FIT3077 Assignment 2

Monitor Application - Design Report

By Douglas Rintoul and Thomas Krasicki

We have designed and created a Weather Monitor application that allows a user to view weather data from various SOAP based web services. We have attempted to design our software with adherence to object orientated principles, and have focused on creating an extensible and adaptable piece of software to allow for any change in requirements.

The most important relationship in this design is the between the Location and WeatherMonitor classes. This relationship utilises the observer design pattern - a Location is a subject, and a WeatherMonitor is an observer. The main controller will schedule the Location objects to update it's data, and the Location will be responsibile for updating all of its listeners. The monitors are then responsible for handling data, converting them to the correct format, and displaying the elements for the GUI frame that is created in the controller. We have decided to use this pattern for several reasons:

- No modification is needed to the subject to allow for more observers to be implemented, this gives us the ability to add and remove observers easily with varying types of data to display to the user, promoting extensibility and providing loose coupling between the two objects.

- Data is efficiently sent through to many different objects at one time and by the nature of the update call only objects that require updating are given the call, thus reducing any unnecessary calls to the web service.

Following the Dependency Inversion Principle, we have implemented an abstract WeatherWebService class which allows our application to use varying web services regardless of their implementation. Concrete classes are derived from the abstract class for each web services that needs to be implemented, and this design allows us to easily integrate a web service's data for our system to use.

Locations for each web service are stored and referenced in a LocationList object that is in charge of updating Location objects, and also allows us to generate a list of locations for the user to select. Unfortunately these two objects form a cyclic dependency in our design. We have decided keep this tight coupling, as the trade off allows us to deference and destroy Location objects when they are no longer needed, which reduces the number of calls to the web services.

Locations store WeatherData objects. The WeatherData objects contain all related data information relating to a single type of WeatherData, such as temperature. These objects allow Location objects to store any varying amounts of data types, each with it's own implementation and typing. Doing so allows us to have Locations store any combination of weather data, and allows extensibility in storing unforeseen types in the future.