TABLE OF CONTENTS

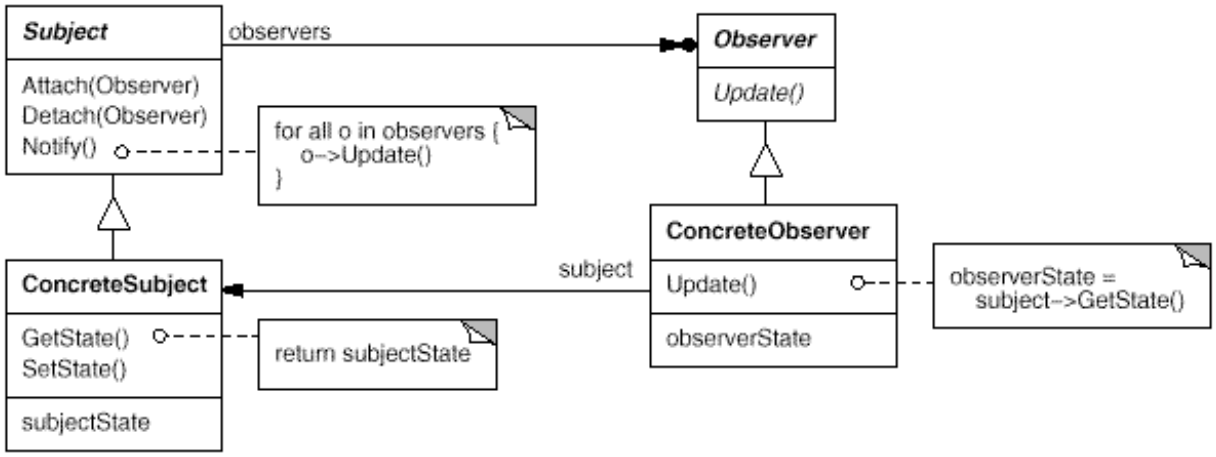
1 INTRODUCTION

This is an assignment to test the programming skills in Java and the knowledge on the Observer pattern and the Model View Controller (MVC) Framework.

According to *pcmag.com,* **MVC** is An architecture for building applications that separate the data (model) from the user interface (view) and the processing (controller).

Observer Pattern on the other hand, is a design pattern implemented in Java that defines a one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically. It is also known as **Dependents** or **Publish-Subscribe.**

This is a class diagram to further explain the Observer pattern.



To implement the Observer pattern and the MVC framework in this assignment, the problem had to be separated into a Model, separate Views and a Controller. It was separated into the following Java classes;

MODEL

- ClockModel.java
- ClockUpdate.java

VIEWS

- ClockAnalogView.java
- ClockDigitalView.java
- ClockSliderView.java
- ClockButtons.java

CONTROLLER

ClockController.java

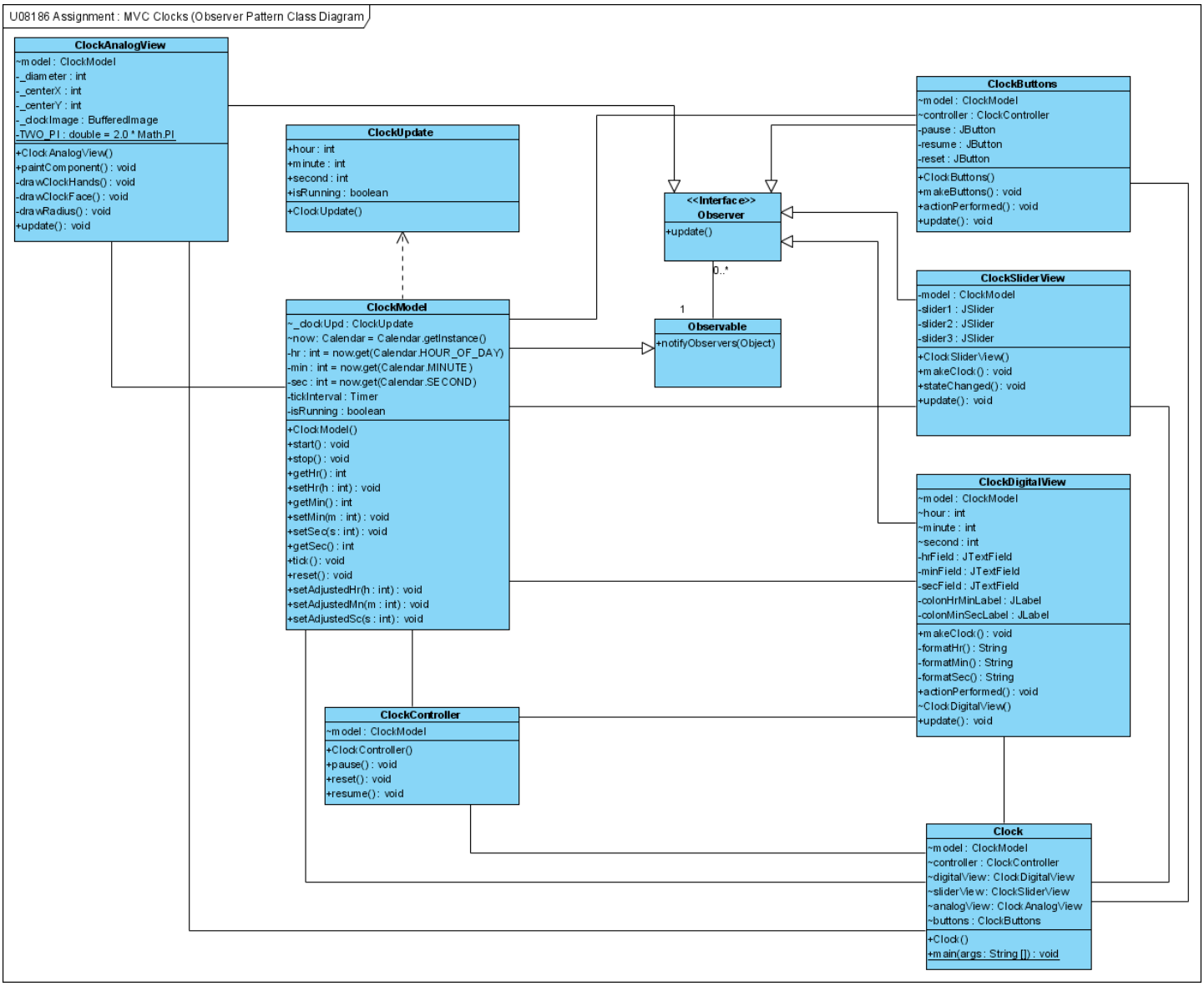
The Class Diagram in the following chapter explains how the classes relate to each other when implemented using the Observer Design Pattern and MVC Framework.

