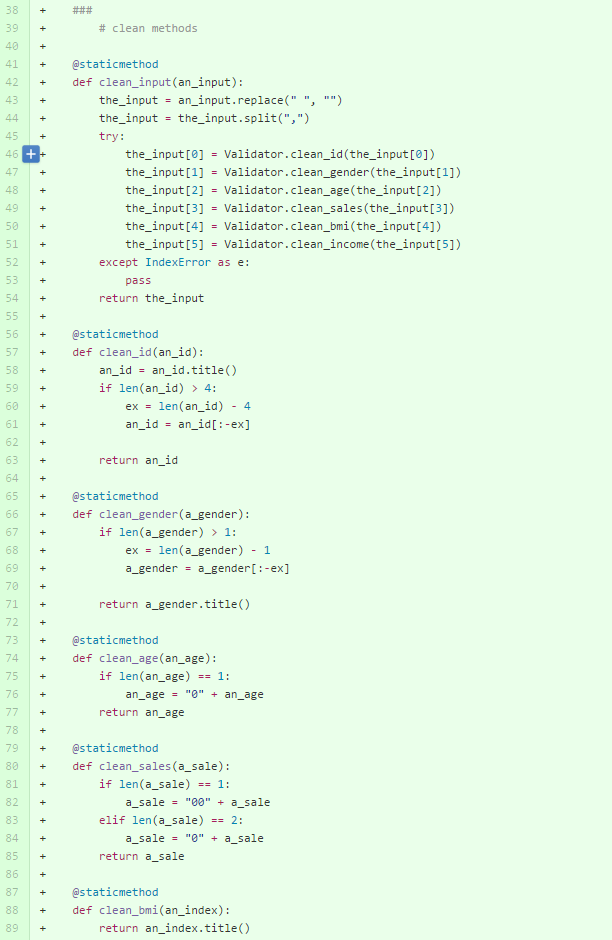
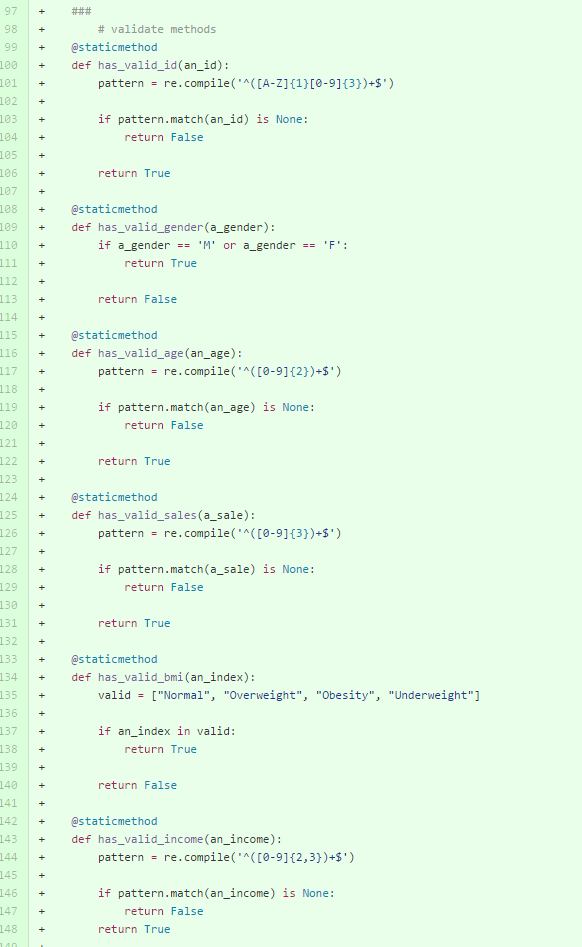
# Smell 1 evaluation:

