Lint report

Workgroup number: E8.01
Repository: https://github.com/Icaro212/Acme-Toolkits
Date: 2022/05/23

Name	Corporate e-mail
Carrasco Núñez, Alejandro	alecarnun@alum.us.es
Durán Terrero, Andrés	anddurter@alum.us.es
López Benítez, Pablo Delfín	pablopben@alum.us.es
Núñez Moreno, Pablo	pabnunmor@alum.us.es
Robledo Campa, Pablo José	pabrobcam@alum.us.es
Vázquez Monge, Francisco Javier	fravazmon@alum.us.es

Table of Contents

Executive summary

The following document contains an analysis of the bad smells found on our project. The bad smells shown in this report are those that we think are innocuous to the project, and that must be kept until said otherwise by the professor.

Revision table

Revision number	Date	Description
v1	2022/05/22	Initial version

Introduction

The contents of the paper focus specifically on the different types of bad smells that exist in our project. These are specifically the ones to be kept in the project as we have considered them to be harmless to the future development of the project, meaning that pose no threat to the correct progression and behavior of our application. Of course, to each of these bad smells, an explanation is provided so that the reader may understand why there are to be remains.

The structure upon which we will take a look into this report will be the following: We will by giving a list of the different bad smells found in the project, and explain in each one of them the reason behind the no removal of it.

Contents

During the analysis, SonarLint has reported three different types of potential bugs and no code smells.
However, we believe that these potential issues represent no harm in the code and we will now state why we think this.

Firstly, in the SystemConfiguration.java class, SonarLint reports two equal major potential bugs. The description of these potential bugs is the following: “Refactor this repetition that can lead to a stack overflow for large inputs”. This is related to two attributes (“strongSpamTerms” and “weakSpamTerms ”) and the regular expression they have to follow. This potential bug warns us that the regular expression might cause a stack overflow when trying to match the regular expression, if the attribute is big enough. In this case, as these two attributes are provided in the sample data, we know what we are introducing so, a stack overflow will not occur. Therefore, we do not consider this a problem in our context.

Moreover, we are informed of a possible code smell: “Regular expressions should not be too complicated”. This mainly happens, we believe, to the complexity that the “or” operator (|) introduces in our expression, as well as making use of the negative lookahead (?!,). This is necessary for considering all the punctuation symbols (\p{P}) except the comma, because that would indicate our delimiter.

After discussing both these situations with our professor, Rafael Corchuelo, we concluded saying that there was nothing to worry about.

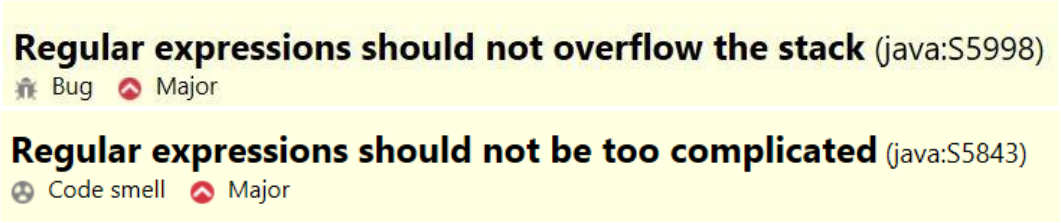

Image 1: title of the first potential bug and code smell

```
@NotBlank
@Pattern(
        regexp = "^([\\p{L}\\p{N}\\s]|(?!,)[\\p{P}])+(, *([\\p{L}\\p{N}\\s]|(?!,)[\\p{P}])+)*$",
        message = "{acme.validation.system-configuration.pattern.spam-terms}")
protected String strongSpamTerms;
```

Image 2: example of the code that gives the potential bug and code smell

Secondly, in several “form.jsp” files, we encounter another major potential bug repeated several times. This potential bug, described as “Add either an 'id' or a 'scope' attribute to this <th> tag.”, states that we should add an identifier to each of the <th> tags in our project. However, taking into account that we use the code of the <acme:message> tag inside the <th> tag, we consider that in this way each of the <th> tags is identified so further identification is not needed.

Image 3: title of the second potential bug

Thirdly, we also encounter a minor potential bug in several “form.jsp” files. This potential bug is similar to the previous one we talked about, but in this case it refers to the <table> tags. So, as in the previous potential bug, the conclusion is similar. We believe that the contents inside the table describe well enough the table itself so there is no need to describe them.

Image 4: title of the third potential bug

```
<table class="table table-sm">
    <tr>
        <th><acme:message code="administrator.dashboard.form.label.currency"/></th>
        <th><acme:message code="administrator.dashboard.form.label.average"/></th>
        <th><acme:message code="administrator.dashboard.form.label.deviation"/></th>
        <th><acme:message code="administrator.dashboard.form.label.min"/></th>
        <th><acme:message code="administrator.dashboard.form.label.max"/></th>
    </tr>
```

Image 5: example in the source code showing the code that gives the second and third potential bugs

Conclusions

In conclusion, we have not found any relevant code smell in our code. Apart from this, we have revised the three types of potential bugs found and have come to the conclusion that they are not bugs in the end, so we can say that our code is clean and no bugs have been identified.

Bibliography

Intentionally blank.