# Smell Detection

## Bad Smell 1

### **Name:** Large Class

### **Location:** PlantUML/module\_builder/class\_builder.py/ClassBuilder/ (between lines 6-77)

### **Reasons why it is bad:**

1. It is quite a large class (77 lines of code)
2. It has long methods, where some methods are repeated. This makes it harder to extend for further functionality as it is performing more than one task, which makes it difficult to understand e.g. the print\_class method (lines 60-76) is performing more than one task and repeating some
3. There are lots of methods doing too many things, therefore, it is harder to debug and extend e.g. the print\_class method (lines 60-76)
4. It is hard to read as the class is so long you don’t know what method relates to what section

### **Strategies:**

1. Figure out what the ClassBuilder class and its methods should do
2. Remove any unnecessary code to reduce the class size e.g. get rid of long methods, change attributes, remove long ‘if’ statements
3. Use the Move Attribute and Move Methods to relate methods in the new sub-classes
4. Split up the class into new classes with single responsibilities so methods are only performing one task
5. Put the new classes/code into new files for readability and functionality

## Bad Smell 2

### **Name:** Long Method

### **Location 1:** PlantUML/module\_builder/interpreter.py/Interpreter/find\_relationship/ (between lines 45-60)

### **Location 2:** PlantUML/module\_builder/interpreter.py/Interpreter/find\_classes/ (between lines 67-83)

### **Location 3:** PlantUML/module\_builder/class\_builder.py/ClassBuilder/print\_class (between lines 60-76)

### **Reasons why it is bad:**

1. The methods are performing more than one task. E.g. in the ClassBuilder class the print\_class method (lines 60-76) is performing many tasks such as getting the length of the parent classes and composite classes. It is also returning outputs and creating many different new lines.
2. The methods are longer than ten lines. According to the bad smell standards, this is not ideal, and it will make you start to ask questions about the code.
3. There are multiple if statements in e.g. the Interpreter class of find\_classes (lines 67-83) and find\_relationship (lines 45-60). This makes it harder to understand and read.
4. It is harder to debug as the methods are doing too many things e.g. find\_classes method (lines 67-83)

### **Strategies:**

I will use the Extract Method for refactoring because the methods are performing more than one task, longer than ten lines and has multiple if statements. This method will get rid of this bad smell and make it easier to find out what the methods actually do.

1. Figure out what the methods should actually do and name it in a way that is purposeful
2. Copy relevant code fragments to new methods, and make sure they are performing one task
3. I will delete the fragment from its old location and put it each into a new method and replace the instances of it with calls to the new method
4. I will pass variables to the parameters of the new method in order to use the values previously contained in them. This is only if the variables are declared prior to the code that I am extracting
5. Check that the methods I have changed are no longer than ten lines

## Bad Smell 3

### **Name:** Switch Statements

### **Location 1:** PlantUML/module\_builder/method.py/Method/find\_type/ (between lines 10-21)

### **Location 2:** PlantUML/module\_builder/attribute.py/Attribute/find\_type/ (between lines 21-32)

### **Location 3:** PlantUML/module\_builder/attribute.py/Attribute/\_\_str\_\_/ (between lines 9-19)

### **Location 4:** PlantUML/module\_builder/class\_builder/ClassBuilder/add\_relationship/ (between lines 38-48)

### **Reasons why it is bad:**

1. It is hard to understand as there are multiple switch statements which are complicated e.g. the add\_relationship method in the ClassBuilder class (lines 38-48)
2. It makes the methods and classes longer than they should be e.g. the ClassBuilder class mentioned in the first bad smell
3. It is harder to read and more confusing with many statements as it is harder to tell what is happening to the data e.g. the find\_type method in the Method class (lines 10-21)
4. It is harder to debug the methods if there are many switch statements and it is doing many things at once e.g. in all the locations mentioned above

