FIT3077 - Assignment 2
Design Document

By Douglas Rintoul and Thomas Krasicki

Overview

For this assignment, we have developed and implemented a design which we believe is extendable and adaptable for the second stage of the Assignment 2. We attempted to design our software in such a manner that should the requirements for the project change, there would be minimal changes needed to adapt the program.

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. Using this pattern was important for ensuring that any number of monitors could be listening to a Location, and that unnecessary calls to the web service would be avoided.

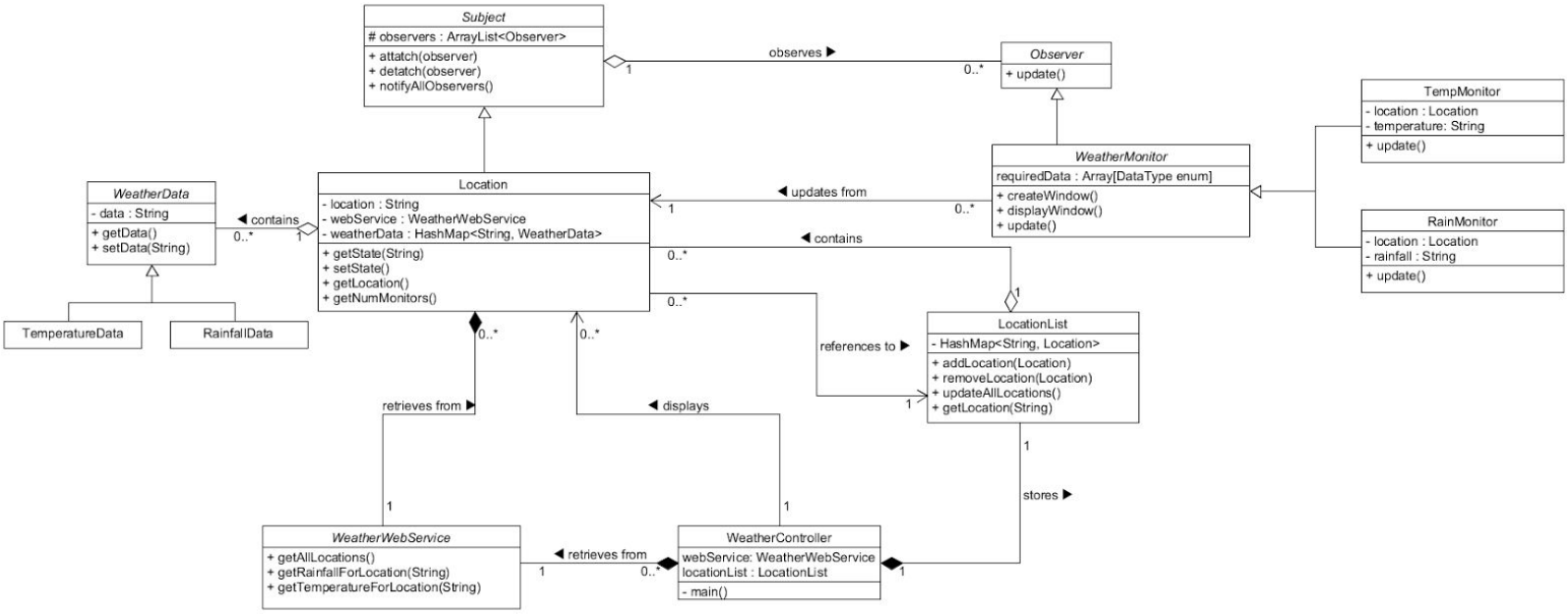
Using this pattern means that when the main controller tells each Location to update its data, the Location then tells all of its monitors to update themselves. These monitors then take the data they need from the Location and display it in the way they need to. This model allows for more monitors to easily be added, and for monitors to do whatever they want to do with the data they retrieve - manipulating and displaying it as they need.

Locations track what data they need to be storing based on what monitors are attached to them - this creates coupling between the Location and WeatherMonitor classes, however it allows us to more efficiently manage the calls to the web service - there is no point in a Location getting rainfall data from the web service if there are no monitors that need it.

