Kieran Abelen

Sidder19@hotmail.com

Assignment 2

Coder and Tester refactoring

Contents

[Repository link: 2](#_Toc19946827)

[Refactoring 2](#_Toc19946828)

[Introduction 2](#_Toc19946829)

[Smell detection 2](#_Toc19946830)

[References 3](#_Toc19946831)

Coder – Kieran Abelen

# Repository link:

Original: (Marcus)  
Branch repository: (Kieran)

# TIGr Solution

## Brief

There is a lot of duplicate code present since there are 3 parser subclasses and 3 drawer subclasses. Some of this duplicate code is also bad smells so eliminating the duplicate code could solve more than one bad smell.  
An example of a refactor would be 2 drawers were Extracted into one 1 drawer and this method also solved another bad smell Long method. 2 bad smells both used Extract Method.

Smell detection

### Duplication Code/Shotgun Surgery

Both don’t really describe a situation where the entire class is a duplication.

#### Location

1. DrawerKieran.py, DrawerJack.py: Of the three drawers both Tkinter drawers with the exact same functions.
2. ParserTK.py, ParserJonathonV2.py, ParserJerry.py: Three parsers are present, and all have the same functions.

Reason

1. Tkinter drawers have the same methods for drawing while the turtle drawer will use different methods. It would be better to have two drawers and not three.
2. There are differences inside the functions, but they all do the same thing and ParserJonathonV2 has the best functionality.

#### Refactoring Strategies

* Extract Method: Simplest and the most universal option
* Extract Class: Normally you extract part of the class but could be used for whole thing.
* Pull Up Method: Template pattern is the abstract parent class so adding functionality is not advised
* Form Template Method: this could work with the drawer, but it already exists with the template pattern.

Extract Class: will used on both, extracting them into a new class and delete the previous one.  
Not completely what it is used for but there in no need to a child class and there is already an template class that the functions are coming from.

### Long Method

#### Location

1. DrawerJack.py. line 44: has more than 10 line if statements
2. ParserKC.py, line 11: has a multiline if search.
3. ParserJerry.py, line 16: if statements sorting a find all from an input.

Reason

1. To many Ifs to achieve relatively little.
2. Searches the same file for different things in the if very ineffective.
3. Better but still sorts each command against a single condition.

#### Refactoring Strategies

1. Replace Temp with Query: Use simple maths to eliminate the Ifs
2. Replace Method with Method Object: You only need to search once and compare it against a lookup table

Extract Class: This method would solve both since DrawerKieran.py already solved this issue so did ParserJonothanV2

### Long Class

#### Location

ParserJonathonV2.py: The whole class except the parser part on line 31

Reason

This class is doing many things.

* Lookup table stored inside the class.
* The regularly expression sorting is also here.

The real issue is if there two things are added to, they would really make the class to big.

#### Refactoring Strategies

1. Extract Class: This will be a great solution
2. Extract Subclass: could work but there are two extractions
3. Extract Interface: This class doesn’t have any overly complex dependencies so this wont the solution.

The simplest solution is Extract class since it would just create two subclasses.  
Two static subclass, one with the lookup table and the other with regular expression filter.

Refactoring

### Priority

#### Shotgun Surgery

This is the worst since it could solve other bad smells. Some duplicate code could be better than others so implemented only best of a duplicate code would solve other bad smells.

#### Long Class

This will reduce functionality and dependency on the parser class. Not as important to fix.

#### Long Method

Low priority since it will be not only fixed in duplications code bad smell but wont impact the code much if not changed.

Version control via an online repository (Kieran)

1. Modification to remove the worst smell and PEP8 validation (2 \* N marks)
2. Testing and effectively evaluations on your refactored code in a concise fashion (N marks)

# References

Kieran. (n.d.). *mellyonz/Assignment\_2\_G2-\_Kieran-Tester.* Retrieved from github.com: https://github.com/mellyonz/Assignment\_2\_G2-\_Kieran-Tester

Marcus. (n.d.). *forestraindrip/PR301\_Assignment2: The source code for PR301 Assignment2.* Retrieved from github.com: https://github.com/forestraindrip/PR301\_Assignment2