### **Strategies:**

1. Remove any duplication found in the code
2. Create dictionaries and use it to store the variables and methods instead of using switch statements
3. Check for any duplication in the code from refactoring and then remove this duplication
4. Double check and re-assess the code for duplication/long switch statements

# Tests Development

## Set of tests for the methods/classes/modules/packages encompassed by the bad smells created

DocTests/class\_builder\_doctest

DocTests/relationship\_finder\_doctest

DocTests/class\_finder\_doctest

DocTests/type\_finder\_doctest

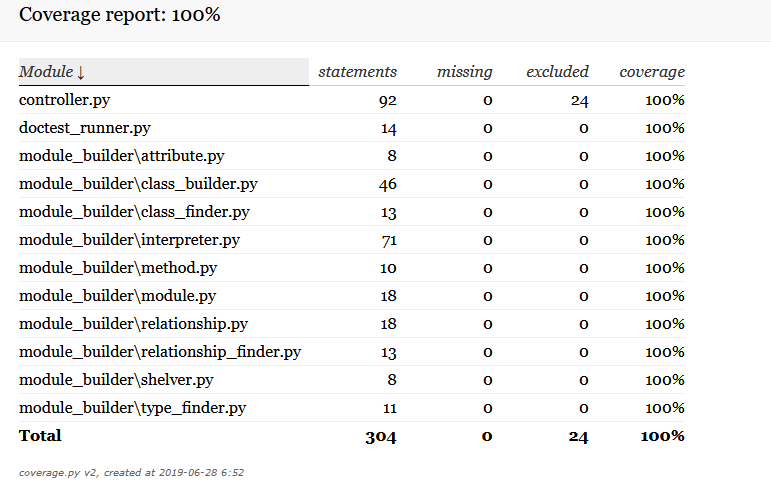
DocTests/attribute\_doctest

DocTests/method\_doctest

DocTests/module\_doctest

DocTests/relationship\_doctest

DocTests/controller\_doctest

DocTests/interpreter\_doctest  
**Coverage package of a HTML report**  


# Refactoring

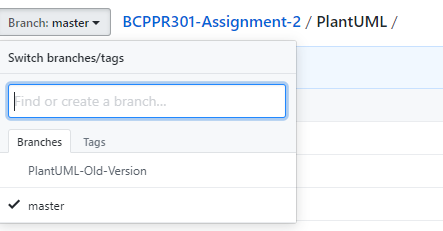
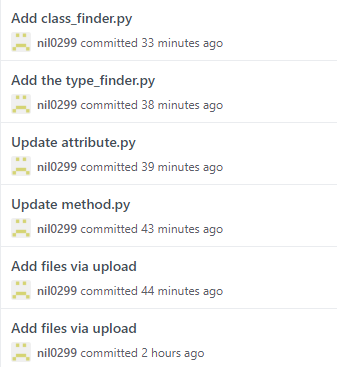
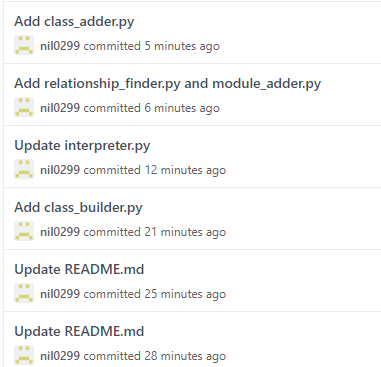
## Identification of the worst bad smell

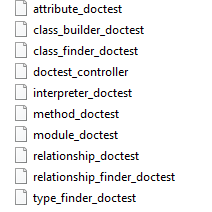
|  |  |  |
| --- | --- | --- |
| **Bad Smell** | **# Appearances** | **# Lines Affected** |
| Large Class | 1 | 77 |
| Long Method | 3 | 47 |
| Switch Statements | 4 | 148 |

The worst bad smell is the Switch Statements. This is because it makes methods longer due to many statements. When the methods are longer, it makes the classes longer as well. Therefore, with complicated and long Switch Statements, it creates other bad smells such as Large Classes and Long Methods (as mentioned above) and also makes it harder to understand and read.

