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 are lot of duplicate classes present, 3 parser subclasses and 3 drawer subclasses. The will account for multiple bad smells  
Based off Mikes restriction of not changing the Abstract parent class, there will be a few bad smells from this like Incomplete library class.

Smell detection

### Shotgun Surgery

Lots of classes have identical functions so to change one mean to change them all.

#### 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. Changing any Tkinter draw\_line or other functions would require repeated changes in each drawer.
2. Any changes to the commands or accepted data would result in changes in all of them.

#### Refactoring Strategies

* Move Method: This would create a single class of all the methods
* Move field: Would just move the data but would keep the classes this won’t remove the shotgun.
* Inline Class: The parent class is abstract so we shouldn’t add functionality to it.

### Duplication Code

The same with the shotgun lots of classes have duplicate code

#### Location

1. DrawerKieran.py, DrawerJack.py: These two classes have duplicate functions
2. ParserTK.py, ParserJonathonV2.py, ParserJerry.py: These three class have duplicate functions and even have similar but different features like regular expressions filter but no one way of doing it.

#### *Reason*

1. Tkinter drawers have almost the exact same code with DrawerKieran.py having a smaller draw\_line
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: The abstract parent class so adding functionality is not advised.
* Form Template Method: this could work with the drawer, but it technically already exists with the Abstract classes.

### Incomplete Library Class

The main entry point to program MainTIGr.py has many functions that have not been outlined in the parent library SourceReader.

#### Location

Child, MainTIGrKieran.py: MainTIGr class  
Parent, TIGr.py: Only the AbstractSourceReader class

#### *Reason*

The Interface global and Config are not variables in the parent class and have no proper method

#### Refactoring Strategies

1. Introduce Foreign Method: Add the functionality to the parent
2. Introduce Local Extension: Add a class in the middle to decorate the library with the extra functionality

### Refused Bequest

MainTIGr.py does not even utilize all the parent variables and opts to just access the parser to pipe information.

#### Location

Child, MainTIGrKieran.py: MainTIGr class  
Parent, TIGr.py: Only the AbstractSourceReader class

#### *Reason*

The child class does not use, file\_name and source. Instead ops for just calling the parse variable.

#### Refactoring Strategies

1. Replace Inheritance with Delegation: change the dependency and elect delegation and not inheritance.
2. Extract Superclass (Refactoring Guru): If inheritance is appropriate, get rid of unneeded fields and methods in the subclass

### 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 bad smells since it could affect other refactors and impacts any changes on the parser and the drawer subclasses. Since the parser is duplicated 3 times as well as the drawer any changes mean a shotgun of changes to each class.

#### Duplicate Code

Same as shotgun but a solution does not necessarily result in the removal of classes just extracted duplicate methods into a shared class. It still affects the 3 parsers and 2 drawers since the DrawerTurtleJack.py drawer doesn’t have duplicate code just functions.

#### Incomplete Library Class

This library class doesn’t even need to exist in its current form since it is only used to store parser. It must be updated with extra functionality. Very important to retain the justification of the Abstract classes. This only affects MainTIGr.py and TIGr.py, library or parent class.

#### Refused Bequest

MainTIGr.py subclass is not using 2 out of the 3 variables present in the parent. They should be moved to a subclass that does need it. This is average priority to remove the redundant variables since they are not used in MainTIGr.py so this bad smell only effects TIGr.py

#### Long Class

This will reduce functionality and dependency on the parser class. Not as important to fix as the first 4 since it only affects ParserJonothanV2.py not any parent classes.

#### Long Method

Very Low priority since it will be not only fixed in Shotgun/duplications code bad smell but wont impact the code much if not changed. Since you could opt not to use PaserKC.py and DrawerJack.py.

Version control via an online repository (Kieran)

### 1st Refactor: Shotgun Surgery

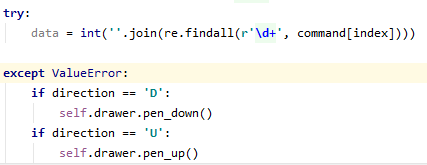
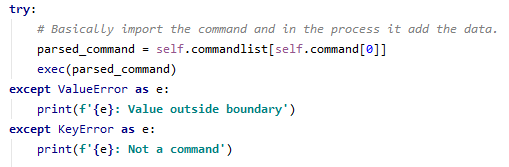
Fixes both Duplicate code and Long Method too.

#### Strategy Used: Move Method

Since the are no dependencies inside the classes outside using the abstract parent class, the move method would be the best solution.

