# Bad Smells Identified, Categorized and Methods for removing

The following is a list of bad smells identified in the code. They are listed in order of the seriousness they present. The reasons why they have been categorized as such are stated along with them. The first “Bad Smells” were found during test writing. These were “dead code” that has already been remove and not mentioned here.

## Most Serious Smell

Name: Long Method

Location: Folder/File: FileManagement/filehandler.py

Class: FileHandler

Method: validate(self, data)

Lines: 50-147

Reasons: 1) Incredible Long

2) Most serious smell because it is an important method

3) Most serious smell because length makes it hard to understand

### Strategies/Approach:

Use Extract Method on each stage of validation for example below at line 71

*# check the format is a letter and 3 digit e.g A002 or a002***if** re.match(**r'[a-z][0-9]{2}'**, person[0].lower()):  
 *# Kris* **if** len(str(person[0])) >= 5:  
 self.valid = **False  
else**:  
 *# Kris* feedback += **"ID is incorrect; must contain a letter and 3 digits e.g. a001.\n"** self.valid = **False**

Place inside method validate\_format(person, feedback)

*# check the format is a letter and 3 digit e.g A002 or a002*self.valid = self.validate\_format(person, feedback)

**def** validate\_format(self, person, feedback):  
 validationcondition = **True** *# check the format is a letter and 3 digit e.g A002 or a002* **if** re.match(**r'[a-z][0-9]{2}'**, person[0].lower()):  
 *# Kris* **if** len(str(person[0])) >= 5:  
 validationcondition = **False  
 else**:  
 *# Kris* feedback += **"ID is incorrect; must contain a letter and 3 digits e.g. a001.\n"** validationcondition = **False  
 return** validationcondition

### Effectiveness

The process describe above was repeated to extract methods for all other validation requirements which reduced the length of the method from 97 lines to 38 which resulted in far more readable and understandable code

## Second Most Serious Smell

Name: Divergent Change/Speculative Generality

\*During the course of removing this smell the added benefit of removing /Long Method and Tempory Fields was achieved

Location: Folder/File: /graph.py

Class: Graph

Method: build\_graph(self, args)

Lines: 50-147

Reasons: 1) Must change the method in multiple places to alter one thing

2) In violation of the single responsibility principle, it may build graphs but each graph is a different responsibility and requires different code.

3) Speculative generality also appears in this method at line 82 and 83 of this method. The reason being unless a bar graph of a certain type is being build the lists graphdata3[] and graphdata4[] are not used.

3) It is also a very long method. Although not as important as the validator method

### Strategies/Approach:

Use the extract method to separate the responsibilities and eliminate all of the bad smells mentioned. IE make pie graph, make bar graph.

Began making the embedded function append\_sql() a class method. Used Extract Method to create individual method for all the sql calls for example on line 88

**if** argss[1] == **'gender'**:  
 graphdata1.append(len(append\_sql(**"""SELECT \* FROM employee WHERE gender = 'm'"""**)))  
 graphdata1.append(len(append\_sql(**"""SELECT \* FROM employee WHERE gender = 'f'"""**)))  
 labels = [**'Male'**, **'Female'**]  
 graphtitle = **"Employees by sex"**

became

**def** append(self, sql):  
 self.database.execute\_sql(sql)  
 **return** self.database.cursor.fetchall()  
  
**def** get\_gender\_data(self, data):  
 data.append(len(self.append(**"""SELECT \* FROM employee WHERE gender = 'M'"""**)))  
 data.append(len(self.append(**"""SELECT \* FROM employee WHERE gender = 'F'"""**)))

**if** argss[0] == **'pie'**:  
 **if** argss[1] == **'gender'**:  
 self.get\_gender\_data(graphdata1)  
 labels = [**'Male'**, **'Female'**]  
 graphtitle = **"Employees by sex"  
 elif** argss[1] == **'bmi'**:

The above would later introduce “LONG PARAMETER LISTS” especially with the bar graphs. For this reason the “PRESERVE WHOLE OBJECT” technique was used. And the graph was passed in and changed via reference.

**def** get\_salary\_by\_age\_data(self, data1, data2, data3, data4):  
 data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))  
 data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))  
 data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))  
 data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))

**if** argss[1] == **'salary-by-age'**:  
 self.get\_salary\_by\_age\_data(data1, data2, data3, data4)

BECAME

get\_salary\_by\_age\_data(self,g):  
 g.data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 g.data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 g.data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 g.data1.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))  
 g.data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 g.data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 g.data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 g.data2.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))  
 g.data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 g.data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 g.data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 g.data3.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))  
 g.data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary<125"""**)))  
 g.data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 126 AND 150"""**)))  
 g.data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 151 AND 175"""**)))  
 g.data4.append(len(self.attach(**"""SELECT \* FROM employee WHERE Age < 26 AND Salary BETWEEN 176 AND 200"""**)))

g.labels = [**'<25'**, **'26-40K'**, **'40-50'**, **'>50'**]  
g.title = **"Salaries by Age"**g.type = **"bar"**g.option = **"salary-by-age"**

**if** argss[1] == **'salary-by-age'**:  
 self.get\_salary\_by\_age\_data(data1, data2, data3, data4)

This would have the added benefit of removing “TEMPORY FIELDS”. For example on line 80 of the original files

graphdata1 = []  
graphdata2 = []  
graphdata3 = []  
graphdata4 = []  
labels = []  
graphtitle = **None**

Along with on…

new\_graph.data1 = graphdata1  
new\_graph.data2 = graphdata2  
new\_graph.data3 = graphdata3  
new\_graph.data4 = graphdata4  
new\_graph.title = graphtitle  
new\_graph.type = argss[0]  
new\_graph.option = argss[1]  
new\_graph.labels = labels

on line 162 of the original code

### Effectiveness

As well as vastly improving readability and maintanence the code was reduce in length from 116 line to a mere 34