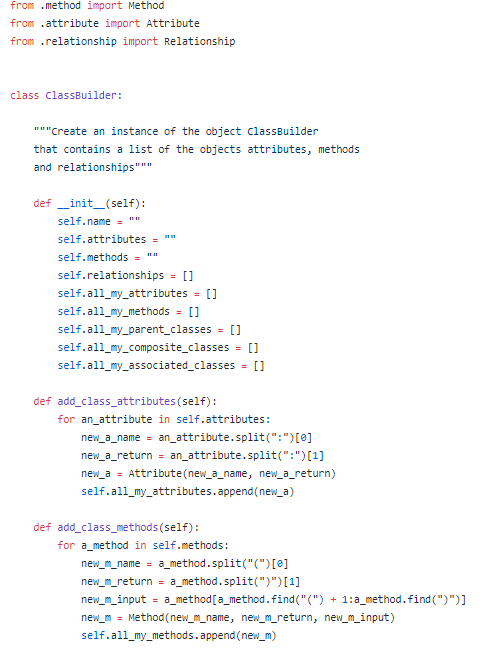
To figure out how the Switch Statements were the worst smell, I calculated how many times it appeared throughout the code. It appeared the most times (4 times) and affected the most amount of lines of code (148 lines) from the table above, therefore creating more bad smells and affecting other aspects of code. This worst bad smell was removed first (to make it easier to remove the other bad smells), followed by the Long Methods and then Large Classes.

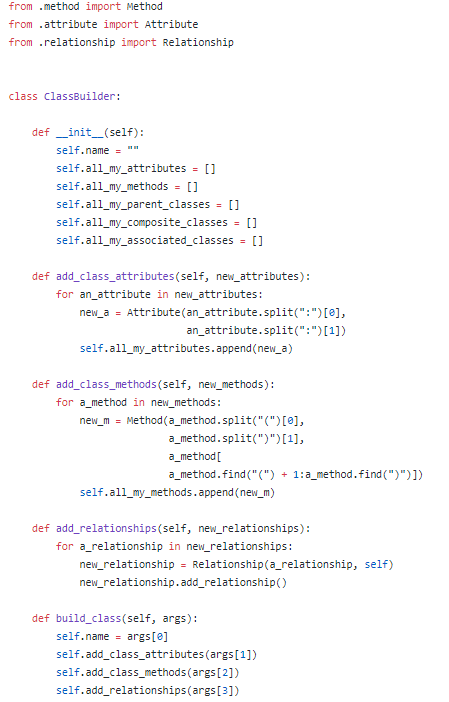
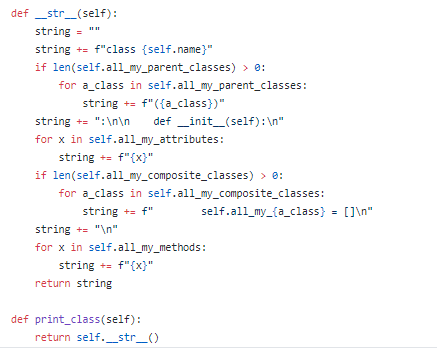
## Version control and testing

