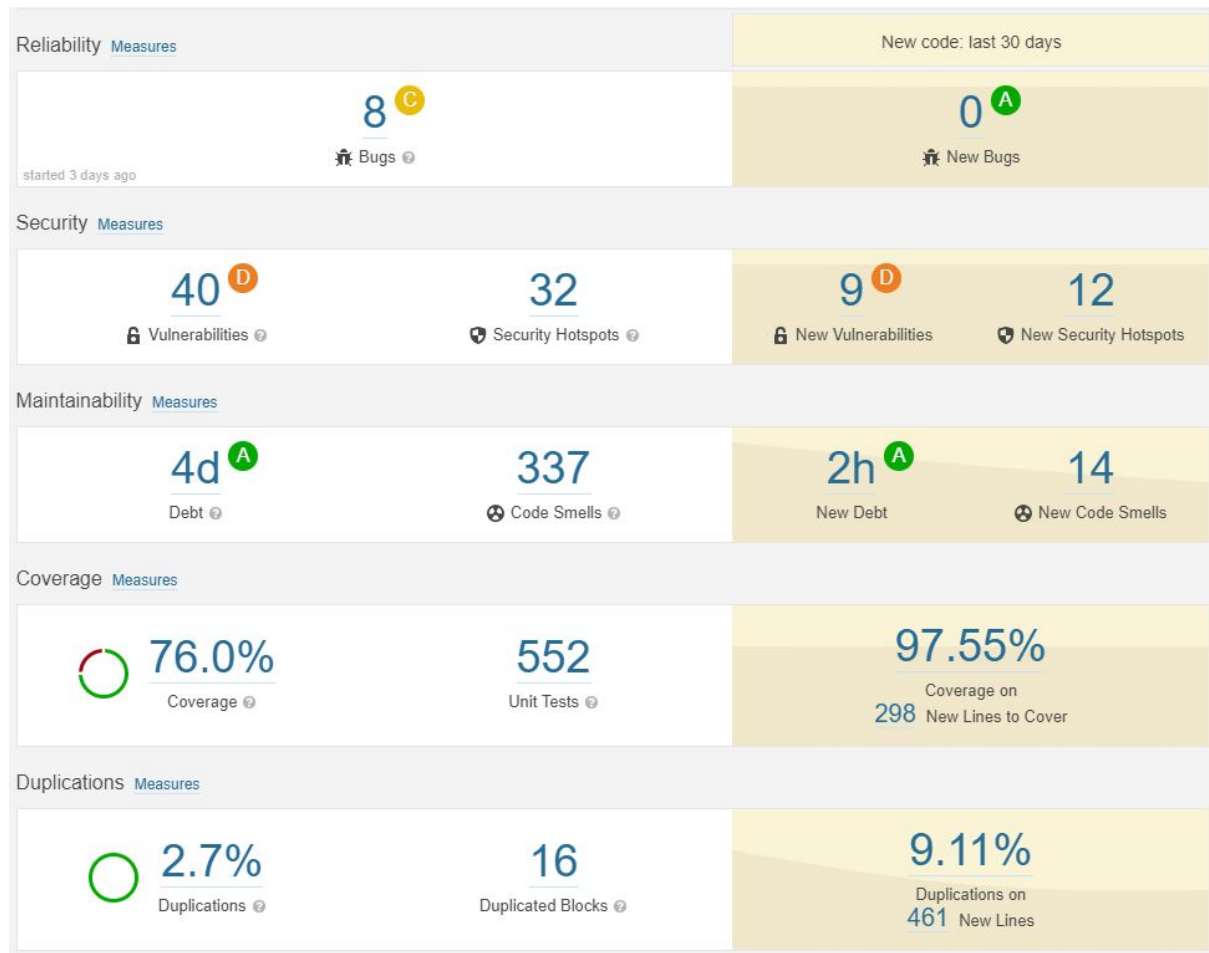
Code changed

Here we have every class that have been refactored. The classes with the “+” means that every bad smell have been solved for this class, if they have “-” then it means that nor every bad smell could be solved.

- InterventionController +
- OwnerController +
- PetController +
- UserController +
- VetController -
- VisitController +
- WelcomeController +
- OwnerSeesVetPersonallInformationNegativeUITest -
- OwnerSeesVetPersonallInformationPositiveUITest -
- VetManagesInterventionPositiveUITest -
- VetManagesInterventionNegativeUITest -
- OwnerRepository +
- InterventionRepository -
- InterventionService -
- OwnerService -
- VisitService -
- InterventionValidator +
- Owner +
- PetTypeFormatter +
- SpecialtyFormatter +

Result

After every changed I have performed a new analysis of the code in SonarCloud. The result was the next:



The bad smells have decreased in 55. It is a significant result keeping in mind that this refactorization only embrace a 20% of the code.

Even with this positive result, the report shows that there were more bad smells that can be solve and it is recommended a more deep refactoring an perform more analysis for each iteration.