2 DESIGN

2.1 Class Diagram

In this class diagram, the model *ClockModel.java* notifies the *Observable.* The Views that are listening to *Observer interface* class call the update function that updates their data to match that of the *ClockModel.*

ClockController.java gets user input from the Views and updates the Model accordingly, which then updates the other Views that are listening to the *Observer interface.*



2.2 Algorithm/Pseudocode

On program start, this is how the Pseudocode flows:

- READ System Time Values; Hour, Minute, Second.
- SET System Time Values as the starting values for all the clocks.
- SHOW the Time on the Views.
- REPEAT

    every after 1 second, INCREMENT the value of Seconds by 1

    IF seconds > 59 THEN

        INCREMENT the value of Minutes by 1

            IF minutes > 59 THEN

                INCREMENT the value of Hours by 1

    setChanged

    notifyObservers.

    ELSE

        seconds = seconds

  UNTIL

    pause_button is Clicked

- IF pause_button_Clicked THEN

    resume_button_Enabled,

    resetClocks_button_Enabled,

    stop all Clocks

   ELSE

    continue running the clocks.

- IF sliders are changed THEN

    update the model

    change all clock values to match the sliders

  ELSE

    continue running the clocks.

- IF textfields are changed THEN

    update the model

    change all clock values to match the textfields

    ELSE

    continue running the clocks.

- EXIT.

3 TEST PLAN

Test Cases/Data	Expected Result	Actual Result	Comments
“Clocks.jar” is started.	- All 3 clocks should display current System Time. - “Pause” button should be enabled. - “Resume” button should be disabled. - “Reset Clocks” button should be disabled.	- All 3 clocks display current System Time. - “Pause” button is enabled. - “Resume” button is disabled. - “Reset Clocks” button is disabled.	Result is as expected
“Pause” button is clicked.	- All clocks should stop running. - “Pause” button should be disabled. - “Resume” button should be enabled. - “Reset Clocks” button should be enabled.	- All clocks stop running. - “Pause” button is disabled. - “Resume” button is enabled. - “Reset Clocks” button is enabled.	Result is as expected
“Resume” button is clicked.	- All clocks should continue from where they stopped. - “Pause” button should be enabled. - “Resume” button should be disabled. - “Reset Clocks” button should be disabled.	- All clocks continue from where they stopped. - “Pause” button is enabled. - “Resume” button is disabled. - “Reset Clocks” button is disabled.	Result is as expected
“Reset Clocks” button is clicked.	- All clocks should be set to 00:00:00 time.	- All clocks are set to 00:00:00 time.	Result is as expected
“Hours Slider” is changed.	- All clocks should change to the slider’s allocated hour.	- All clocks change to the slider’s allocated hour.	Result is as expected
“Minutes Slider” is changed.	- All clocks should change to the slider’s allocated minutes.	All clocks change to the slider’s allocated minutes.	Result is as expected
“Seconds Slider” is changed.	- All clocks should change to the slider’s allocated seconds.	All clocks change to the slider’s allocated seconds.	Result is as expected
“10” is entered in the Hours TextField.	- All clocks should set their Hour value to “10”.	- All clocks set their Hour value to “10”.	Result is as expected
“29” is entered in the Hours TextField.	- All clocks should set their Hour value to “00”.	- All clocks set their Hour value to “00”.	Result is as expected
“aodud” is entered in the Hours TextField.	- All clocks should set their Hour value to “00”.	- All clocks set their Hour value to “10”.	Result is as expected

