FIT3077 Assignment 2 – Phase 2

Matt Gueit and Jane Tran (Team Name: Hanzomain)
May, 2017

1.0 Project Implementation and Design Principles

The Weather Monitor application implemented in this assignment makes use of the Model-View-Controller software architectural pattern. The three main components include; the model and data classes dependent of the user interface; the view of data generating the output including the graphs for Rainfall and Temperature; and the controller which manages the data flow of the user input in order to establish the monitors. This architecture allows for parallel development of the front and back end of the software as well as improve code reusability. Further, this enables high cohesion as demonstrated by classes within the program initialising instances of other classes such as Location having separate Rainfall and Temperature classes as opposed to rainfall and temperature being attributes, and minimises coupling which has been useful in having to not only maintain the program, but also modify it so as to extend the program's functionality allowing for multiple views of the model.

One of the most apparent software principles applied in the Weather Monitor application was the Common Closure principle. The classes within the application that are linked have been grouped together for ease of modification and testing. This is also minimises unnecessary dependencies. An example of this is apparent in the Locations class where a type of Web Client Class is instantiated for every location. This is an abstract class, which Web Client Melbourne and Web Client Time Lapse inherit from. Its purpose is to manage calls of a specific location to the web client and feed this information back to the Location so that the data can be shown on the Monitor. In addition to this, an update timer instance is created for each location to update the weather information at the right intervals. Originally for Stage 1, a 'central' Timer was generated within the controller and this would notify the program to update the data of every active location at the one time. Although this was very effective if the main goal was to design the Weather Monitor application with the most minimal network traffic, it is less effective when a new web client is used i.e. the time-lapse monitor refreshes faster than the original. It is no longer the controller's responsibility to keep track of time and removes the association between the controller and the web client. This segregates the dependencies between classes that do not have to be associated with each other for ease of reusability, underlining the Common Closure principle.

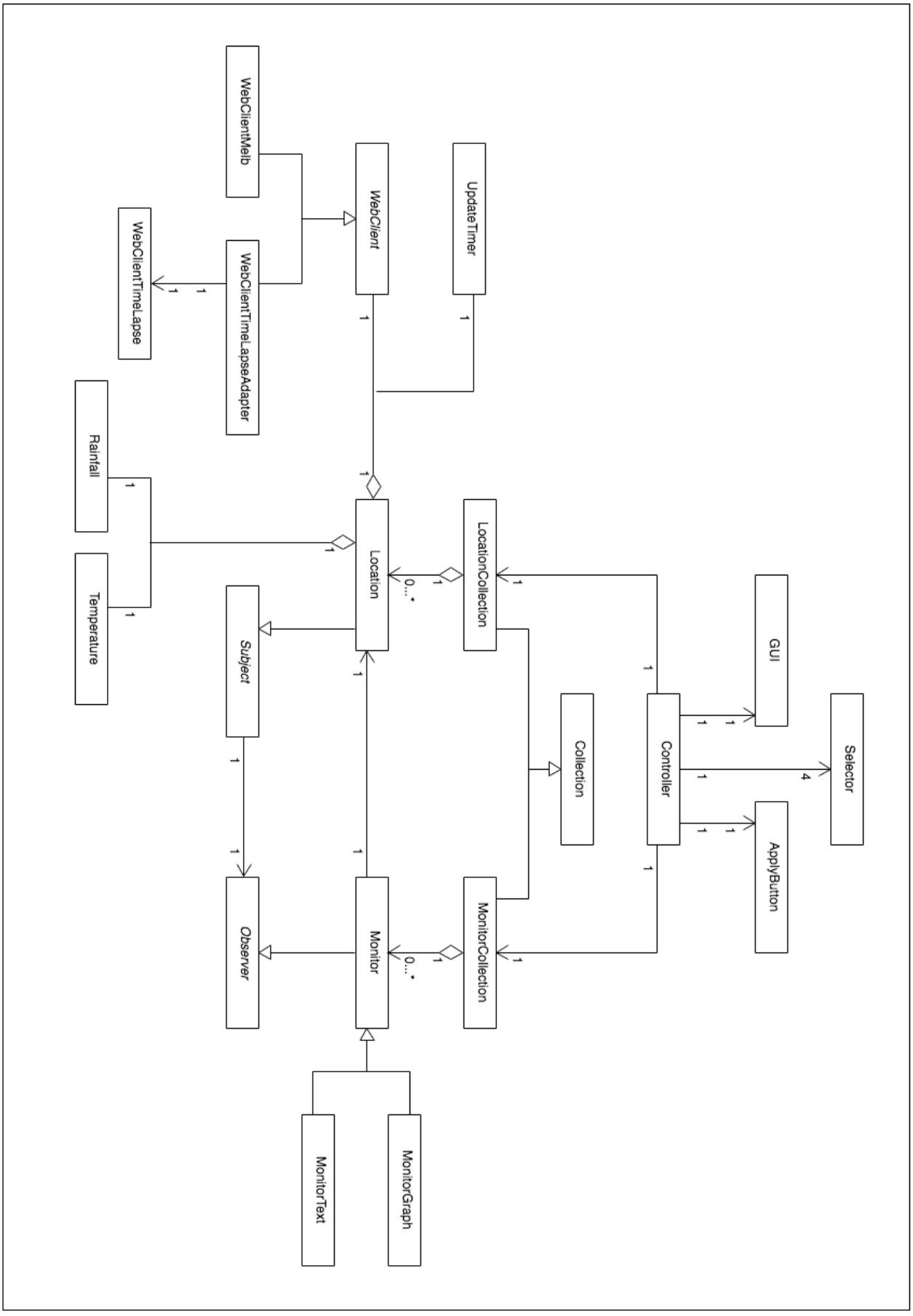
A main aspect of the Model-View-Controller methodology relies on the Interface Segregation principle whereby the main controller facilitates various services. Each of which are grouped through abstract classes. An example of this is the Monitor class. This class contains base attributes such as location, timestamp, rainfall and temperature and functions such as update and remove. It is then inherited by Monitor Graph and Monitor Text. This eliminates implicit coupling and dependencies between the clients as they share the Monitor observer interface. If the Monitor observer class did not exist, any changes to its main functions would have to be changed separately in the Monitor Graph and Monitor Text classes. With the implementation of the Monitor class, the overhead is minimised and these functions are able to be changed in one