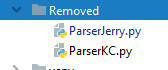
#### Refactor

1. Move Method into ParserJonathonV2.py
   1. The only thing moved was the Try and Except from ParserJerry.py

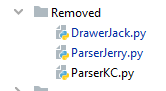
 

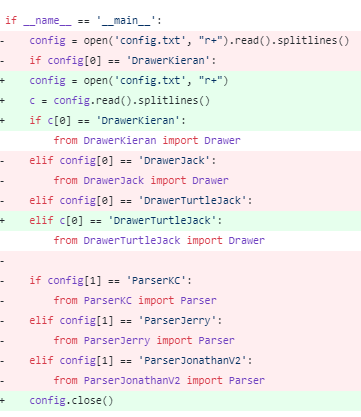
ParserJerry.py Moved -> ParserJonathonV2.py

1. Remove the other two parser classes: No longer needed



1. Remove duplicate DrawerJack.py: Since it is a worse DrawerKieran.py Tkinter drawer.  
   No Move method required.



1. Remove the classes being imported  
     
   MainTIGrKieran.py

#### Effectively evaluation

This not only removed Shotgun Surgery but also removed Duplicate Code and all the Long methods

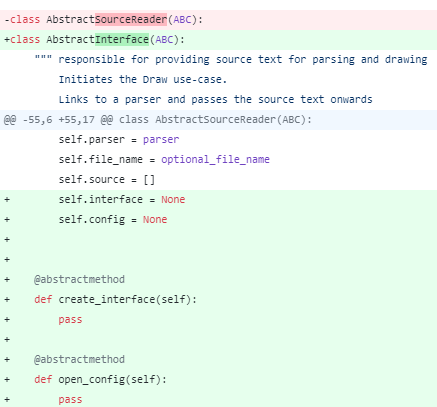
### 2nd Refactor: Incomplete Library Class

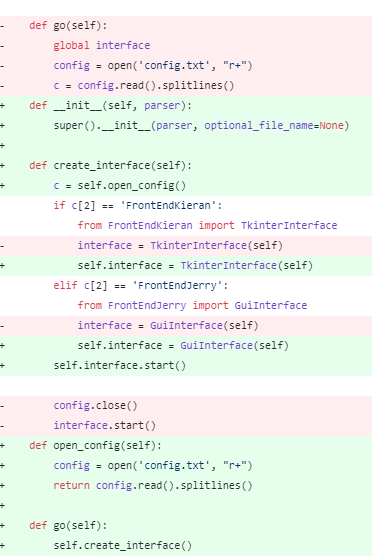
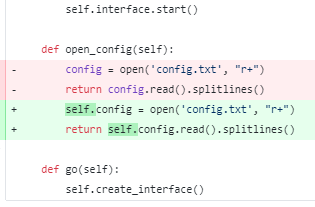
#### Strategy Used: Introduce Foreign Method

The library I am using is an Abstractions so there is no reason I can’t modify it to add the extra functionality, in this case it is just the Interface and config variable/method.

#### Refactor

1. Change parent or library class

  
TIGr.py

1. Change child class to use the new functions  
   MainTIGrKieran.py
2. config to self.config  
   

#### Effectively evaluation

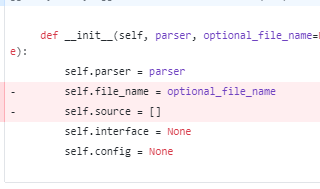
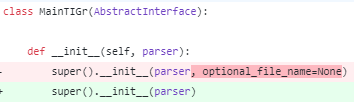
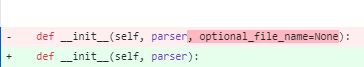
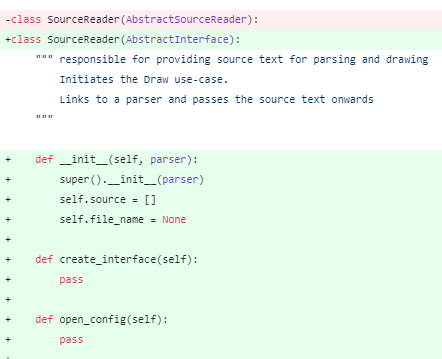
This if an effective addition since it added the extra functionality, I have created by reading a config and initializing and interface into the parent class

### 3rd Refactor: Refused Bequest

#### Strategy Used: Extract Superclass

This will be used to remove the unused variables in the parent or inherited class.

#### Refactor

1. Remove unused Abstract variables  
   
2. Remove parameter use  
     
     
   TIGr.py
3. Extracted superclass is WordyTIGr.py but will not implement all new features  
   

#### Effectively evaluation

This removed the unnecessary variables without changing functionality of the code.  
The original WordyTIGr.py is the new home of the extracted variables but its functions are obsolete.

# 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