**BCPR301 –** Refectoring

Hasitha Dulanjana Palihenage Don

Link :<https://github.com/hasithadulanjana/assignmnt-2-Python>

**Name:**

Replace Nested Conditional with Guard Clauses

**Location:**

**C**ommand.py

**Problem:**

There are huge numbers of if else, statements repeating in methods, so this makes it has to read hard and indentation makes to see the code long. It is difficult to figure out what each conditional does and how, since the "normal" flow of code execution is not immediately obvious. These conditionals indicate helter-skelter evolution, with each condition added as a stopgap measure without any thought paid to optimizing the overall structure.

**Strategy:**

By isolating the special cases into separate conditions that immediately end execution and return a null value if the guard clauses are true. As well as getting rid, the code of side effects may be helpful for the purpose as the first step we can isolate all guard clauses that lead to calling an exception or immediate return of a value from the method. Place these conditions at the beginning of the method.

**Name:**

Duplicated code

**Location:**

Validator.py,

**Problem:**

Most of the time the same methods are over written to do the same functions. For example, each check method is different from one method to another only by the parameters, which are pass to the Rule check method and the class field method. Because of this scenario, this makes the all class bigger than it actually requires. As well as if a change is made this will affect all the code and it will be lead to re write all the methods. In addition, it is the good practises to keep the code dry “Don’t repeat yourself”.

**Strategy:**

As the first strategy, I will be using [Extract Method](https://refactoring.guru/extract-method)s, followed by [Pull up Field](https://refactoring.guru/pull-up-field) for the fields used in the method that you are pulling up. As well as if the duplicate code is inside a constructor, use [Pull up Constructor Body](https://refactoring.guru/pull-up-constructor-body). The code similar but not identical, use Form Template Method. If two methods do the same thing but use different algorithms, select the best algorithm and apply [Substitute Algorithm](https://refactoring.guru/substitute-algorithm).

**Name:**

Switch statement

**Location:**

File\_handler.py in the open method

**Problem:**

using switch and case operators is one of the symbols of object-oriented code. This is often code for a single switch can be scattered in different places in the program. When a new condition is added, you have to find all the switch code and modify. it.If/elif/elif statement is like a switch in python. It may also needed to extend in future for more file types so could potentially have even more ifs. I can be varies between cases is input string and which method to call

**Strategy:**

Isolating the switch class and putting it to a right class, you may need to extract methods and use the move method. Using a dictionary of file extensions to file reader subclasses in the file handler class file is opened based on which subclass, choosing the subclass from the dictionary based on file extensions

**Name:**

Shotgun Surgery

**Location:**

View.py class and command class

**Problem:**

This is becomes a hazard when small modifications requires you to do many small changes to different classes. Therefore, the single responsibility has been split up among a large number of classes. This can happen after overzealous application of Divergent Change**.**

**Strategy:**

As the first strategy, I will be using move methods and move field to move existing class behaviours into a single class. If there is no class, appropriate for this, creating a new one? If the move class is same as the old leaving original class almost empty, try to get rid of these now-redundant classes via [Inline Class](https://refactoring.guru/inline-class).

**Refactoring**

**Switch statement**

**Why it's the worst:**

The file types that used in this program is “csv, xlxs, and txt” if the program used something else it needs to throw error and data cannot be insert to the database here. Switch statement selects the file type by going thru individually using the if else lop to check the file type.

**Testing**

Test cases are run too sure whether the methods select the final format or else if it’s wrong it will throw an error saying wrong convention selected. There are several test case, which validate the files and its type.

**Version Control**

All the tests are updated in the GitHub under the file name called refactoring tests

**Evaluation**

After evaluating the current code, it was easier to read and understand which extends the duties of single responsibility. The bad day scenario is if a new business rules adopt the company needs to change the whole coding.

**Duplicate code**

**Why it's the worst:**

This is a bad principle that a programmer can inherit, It is sometimes hard to determine if the similar looking code is actually identical, or just nearly the same [and if it's actually supposed to be the same or not]. It adds to the code-space used in the executable, and thus makes the executable larger. This is one of those things that is difficult to balance - larger code MAY run faster because it's "inline", meaning that there is no jump to another function - or it may run slower because the caches are more used.

**Testing**

All the test cases that in the validator are updated to check whether code works or not.

**Version Control**

All the tests are updated in the GitHub under the file name called refactoring tests

**Evaluation**

Classes are not part of a hierarchy; use [Extract Superclass](https://refactoring.guru/extract-superclass) in order to create a single superclass for these classes that maintains all the previous functionality. If it is difficult or impossible to create a superclass, use [Extract Class](https://refactoring.guru/extract-class) in one class and use the new component in the other.by following above process the following advantages can be gained merging duplicate code and its simples the structure of the code and make it shorter. It will be easy to simple plus shortness so it can be easily supported.

**Short gun**

**Why it's the worst:**

Shotgun surgery is bit like the divergent change but is the opposite. When every time change occurs, you have to make many little changes to many different classes. When the changes are all over the place, they are hard to find, and it is easy to miss an important change. So to over come the scenario need to use Move Method and Move Field to put all the changes into a single class. If no current class looks like a good candidate, create one. Often you can use Inline Class to bring a whole bunch of behaviour together. You get a small dose of divergent change, but you can easily deal with that. Either way you want to arrange things so that, ideally, there is a one-to-one link between common changes and classes.

**Testing**

All the test cases that in the validator are updated to check whether code works or not. For this to I am using the test\_validator so they have been re-used for the refactoring tests. I am trying toprove the methods of method of the validator before and after I can ensure that the refactoring does not break it's functionality

**Version Control**

All the tests are updated in the GitHub under the file name called refactoring tests

**Evaluation**

After changing the validator there wasn’t a much to change, but other than the rule check which comprise with the business rules.While changing the Validator's checking methods to use the RuleChecker, I had to change one line many times, due to the bad smell of duplicate code,