Table of Contents

[1. Name: Long Method – FileView().get() [MEDIUM] 2](#_Toc482108993)

[Location: 2](#_Toc482108994)

[Reasons: 2](#_Toc482108995)

[Strategies/ approaches: 2](#_Toc482108996)

[2. Name: Feature Envy – Controller().display() [DONE] 2](#_Toc482108997)

[Location: 2](#_Toc482108998)

[Reasons: 2](#_Toc482108999)

[Strategies/ approaches: 2](#_Toc482109000)

[3. Name: Long Method – CmdView().help\_cmd() 3](#_Toc482109001)

[Location: 3](#_Toc482109002)

[Reasons: 3](#_Toc482109003)

[Strategies/ approaches: 3](#_Toc482109004)

[4. Name: Long Method – Controller().display() [HIGH] 3](#_Toc482109005)

[Location: 3](#_Toc482109006)

[Reasons: 3](#_Toc482109007)

[Strategies/ approaches: 3](#_Toc482109008)

[5. Name: Long Method – Controller().serialize() [HIGH] 4](#_Toc482109009)

[Location: 4](#_Toc482109010)

[Reasons: 4](#_Toc482109011)

[Strategies/ approaches: 4](#_Toc482109012)

[6. Name: Long Method – Validator().validate() [MEDIUM] 4](#_Toc482109013)

[Location: 4](#_Toc482109014)

[Reasons: 4](#_Toc482109015)

[Strategies/ approaches: 4](#_Toc482109016)

[7. Name: Switch Statement – CmdView().initialize() [LOW] 5](#_Toc482109017)

[Location: 5](#_Toc482109018)

[Reasons: 5](#_Toc482109019)

[Strategies/ approaches: 5](#_Toc482109020)

[8. Name: Switch Statement – Controller().display() [DONE] 5](#_Toc482109021)

[Location: 5](#_Toc482109022)

[Reasons: 5](#_Toc482109023)

[Strategies/ approaches: 5](#_Toc482109024)

[9. Name: Duplicate Code – Visualiser().display\_\*() [DONE] 6](#_Toc482109025)

[Location: 6](#_Toc482109026)

[Reasons: 6](#_Toc482109027)

[Strategies/ approaches: 6](#_Toc482109028)

[10. Name: Speculative Generality – View() [DONE] 6](#_Toc482109029)

[Location: 6](#_Toc482109030)

[Reasons: 6](#_Toc482109031)

[Strategies/ approaches: 7](#_Toc482109032)

[11. Name: Large Class – Controller() [DONE] 7](#_Toc482109033)

[Location: 7](#_Toc482109034)

[Reasons: 7](#_Toc482109035)

[Strategies/ approaches: 8](#_Toc482109036)

# Expected Marks

## Smell Detection **[4/4]**

1. 4 different types of smells: **[1/1]**

* Switch Statement
* Feature Envy
* Large Class
* Speculative Generality

1. Location of smells identified to file > class > method > line. **[1/1]**
2. Reasons why the code is a smell. **[1/1]**
3. Discussion of strategies. **[1/1]**

## Test Development **[3.5/4]**

1. I had 100% branch coverage for all of my application. **[3/3]**
2. All of my tests were run by the one file (unittests.py), **[0.5/1]**

but not separated into suites.

## Refactoring **[4.5/5]**

1. Reasons why the smell was the worst. **[0.5/1]**
2. Version control **[1/1]**
3. Removing smell **[2/2]**
4. Evaluations **[1/1]**

Total =

# **Name:** Long Method – FileView().get() [MEDIUM]

## Location:

Package: Interpreter

File: fileview.py

Class: FileView

Method: get()

Lines: 17 – 42 (25)

## Reasons:

1. …

## Strategies/ approaches:

<e.g., extract method>

# **Name:** Feature Envy – Controller().display() [DONE]

## Location:

Package: Interpreter

File: controller.py

Class: Controller

Method: display()

Lines: 17, 18

## Reasons:

1. This is the third bad smell I chose because it affects multiple modules, controller.py and validator.py.
2. It is bad because the controller needs to check if some of its data is valid. It does this by accessing the data stored in the validator and performing the comparison in the controller.
3. This is feature envy because all of the data is being brought into the controller and being used there.

## Strategies/ approaches:

1. Extract method: isValidColumn()

New method in controller

1. Extract method: isValidFlag()

New method in controller

1. Move method: isValidColumn()

Move to validator

1. Move method isValidFlag() => validator

Move to validator

# **Name:** Long Method – CmdView().help\_cmd()

## Location:

Package: Interpreter

File: cmdview.py

Class: CmdView

Method: help\_cmd()

Lines: 14 – 22 (8)

## Reasons:

1. …

## Strategies/ approaches:

<e.g., extract method>

# **Name:** Long Method – Controller().display() [HIGH]

## Location:

Package: Interpreter

File: controller.py

Class: Controller

Method: display()

Lines: 11 – 45 (34)

