Lawrence, Sean

10/28/2023

CSULB Fall 2023

CECS 547: Software Maintenance, Re-engineering and Reuse

Lab – Code Smells

When I first forked the repository from [1] into my own LabCodeSmells repository, and allowed CodeClimate to identify the issues, it gave the codebase a Maintainability grade of F. This is because of one issue of Code Smells and two issues of Duplication, which CodeClimate estimated would take three days for me to resolve, as shown in Fig. 1 below:

A screenshot of a computer

Description automatically generated

Fig. 1: Initial LabCodeSmells Repository Assessment by CodeClimate

The issues include bad code smells, coming from Code\_smells\_example.py, in which the function calculate\_total\_price() has a cognitive complexity of 7, which exceeds the allowed limit of 5. CodeClimate asked me to consider refactoring the code, as shown in Fig. 2 below:

A screenshot of a computer

Description automatically generated

Fig. 2: Excessive Cognitive Complexity, a bad smell, of the function calculate\_total\_price() in Code\_smells\_example.py. CodeClimate advised me to refactor the code, which I have and will show you soon.

Another big issue is duplicated, unnecessary and wasteful code. CodeClimate found that Duplicate\_code\_example.py and Duplicate\_example.py had the exact same code in the same repository, which would confuse users as to which one to use. On top of that, there are duplicate functions that are not needed, such as the function calculate\_area\_perimeter() when the calculate\_area() and calculate\_perimeter() functions already calculate these measurements separately when users need them. The expression (length \* width) is calculated multiple times throughout the file, when it is only necessary to calculate it once and store it in a variable for later use. These duplicated codes are shown in Fig. 3 below:

A screenshot of a computer

Description automatically generated

Fig. 3: Duplicate Code files, functions and calculations, that must be deleted for the program to work more efficiently.

Therefore, all these issues must be resolved, for code to be more efficient and useful.

save the (length \* width) result in an “area” variable, and just use the variable where it is needed elsewhere instead of doing this multiplication again. This is a solution that I used when I refactored this code,

Works Cited

[1] Waseemk9 (2023) Example [Source code]. <https://github.com/Waseemk9/Example.git>.