DP2 2023-2024

Lint report D03

Acme Software Factory

Repository: https://github.com/rafcasceb/Acme-SF-D03

Student #4:

- Mellado Díaz, Luis luimeldia@alum.us.es

Other members:

- Flores de Francisco, Daniel danflode@alum.us.es
- Heras Pérez, Raúl rauherper@alum.us.es
- Castillo Cebolla, Rafael rafcasceb@alum.us.es
- Vento Conesa, Adriana adrvencon@alum.us.es

GROUP C1.049

Version 1.0

24-04-24

Content Table

Executive summary

This report will provide a list of issues detected by Sonar Lint in the project, along with explanations of why they may not be cause for concern. Clear and straightforward language will be used to ensure easy understanding and to help ensure a high-quality final product.

Revision Table

Date	Version	Description of the changes	Sprint
24/04/2024	1.0	• Executive summary • Version Table • Content • Conclusion • Bibliography	3

Introduction

I'll be using the Sonar's Lint plugin for Eclipse to analyze all the files I've created as Student 4 for my individual assignments.

Specifically, we'll be looking at the contents of the Java packages named "entities.sponsorships," "features.sponsor," "features.any.sponsorship," and "features.authenticated.sponsor." Additionally, we'll be analyzing the Sponsor role and the SponsorDashboard form. Populator files such as Sponsorship, Invoice and Sponsor will also undergo analysis, along with view files (forms.jsp, list.jsp, and i18n files) found in folders named "views/sponsor," "views/any/sponsorship," and "views/authenticated/sponsor."

Since most of the bad smells are common to several files I will analyze code smell by code smell instead of class by class.

Contents

Override the "equals" method in this class:
This suggestion arises because it's a common practice to override the "equals" method in Java classes. However, since it's not utilized in our implementation and has been commented out in the class, there's no need for any correction.

Remove the unnecessary Boolean literal:
This bad smells is caused by validations that check if an entity is published: “object.isPublished() == false”. It is true that you can just write “!object.isPublished()” but since it is part of a bigger assertion I consider that is more readable with == false. In any case, this only happens a couple times an I think is worth a little longer expression for the sake of understandability.

Use concise character class syntax '\d' instead of '[0-9]':
This recommendation pertains to the code – like attributes. However, the current implementation using '[0-9]' in the regular expression is valid and arguably more understandable for representing ranges of Latin numbers. Therefore, no correction is necessary.

Rename this package name to match the regular expression '^[a-z_]+(.[a-z_][a-z0-9_])$':
Although there's a suggestion to rename the package, the current name adheres to the framework standard and follows what was taught by professors. Thus, no correction is needed.

Define a constant instead of duplicating this literal:
This recommendation suggests defining a constant instead of duplicating a literal value multiple times in the code. However, in this case, duplicating the literal may not be a significant issue. The justification for its innocuous nature is that the literal is used only in a few places (2 or 3 times at most).

Replace this assert with a proper check:
The assertion 'object != null;' is flagged, but it's the recommended implementation by both professors and the framework. Therefore, there's no need for any modification.

Conclusions

In conclusion, while the code analysis has flagged several potential areas for improvement, upon closer examination, many of these suggestions are deemed innocuous or even preferable based on the specific context of the project.

For instance, suggestions such as overriding the "equals" method, utilizing concise character class syntax, and renaming package names are considered unnecessary as they align with project standards and practices taught by professors.

Additionally, decisions to retain certain code structures, like using explicit boolean comparisons or duplicating literals, are justified by their contribution to code readability and must be said that they only happen a couple times.

Overall, while it's essential to consider and evaluate code suggestions, it's equally important to weigh them against project context and standards to determine the most appropriate course of action. In this case, the existing code practices appear to be well-suited for the project's needs.

Bibliography

Intentionally blank.