## Reasons:

1. …

## Strategies/ approaches:

<e.g., extract method>

# **Name:** Long Method – Controller().serialize() [HIGH]

## Location:

Package: Interpreter

File: controller.py

Class: Controller

Method: serialize()

Lines: 83 – 109 (26)

## Reasons:

1. …

## Strategies/ approaches:

<e.g., extract method>

# **Name:** Long Method – Validator().validate() [MEDIUM]

## Location:

Package: Interpreter

File: validator.py

Class: Validator

Method: validate()

Lines: 51 – 67 (16)

## Reasons:

1. …

## Strategies/ approaches:

<e.g., extract method>

# **Name:** Switch Statement – CmdView().initialize() [LOW]

## Location:

Package: Interpreter

File: cmdview.py

Class: CmdView

Method: initialize()

Lines: 24 - 38

## Reasons:

1. The initialize() method is used to parse the sys.argv list. It loops through the list and tests every argument against a series of conditional statements.

## Strategies/ approaches:

<e.g., extract method>

# **Name:** Switch Statement – Controller().display()

This was the worst smell to me because…

## Location:

Package: Interpreter

File: controller.py

Class: Controller

Method: display()

Lines: 26 - 33

## Reasons:

1. The code block in the Controller.display() method that is a **switch statement** that compares an input passed into the method and calls a different display\_chart() method of the Visualizer class depending on what conditional branch was executed.
2. This area of the program breaks the Open/Close principle, meaning that it is not easily extensible. To extend the functionality to display more chart types I would have to add another branch to the if/else in Controller.display(), and add a whole new method in Visualizer for the new chart type, but it would contain more of the same duplicate code.

## Strategies/ approaches:

1. **Extract Method:** Firstly, I will extract the switch code block into a private method inside Controller.
2. **Move method:** Next, I will move the method into the Visualizer class.
3. **Replace Conditional with Polymorphism:** Lastly, I will replace the switch statement with an inheritance hierarchy. This will reduce the duplicate code, as well as following the Open/Close principle, making it easy to extend in the future.

# **Name:** Duplicate Code – Visualiser().display\_\*() [DONE]

## Location:

Package: Interpreter

File: visualiser.py

Class: Visualiser

Method: display\_bar() , display\_line(), display\_pie(), display\_radar()

Lines: 6 - 48

## Reasons:

1. The code in these four methods are nearly identical, except a different pygal object is called to display a different chart type for each method.

## Strategies/ approaches:

<e.g., extract method>

# **Name:** Speculative Generality – View() [DONE]

## Location:

Package: Interpreter

File: fileview.py

cmdview.py

Class: FileView

CmdView

Method: set()

Lines: fileview.py : line 44

cmdview.py : line 92

## Reasons:

* I attempted to create an interface called View in order to generalize the reading and writing of data to and from different domains with get() and set() methods.. I then implemented this interface with CmdView, FileView, and DatabaseView.
* The problem is that I never ended up implementing all of the functionality. DatabaseView implemented both get() and set(), but FileView and CmdView only implemented get(), while set() remained un-implemented.
* The empty set() methods are cluttering up the respective classes.
* Using such basic names as get and set makes the functionality of the modules vague. Removing the interface would mean these methods could be renamed to better describe their behavior, making the code easier to understand in these three modules.

## Strategies/ approaches:

1. **Collapse Hierarchy:** I will first remove the implementing classes dependency on the interface by removing the empty set() methods, and making them no longer inherit from View. Once this is done I can safely delete the View.py ABC module.
2. **Rename Method:** Next, I will rename the methods that were used by the interface, and name them to more clearly describe their behavior.

Sub-Steps: *[old-name] > [new-name]*

1. FileView.get() > FileView.read\_file()
2. DatabaseView.set() > DatabaseView.insert()
3. DatabaseView.get() > DatabaseView.retrieve()
4. CmdView.do\_get() > CmdView.do\_read()
5. **Rename Class:** I will also rename the classes because they no longer need to indicate an interface implementation. This will hopefully make their purpose easier to understand.

Sub-Steps: *[old-name] > [new-name]*

1. FileView > FileReader
2. DatabaseView > Database

I won’t rename CmdView because I feel the name accurately reflects the module’s purpose as a cmd-line interface (View in MVC). However, renaming FileView and DatabaseView makes sense considering they are no longer implementing the View interface.

# **Name:** Large Class – Controller() [DONE]

## Location:

Package: Interpreter

File: controller.py

Class: Controller

Method: serialize ()

Lines: 76

## Reasons:

* The controller class is used to manage the flow of data between the view (CmdView) and the various models (Validator, DataParser, Visualiser, DatabaseView), but near the end of the project it started to contain functionality that would be better suited in it’s own class, for example, the serialization functionality.
* I chose this as the next important bad smell to fix because it was cluttering up the controller and making it confusing to understand what it should be doing. It will also make it easier to clean up the serialize functionality when it is in its own class.

## Strategies/ approaches:

1. **Extract Class:** Mainly, I will move the serialize method into its own class. This should be fairly simple to do since it is only one method.

View ‘interface’