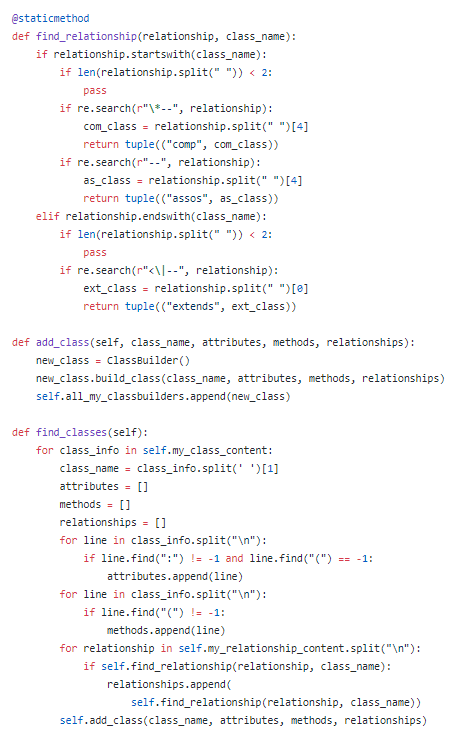


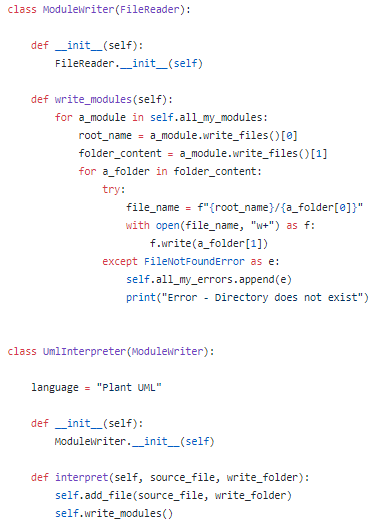
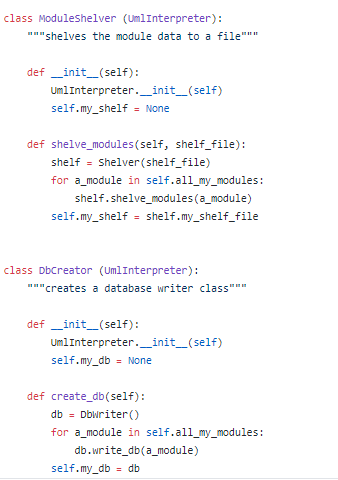
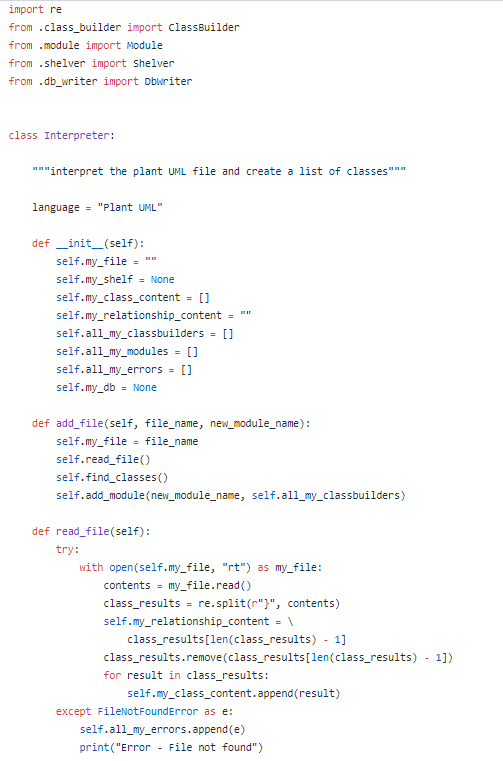
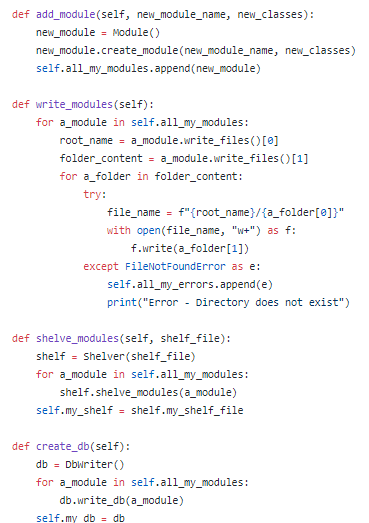