## Issue: Large Class

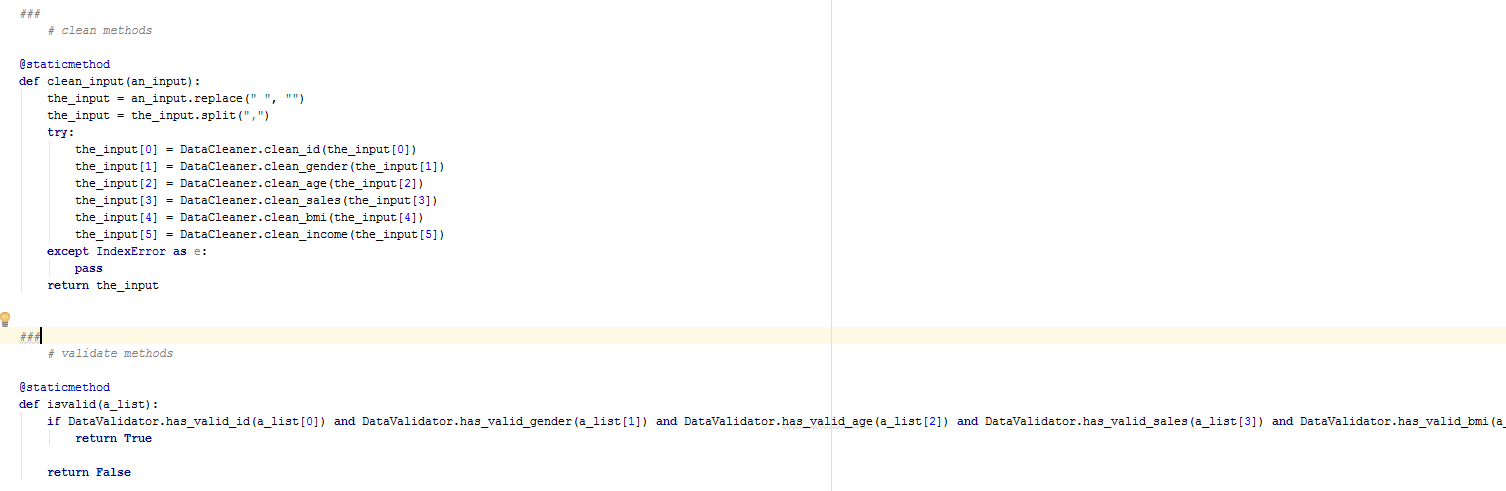
In the initial iteration, the Validator class was wearing too many hats. Its primary purpose was to take raw data, in string format, and separate the valid data from that which could not be parsed into a storable object.

However, as the application grew, the class’ responsibilities grew. The issue which had to be dealt with was that there were two groups of methods that needed to be extracted from the Validator class into separate classes. These groups were: the methods to ascertain whether data was in the correct format, and the methods to automatically parse data that was ‘close’ to being correct, so that it was in the correct regex format.

The refactoring method required was ‘Extract Class’ – to remove the data cleaning and data validation methods into separate classes, resulting in 3 single responsibility classes.



*Clean and is valid methods pre-refactoring – inside Validator class*



*Validator class accessing the static classes’ methods post-refactoring*

## Evaluation:

I think that the refactoring was successful. The interface didn’t change, as the validation and cleaning of the data accessed by the controller still goes through the Validator class. Thus code changes around the project were minimal – the only code that needed to be changed was in the Editor class, which uses the validation and cleaning functions in its validate() method.

# Smell 2 evaluation:

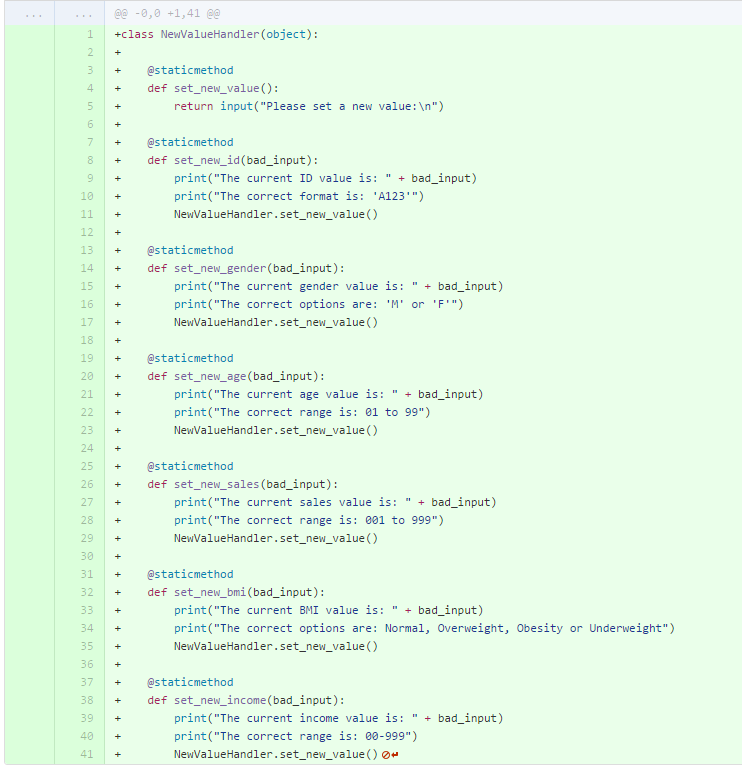
## Issue: Overly relied upon generic method, with overly complicated method parameters

The bulk of the work done when allowing users to edit bad data so that it was convertible to a storable object was done in the Editor.validate() method. One glaring issue in this method was that it overly relied upon Editor.set\_new\_value(), which was an example of a generic method making the process more complicated.

