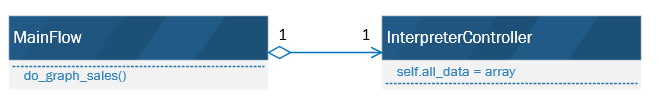
BCPR301 - Assignment 3 Documentation

# Identify Design Patterns in Frameworks

# Implement Design Patterns in Assignment 2

## Class Diagram Before Implementation



## Location

* + 1. Before Code (Assignment 2)
       1. ~/main.py \\ MainFlow.do\_graph\_sales \\ 172 - 191
    2. After Code (Assignment 3)
       1. ~/main.py \\ MainFlow.do\_graph \\ 173 - 201
       2. ~/builder/graph\_builder.py \\ All classes \\ Whole file

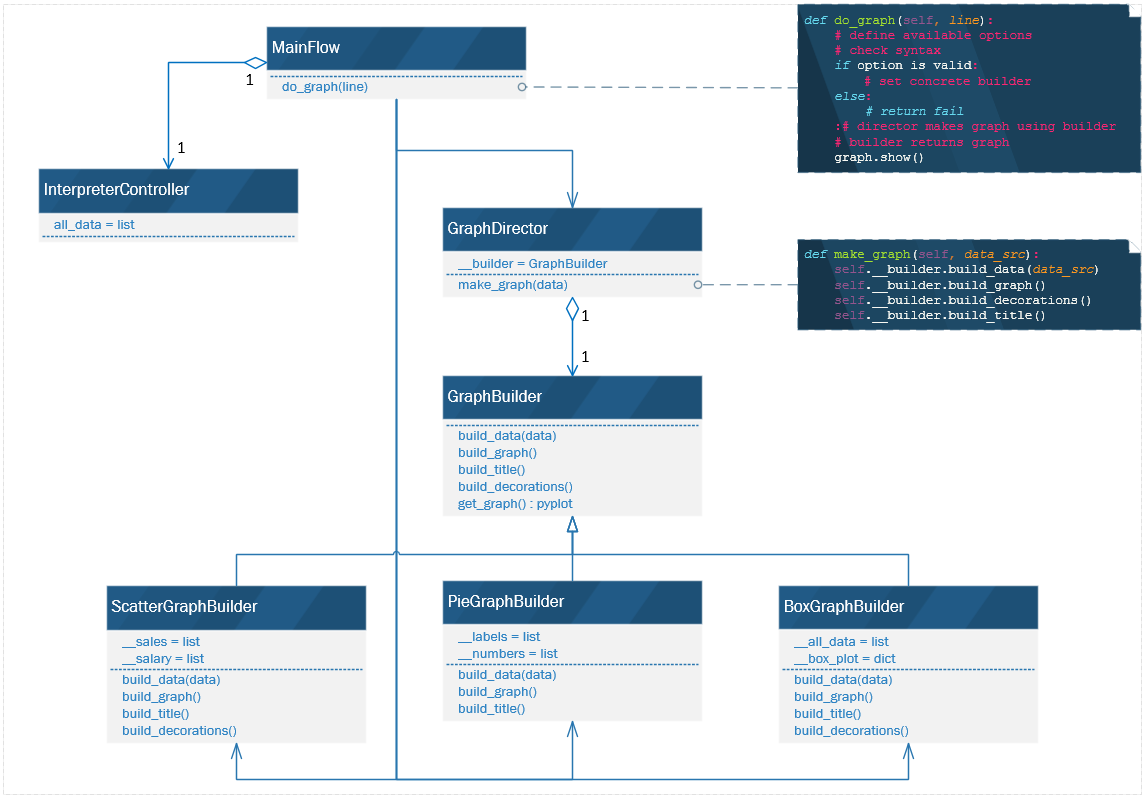
## Pattern

* + 1. Builder

## Reasoning for Implementation

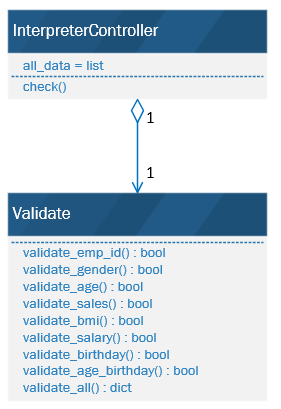
* + 1. Based on the assumption that user requirements would require further ways of displaying graphs, a builder design pattern allows increased functionality for creating different charts using matplotlib, having a single concrete builder for each chart to be displayed.

## Class Diagram After Implementation



# Implement Design Patterns in Assignment 2

## Class Diagram Before Implementation



## Location

* + 1. Before Code (Assignment 2)
       1. ~/controllers/main\_controller.py \\ InterpreterController.check \\ 62
       2. ~/models/model\_validator.py \\ Validate \\ 11 - 173

* + 1. After Code (Assignment 3)
       1. ~/controllers/main\_controller.py \\ InterpreterController.check \\ 56 & 64
       2. ~/models/model\_validator.py \\ Validate \\ 6 - 34
       3. ~/strategy/validate\_strategy.py \\ All classes \\ Whole file

## Pattern

* + 1. Strategy

## Reasoning for Implementation

* + 1. This is a good design pattern for validation because each validation method uses a different variation of a similar algorithm (regular expression). Strategy also allows future proofing if the client were to use different data requiring alternative validation to what currently exists.

## Class Diagram After Implementation