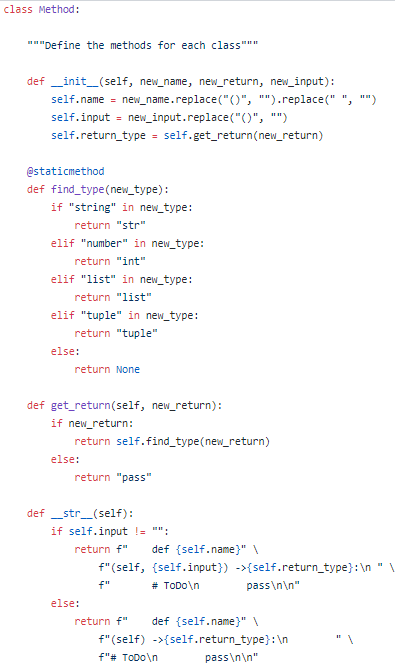
## Implementation and pep8

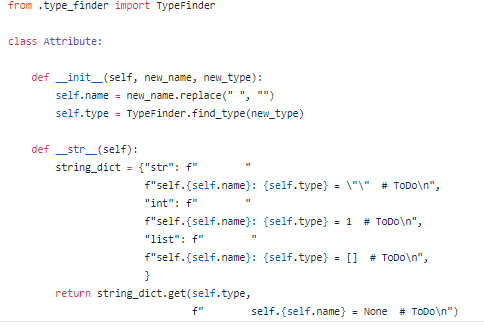
Large class, switch statements and long method before of class\_builder.py

 Large class, switch statements and long method after of class\_builder.py

Long method before of interpreter.py

Long method of interpreter.py after

Switch statements of method.py beforehand and after:

Switch statements of attribute.py before and after:



## Effective evaluations

The evaluations have been done in order. The worst bad smell is done first, then the second etc.

### Switch Statements

This worst bad smell was removed as the switch statements have been put into their own classes and methods, which gives it their own purpose. This has made the code more readable and understandable due to it being in its own classes. The only problem is that it might have made another bad smell such as longer methods by adding more classes and methods. It has also removed complexity by having each class and method doing a separate group of their own functions.

The attribute.py/\_str\_ methods replaced ‘if’ and ‘elif’ statements with a dictionary, which makes it easier to read.

The class\_builder.py/add\_relationship method replace switch statements with shorter functionality and remove the ‘if’ statements.

The find\_type methods were put into a separate file/class as it didn’t need to be removed completely.

### Long Method

This next bad smell was removed as each method/function was split into smaller classes. It is now easier to read and understand, therefore improving the code quality. Classes such as FileReader, ModuleWriter, UmlInterpreter, ModuleShelver and DbCreator have been made in the interpretor.py file.

The Extract Method helped for refactoring because the methods are now not performing more than one task, it isn’t longer than ten lines and doesn’t have has multiple if statements. This method will got rid of this bad smell and made it easier to find out what the methods actually do.

### Large Class

The next last bad smell large classes were easy to refactor because I just extracted them into smaller classes and other files, reduced the long methods and got rid of excessive switch/’if’ statements. It has been shrunk by about 21 lines of code. The class is still large, but it has still been reduced. Therefore, it has also made it easier to read and de-bug.

The Extract Method helped for refactoring because the methods are now not performing more than one task, it isn’t longer than ten lines and doesn’t have has multiple if statements. This method will got rid of this bad smell and made it easier to find out what the methods actually do.

# Self-Marking

|  |  |
| --- | --- |
| Name of bad smells | 3/3 |
| Locations | 3/3 |
| Reasons why it’s bad | 3/3 |
| Refactoring approaches | 3/3 |
| Acceptable tests | 9/9 |
| Coverage | 3/3 |
| Worst smell and why | 3/3 |
| Version control | 3/3 |
| Implementation and pep8 | 6/6 |
| Evaluation | 3/3 |
| **Total** | **40** |