place. Through this, the Monitor class is more stable as there are other classes dependant on itself as opposed to it being dependent on other classes hence exhibiting the Stable Dependency principle.

One of the main benefits of the Weather Monitor application is that there are no acyclic dependencies within the design of the program. This means that changes affecting one class will not have any adverse effects on other related classes and modules can more easily be reused. The key design patterns implemented in this program are the observer and adapter patterns. The observer pattern as mentioned above provides a way for the Monitor (the observer) to be updated when the Location (the subject) has new information. By implementing the Observer abstract class, different kinds of observers can be easily added. By using a Subject abstract class, Location is no longer dependent on a concrete class; this is following the Open-Closed Principle and the Liskov Substitution Principle.

The adapter class in this program is demonstrated through the Web Client Time Lapse Adapter. The web client that was originally being used was the MelbourneWeather2, which already had its values in Celsius and millimetres. However, the MelbourneWeatherTimelapse client has different function calls and represented the rainfall data centimetres, and the temperature data in Kelvin. For those reasons, the new web client was not compatible with Web Client abstract class. So in order for the new web client to inherit from the Web Client abstract class, the adapter was implemented to convert the data. This enables the reusability of the Web Client abstract class.

2.0 UML Class Diagram



3.0 Use Case Scenarios

3.1 View Text Monitor for Location

Use Case Name	View Text Monitor for Location	
Description	User selects a location to view the weather data for that location.	
Requirements	Location List has all the available locations for the user to select. User has already selected a viewing mode of either Temperature, Rainfall or Both.	
End Condition	Weather monitor appears with the location's data, with option to view the graph data.	
Actors	User, Location	
Main Flow	**Step**	**Action**
	1.	User selects a location
	2.	System checks if location is already active
	3.	Monitor and Location objects are created for the location
	4.	Web Client associated with the Location retrieves data specific to the location
	5.	Monitor is updated with this data and displays to the GUI
	6.	User can now see the location data.
Alternate Courses	2a.	Location is already active.
	2b.	Notify user that the location is already active.

3.2 View Graph Monitor for Location

Use Case Name	View Graph Monitor for Location	
Description	User selects to view the graph data in graph form for a location.	
Requirements	User has already selected the viewing mode and location. The GUI is currently displaying the location data.	
End Condition	A graph appears with real time data for the location.	
Actors	User, Location	
Main Flow	**Step**	**Action**
	1.	User selects a to view graph data for the location
	2.	System checks if graph for location is already active
	3.	A graph object is created for the specific location
	4.	Web Client retrieves data to Location
	5.	Graph monitor is updated with location information
	6.	User can now see the graph data for the location.
Alternate Courses	2a.	Graph for location is already active
	2b.	Notify user that the graph is already active.

3.3 Close Graph Monitor for Location

Use Case Name	Close Graph Monitor for Location
Description	User selects to close the graph monitor for a specific location.
Requirements	User has already selected the viewing mode and location. The GUI is currently displaying the graph monitor.
End Condition	The graph containing the location data is closed.

Actors	User, Location, Graph Monitor	
Main Flow	**Step**	**Action**
	1.	User selects a to close the graph data for the location
	2.	Update function is terminated.
	3.	The graph window is terminated.
	4.	Location is removed from the location collection.
	5.	Monitor is removed from the monitor collection.
	6.	The window containing the graph data is destroyed.

3.4 Close Text Monitor for Location

Use Case Name	Close Text Monitor for Location	
Description	User selects to close the text monitor for a specific location.	
Requirements	User has already selected the viewing mode and location. The GUI is currently displaying the text monitor for the selected location.	
End Condition	The graph containing the location data is closed.	
Actors	User, Location, Graph Monitor	
Main Flow	**Step**	**Action**
	1.	User selects a to close the text data for the location
	2.	Update function is terminated.
	3.	The text frame is terminated.
	4.	Location is removed from the location collection.
	5.	The frame containing the data is destroyed.