BCPR301 – Advanced Programming

Assessment: 3

Contents

[Design Pattern Identify: 3](#_Toc516695920)

[ Abstract Factory 3](#_Toc516695921)

[Location: 3](#_Toc516695922)

[Class Diagram: Assignment2: 3](#_Toc516695923)

[Location in Assignment2: 3](#_Toc516695924)

[Class Diagram: Assignment3 4](#_Toc516695925)

[Location in Assignment3: 4](#_Toc516695926)

[Reason Why Applied: 5](#_Toc516695927)

[Design Pattern Identify: 5](#_Toc516695928)

[ Builder 5](#_Toc516695929)

[Location: 5](#_Toc516695930)

[Class Diagram: Assignment2: 5](#_Toc516695931)

[Location in Assignment2: 6](#_Toc516695932)

[Class Diagram: Assignment3 6](#_Toc516695933)

[Location in Assignment3: 7](#_Toc516695934)

[Reason why Applied: 8](#_Toc516695935)

# Design Pattern Identify:

## **Abstract Factory Design Pattern**

# Location:

# Class Diagram: Assignment2:

Class Diagram before applying “Abstract Factory” design pattern

A screenshot of a social media post

Description generated with very high confidence

# Location in Assignment2:

|  |  |
| --- | --- |
| File\_handler.py | Is deal with csv file operation |
| Pickle.py | Change file name and functions from xmp.py to pickle.py  Line: 6 t0 20 |
| Database.py | Change file name to file\_database.py to keep in similarity of file name and easy to understand |
| Interpreter\_controller.py | Line: 12  16 to 18  238 to 275 |

# Class Diagram: Assignment3

Class Diagram after applying “Abstract Factory” design pattern

A screenshot of a social media post

Description generated with very high confidence

**Applying Design Pattern:** Please see attached code at locations mentioned below.

# Location in Assignment3:

|  |  |
| --- | --- |
| File\_handler.py |  |
| File\_Pickle.py | Line: 6 t0 20 |
| File\_Database.py | Change file name to file\_database.py |
| Interpreter\_controller.py | Line: 12  16 to 18  238 to 275 |
| File\_types.py | New Class |
| File\_types\_abstract.py | New Class |

# Reason Why Applied:

Abstracting the process of object generation

Previously all possible concrete products (File objects) were generated in \_\_main\_\_. Now only the concrete product(s) required are instantiated and that is done at run-time.

Encapsulation of all properties of a file object

Previously the file extension type, a list in inpterpreter\_controller.py, was separate from the file object. Thus, can cause future possible problems. Now there is a single place, the concrete factory (FileTypes.py) where both the file extension and file types and concrete products created are stored. They can only be accessed by calls to concrete factory get methods.

# Design Pattern Identify:

## Builder

# Location:

# Class Diagram: Assignment2:

Class Diagram before applying “Builder” design pattern

A screenshot of a social media post

Description generated with very high confidence

# Location in Assignment2:

|  |  |
| --- | --- |
| Interpreter\_controller.py | Line: 5 to 8  171 and 172  177 to 194  192 to 221 |
| View\_console.py | Line: 51 to 163 |
| View.py | Line: 53 to 100 |

# Class Diagram: Assignment3

Class Diagram after applying “Builder” design pattern

A screenshot of a social media post

Description generated with very high confidence

Applying Design Pattern:Please see attached code at locations mentioned below.

# Location in Assignment3:

|  |  |
| --- | --- |
| Interpreter\_controller.py | Line: 5 to 8  171 and 172  177 to 194  192 to 221 |
| Console\_View.py | Line: 51 to 163 |
| View.py | Change to Grapg\_builder.py  Line: 53 to 100 |
| Graph\_director.py | Director |
| Graph.py | ConcreteBuilder1 |
| Show\_graph.py | ConcreteBuilder2 |

# Reason why Applied:

Separate graph displaying from graph building

Previously the responsibility of both building the graph and displaying the graph was handed directly by the interpreter controller class. Which meant MVC design architecture model was not actually functioning.

Builder Pattern gives back a plt graph object

After implementation of the Builder Pattern the interpreter controller unable to use a director to build the graph and then take the returned graph and send it back to the view which takes the built object and shows the graph. Thus, giving the actual View job to the View part of the MVC.

Controller does not need to know steps to build a concrete object

Implementing the Builder Pattern also means that the controller no longer needs to know the steps required to build a graph. The actual concrete builders differ in their individual construction steps but they all adhere to the Graph Interface so the director enable call the methods in the interface and thus create and return a graph object.