# Smell Detection

## Name :Long Method

### Location :

<model><class\_finder><ClassFinder>

<find\_class><between line 9 and 51>

<relationship\_finder><between 57 and 74>

<model><pep8\_converter><PEP8Converter>

<create\_class><between line 37 and 63>

<convert\_method><between 65and 86>

<convert\_constructor><between 88 and 107>

### Reasons:

1. This bad smell is the most prevalent in the program affecting 125 lines of code this means it has the most impact on the code quality.
2. Long methods make the code very hard to read and understand what the methods are doing.
3. Long methods can mean that methods have more than one responsibility which is bad programming practice.
4. Long methods make debugging more difficult due to unreadability

### Strategies

Extract Method will be used to create multiple smaller methods out of the long methods. Each method will have only a single resposibility.

### Test Development

Unit tests will be created for ClassFinder and PEP8Converter classes.

### Refactoring

1. Worst bad smell is long method because it affects the most lines of code and it causes the code to be very hard to read
2. Version control 11 commits, all tests pass before each commit

### Evaluation

The long methods have been effectivly removed they are now made up of many smaller methods. This is effective because the code is now much easier to read and the new methods are short and only have a single responsibility.

## Name : Switch Statements

### Location :

<controller><interpreter\_controller><InterpreterController>

<start\_menu><between 20 to 130>

### Reasons:

1. Complex series of if else statements make the code very hard to read and understand
2. The code becomes harder to maintain
3. The code becomes very difficult to make changes to

### Test Development

Unit tests will be created for InterpreterController.

### Refactoring

1. Next worse bad smell is Switch statements because it affects the next most lines of code and has the worst affect on readability and extendibility of the code
2. Version Control 5 commits for refactoring

### Evaluation

The start\_menu method had a complex series of if else statements (10+ of them) which made the code very hard to read. After refactoring start\_menu has only one if and else statement and the code is much easier to understand. The if statements were removed by creating a dict of possible user inputs and methods. After refactoring the code is not only more readable but is also easier to extend, a new option could easily be added to the dict.

## Name : Duplication

### Location :

<controller><interpreter\_controller><InterpreterController>

<load\_pickled\_file><between 35 to 48>

<load\_text\_file><between 55 to 59>

<write\_file\_to\_code><between65 to 74>

<write\_file\_to\_database><between 79 to 84>

<print\_to\_screen><between 97 to 106>

<load\_text\_file><between 55 to 59>

<pickle\_file><between 110 to 121>

### Reasons

1. After previous refactoring this bad smell affects the most lines of code.
2. There is duplication at many points in the code mainly in regard to error messages. This makes the code much longer than necessary and harder to read.
3. Because the code is longer, and the same thing is being done multiple times there is more chance of bugs being introduced.
4. If changes are to be made, then they must be made wherever the duplicated code exists.

### Strategies

Use Extract Method to extract the duplicated code that can then be called in multiple places.

### Refactoring

1. Next worse bad smell is duplication because it causes code to be longer than needed and more code means more chances to introduce errors.
2. Version Control 4 commits for refactoring

### Evaluation

The InterpreterController class had a series of duplicated code checking for the same set of errors and returning the corresponding error message. By creating a single method check\_errors the duplicated code was able to be removed and instead Mutiple calls to the new method are made this removed the duplication in this class .