# Advance Programming Assessment 2 - Refactoring

## Smell Detection

Smell # 1:

Name: Large Class

Location: <package:refactoring> - <interpreter.py> - <class:Interpreter>

Reasons:

1. The class is doing multiple things at the same time
2. Reduced readability
3. Finding which feature was/is broken will take time the larger the class becomes

Strategies / Approaches:

1. Extract class
   1. Creating a <Loader> class
      1. Relocating the <read\_file> method from <Interpreter> to <Loader>

Smell # 2:

Name: Long Method

Location:

<package:refactoring> - <interpreter.py> - <class:Interpreter> <method:get\_details()>

<interpreter.py> - <class:Interpreter> <method:data\_parser()>

<class\_grabberr.py> - <class:ClassGrabber> <method:add\_under()>

Reasons:

The more functions are added to a method, the more complex it becomes, at the same time it also becomes larger.  
As the method grows, explaining what it does also becomes harder, as it does more than what was initially planned for the method to actually do.

As the method grows, aside from the **Long Method** bad smell, this also leads to another bad smell which is the **Large Class**.

Strategies / Approaches:

* Extract Method
  + Creating another method to break down the stated Long Methods.

Smell # 3:

Name: Duplication

Location:

<package:refactoring> - <class\_grabber.py> - <class:ClassGrabber> <method:get\_attrib/method()>

Reasons:

2 methods doing essentially the same thing, creating unnecessary lines of codes. That is, appending string values inside 2 different string arrays.

Strategies / Approaches:

* Extract Method:
  + Create a singular method based on the duplicating methods, adding another parameter for the array in which a input string will be appended to.

Smell # 4:

Name: Switch/If Statements

Location:

<package:refactoring> - <class\_grabber.py> - <class:ClassGrabber> < method:add\_under ()>

Line 44 - 55

<loader.py> - <class:Loader> <method:load\_file()>

Line 16 - 24

<interpreter.py> - <class:Interpreter> <method:data\_parser()>

Line 50 – 67

<interpreter.py> - <class:Interpreter> <method:get\_details()>

108 - 136

Reasons:

Hard coded conditions that only really help a couple of problems but requires multiple lines of codes.

Strategies / Approaches:

* Extract Method:
  + Create a singular method based on the duplicating methods, adding another parameter for the array in which a input string will be appended to.

## Refactoring

Worst Smells (from top to bottom)

**Switch/If Statements**

Why:

Because this bad smell causes most of the other bad smells that has been stated. For example, because of multiple **If** statements in the <get\_details()> method of the <Interpreter> class, the method grew large, way too large for a method that essentially should only be doing one thing which is to grab data and return an array.

This situation can also be found on different methods in different class.

**Large Method**

Why:

Because this bad smell leads to the next bad smell in this list with is **Large class.**

I’ve also made this one my 2nd worst bad smell because this bad smell is found on multiple methods, and while said methods runs fine as it is, refactoring them will make it easier for the next coder to understand what they do

**Large Class**

Why:

Because having a class that does multiple jobs at the same time will make it harder to understand the purpose of why the class is created. And refactoring this classes will make the specifications of functionality a lot clearer and more precise.

## Version Control

Version 2

* Made a <Loader> class to do the reading of the file input rather that <Interpreter> class doing it.
* Cleaned up the <read\_file/load\_file> if statements by creating 2 new methods located in the <Loader> class

Version 3

* Refactored <Interpreter.getDetails()
* Added <Interpreter.attrib\_type()> which does the long <if> statement in the refactored method that does the calculation on what type of attribute is being passed on the given class