DP2 2023-2024

Lint Report

Acme Software Factory

Repository: https://github.com/rafcasceb/Acme-SF-D04

Members:

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

GROUP C1.049

Version 1.0

25-05-24

Content Table

Abstract

This report offers a detailed examination of the lint analysis conducted for the individual deliverable D03 and D04 pertaining the group requirements of the project. It presents the evaluation of "bad smells" reported by Sonar Lint, and analyzing their importance.

Revision Table

Date	Version	Description of the changes	Deliverable
25/05/2024	V1.1	• Content written.	4

Introduction

In these two phases of the deliveries, the focus was directed towards the implementation of different features, creating multiple bad smells in the process of doing so. The subsequent analysis in this document pertains solely to the lint evaluation of the work done by the student 3, which revolves in the *developer* features. It was done through the analysis of all files modified during the duration of this delivery for both obligatory and supplementary requirements.

Contents

Bad Smells

The following bad smells appeared several times in the individual files analyzed for this report, belonging to all the features of the *developer* role. For the sake of simplicity, they have been consolidated into a singular enumeration.

1. Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
   As we follow the structure recommendations given by the Acme Jobs example project, this bad smell can be ignored.

2. Replace this assert with a proper check.
   As we follow the Acme Framework recommendations given in the theory lectures, this bad smell can also be ignored.

3. Define a constant instead of duplicating this literal “X” times.
   This bad smell is innocuous by nature, as the duplication of literals does not directly affect the implementation of the features. It could potentially pose a refactoring problem, but as the recommendations and examples given by the subject also duplicate literals, we can safely ignore this.

4. Refactor this method to reduce its Cognitive Complexity from X to the Y allowed.
   This is a typical bad smell found in several software projects. However, as though as the complexity seems high, the code is organized so that is easily read, so we can say that it is not important.

Conclusions

In conclusion, this lint report document allows us to examine the obstacles faced in analyzing and implementing diverse requirements, as well as a revision of our code to avoid future maintainability problems.

Bibliography

Intentionally blank.