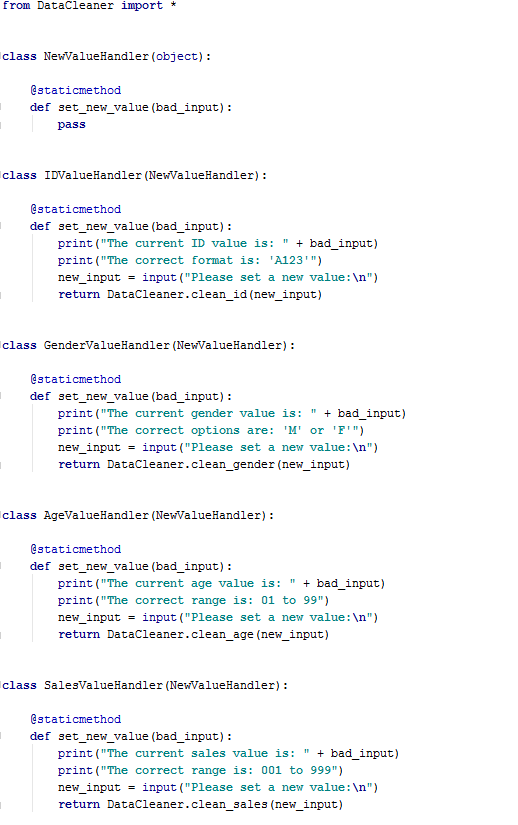
To resolve this issue, I used ‘Introduce Parameter Object’ – I removed the generic method completely, by creating a NewValueHandler static class, with a specific method for each value which may need to be edited.

## Evaluation:

After attempting the refactoring, and looking at the new code, I realised that the first iteration was not successful – it did not serve its purpose of making the validate() method more readable, nor was it efficient. After discovering this, I re-evaluated the structure of the static class. I decided to create a subclass for each editable value.



*The first attempt*



*Attempt #2*

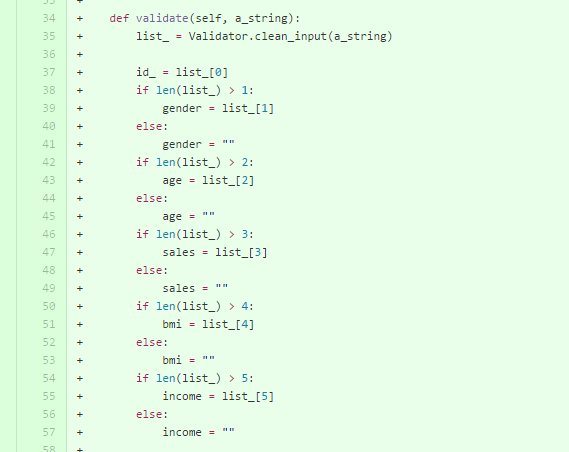
I believe that removing the generic method was the correct decision – it made the code a lot cleaner, and more easily extendable. It also served the purpose of cleaning up the Editor.validate() method.

# Smell 3 evaluation

## Issue: Complicated Conditional Expression

This bad smell was another contained within the Editor.validate() method. The purpose of the long if statement was to take a string, parse it to a list, and assign a local variable to each section of the list.

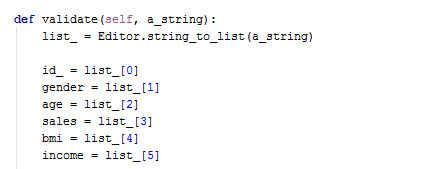
The issue was that the implementation was difficult to read, and overly complicated. To remedy it, I attempted to Consolidate the Conditional Expression, in a more pythonic manner.

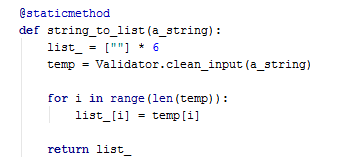


*The original conditional*

## Evaluation:

I believe that this refactoring was successful. The new method is a lot more readable, and easier to be extended. I also extracted the conditional statement into its own method, as the Editor.validate() method was wearing too many hats.





*The refactored methods*