This system requires the use of a web service to get a list of locations, and rainfall and temperature data for each of these locations. We created an abstract class WeatherWebService which allows the system to use such a service regardless of how it is implemented.

We chose to only implement 2 concrete WeatherMonitor classes - RainfallMonitor and TemperatureMonitor. We believe that since one of each of these can be open at the same time, it fulfils the requirement of displaying “rainfall and/or location” from the system requirements. Adding such a class in would be easy - it would only require creating the class, modifying the monitor creation code, and then creating a new button on the GUI for it.

Overall Class Diagram



Extended Use Case - Viewing the Temperature of a Location

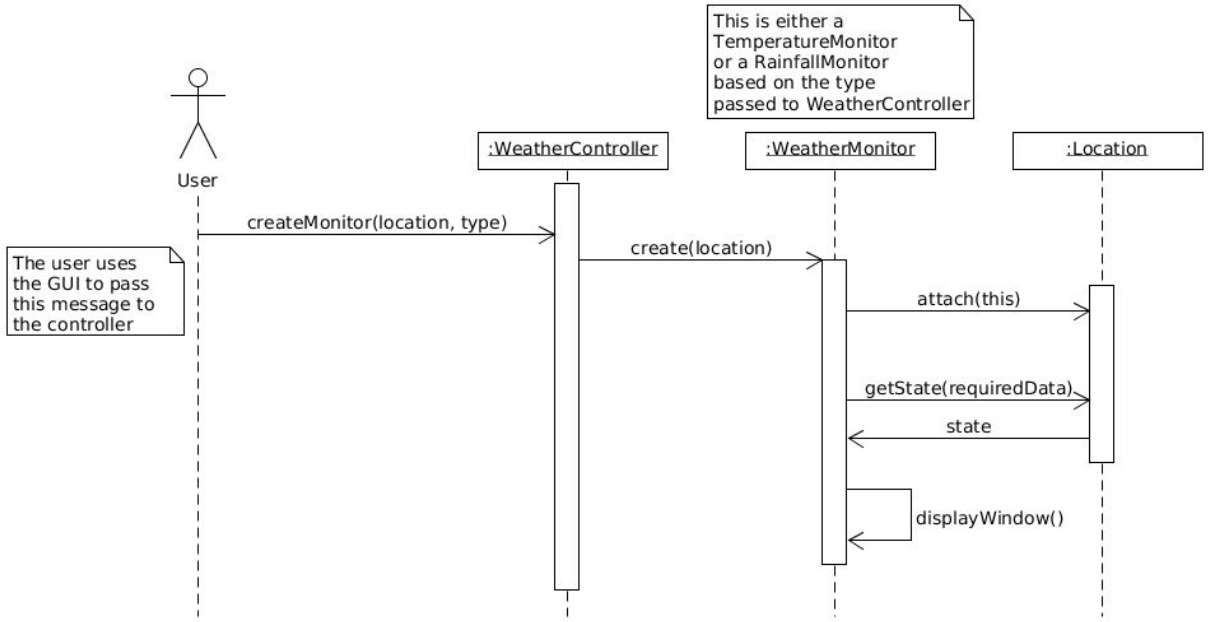
Use Case: Viewing the temperature of a location
Actors: User
Overview: A user requests to see the locations available to view. The user selects which location they wish to view information of, and then selects to view the temperature. The system displays the selected location and temperature to the user.
Type: Primary
Cross-References: None

Main Flow of Events:

Actor Action	System Action
1. The user requests to see the locations that are available.	2. The system displays the locations to the user.
3. The user selects which location they wish to view information of.	
4. The user selects to view the temperature of their selected location.	5. The system displays the name of the location and the temperature to the user.

Alternate Flows:
2a. The system cannot find the locations from the web service, and displays a message informing the user that the data could not be retrieved
5a. The system cannot find the temperature of the requested location, and displays a message informing the user that the data could not be retrieved.

Sequence Diagram - Create Monitor



Sequence Diagram - Closing a Monitor