Workgroup C1.067

LINT report

26/05/2025

Tudor Cristian Lacatus Cosma – ionlac@alum.us.es

https://github.com/javiarellanoo/Acme-ANS-D04

Table of Contents

[EXECUTIVE SUMMARY: 2](#_Toc199112445)

[REVISION TABLE 3](#_Toc199112446)

[INTRODUCTION 4](#_Toc199112447)

[IDENTIFIED BAD SMELLS 5](#_Toc199112448)

[CONCLUSIONS 7](#_Toc199112449)

[BIBLIOGRAPHY 8](#_Toc199112450)

# EXECUTIVE SUMMARY:

This document offers a comprehensive overview of the potential bad smells identified by Sonar's Lint within Student #2's code. Each reported bad smell will be analyzed, with clear justifications provided for any that are deemed innocuous. Any code smells not included in this report have already been addressed and corrected.

# REVISION TABLE

|  |  |  |
| --- | --- | --- |
| Revision Number | Date | Description |
| 1.0 | 25/05/2025 | Initial version of the document |
|  |  |  |
|  |  |  |

# INTRODUCTION

The aim of this document is to analyze the bad smells found in Student #2's code. These smells are grouped by the different sections of the project where they appear, and a clear justification will be provided for why each code smell has been deemed innocuous.

When a certain code smell is repeated across different features (e.g., recommendations for defining constants), they will be assessed as one, with clear indications of where they were detected.

Finally, a conclusion on the utility of this tool will be presented.

# IDENTIFIED BAD SMELLS

Define a constant instead of duplicating the literal

Sonar's Lint recommended defining constants to avoid repetition of strings throughout the code. This was detected in CustomerBookingCreateService, CustomerBookingPublishService: "Define a constant instead of duplicating the literal flight 3 times." The duplicated literal is found within requests for data to prevent hacking attempts and for the unbinding of the Flight. While it could be replaced by a constant, given its infrequent appearance, it doesn't seem appropriate to use constants when adding entries to the response dataset. The same reasoning applies in this case.

Override the “equals” method in this class

This applies to booking, bookingrecord, and customer classes. This potential bad smell is observed across all entities and realms of the project, as no explicit equals methods are defined for them. However, since they extend AbstractEntity (for booking and bookingrecord) or AbstractRole (for customer), this is already handled by the framework.

Use a primitive boolean expression

Sonar's Lint suggested changing the expression !booking.getDraftMode() or booking.getDraftMode() from a Boolean type to boolean to avoid potential null values. Since these values cannot be illegally altered, they will never be null, making the change unnecessary. This applies to BookingValidator, CustomerBookingListService and CustomerBookingPublishService.

Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation.

It was flagged the presence of empty validate methods in CustomerBookingCreateService & AdministratorRecommendationPerformService. Adding logic here would result in duplication, thus it doesn't make sense to complete the implementation or throw an Exception. As learned in other subjects, clean code doesn't require comments to be understood, so no comment was added.

Rename this package name to match the regular expression

Sonar's Lint recommended renaming the package name to match the regular expression '^[a-z\_]+(\.[a-z\_][a-z0-9\_]\*)\*$'. This was innocuous because the current names are already clear and readable, changing them would break consistency with other package names in the project and it has no impact on code functionality.

Use a primitive Boolean expression

Sonar's Lint suggested changing the values of the expressions booking.getDraftMode() and !booking.getDraftMode() from a Boolean type to boolean to avoid possible null values. Since these values cannot be illegally altered, they will never be null, making the change unnecessary. This code smell can be seen at the CustomerBookingPublishService and CustomerBookingListService services.

# CONCLUSIONS

Sonar's Lint turned out to be a really helpful and powerful tool for spotting potential code issues in our work. It let us find risks without needing to run Eclipse's coverage, compile the whole project, or even re-run our tests. However, it is not perfect, since it analyzes code based on common mistakes and patterns, so it won't catch everything. But having the option to see these potential problems *as we type* makes it incredibly useful. It's definitely a tool we'll keep using as we continue to develop.

# BIBLIOGRAPHY

Intentionally blank.