Test Cases/Data	Expected Result	Actual Result	Comments
“28” is entered in the Minutes TextField.	- All clocks should set their Minutes value to “28”.	- All clocks set their Minutes value to “28”.	Result is as expected
“72” is entered in the Minutes TextField.	- All clocks should set their Minutes value to “00”.	- All clocks set their Minutes value to “00”.	Result is as expected
“djls” is entered in the Minute TextField.	- All clocks should set their Minutes value to “00”.	- All clocks set their Minutes value to “00”.	Result is as expected
“46” is entered in the Seconds TextField.	- All clocks should set their Seconds value to “46”.	- All clocks set their Seconds value to “46”.	Result is as expected
“69” is entered in the Seconds TextField.	- All clocks should set their Seconds value to “00”.	- All clocks set their Seconds value to “00”.	Result is as expected
“pfijahp” is entered in the Seconds TextField.	- All clocks should set their Seconds value to “00”.	- All clocks set their Seconds value to “00”.	Result is as expected

4 PROGRAM LISTING

This is the actual program source code. The files are arranged in order as follows;

- *ClockUpdate.java*
- *ClockModel.java* //Model
- *ClockDigitalView.java* //View
- *ClockSliderView.java* //View
- *ClockButtons.java* //View
- *ClockAnalogView.java* //View
- *ClockController.java* //Controller
- *Clock.java* //Main Program

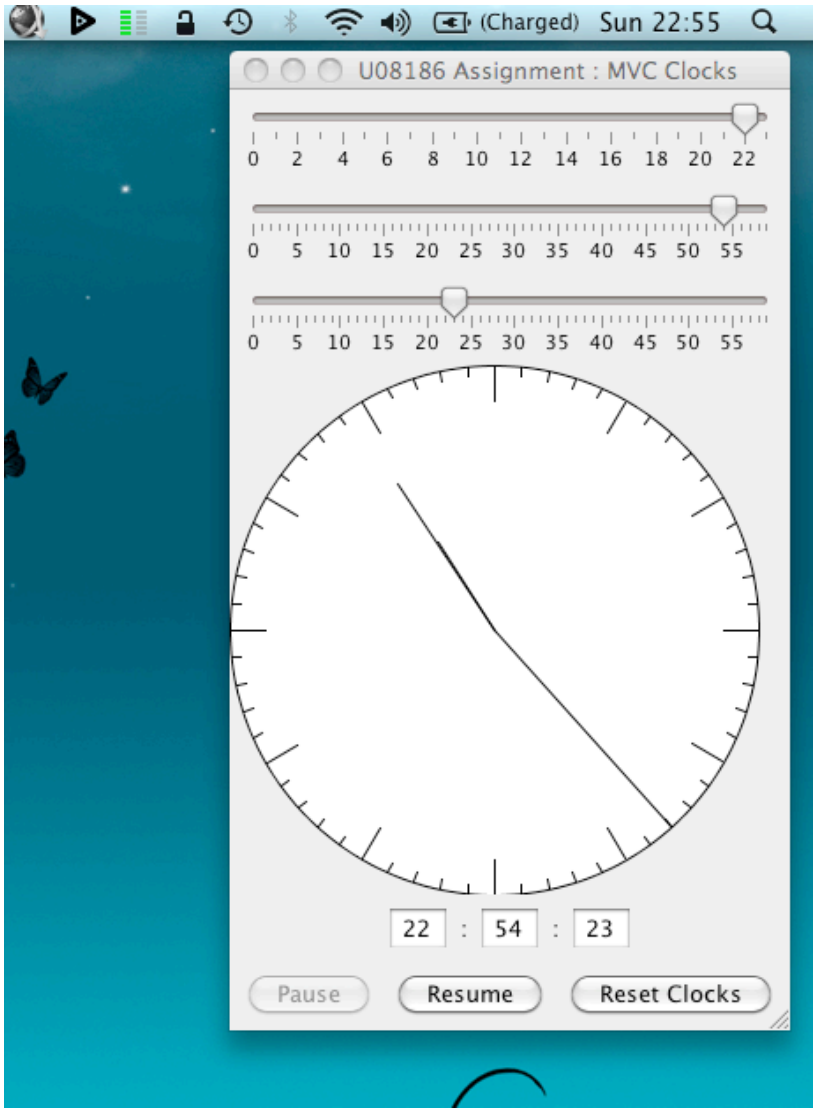
5 SCREEN SHOTS

5.1 Displaying System Time at First Run.

- All the Clocks display Current System Time when “Clocks.jar” is opened.
- “Pause” button is enabled.
- “Resume” button is disabled.
- “Reset Clocks” button is disabled.

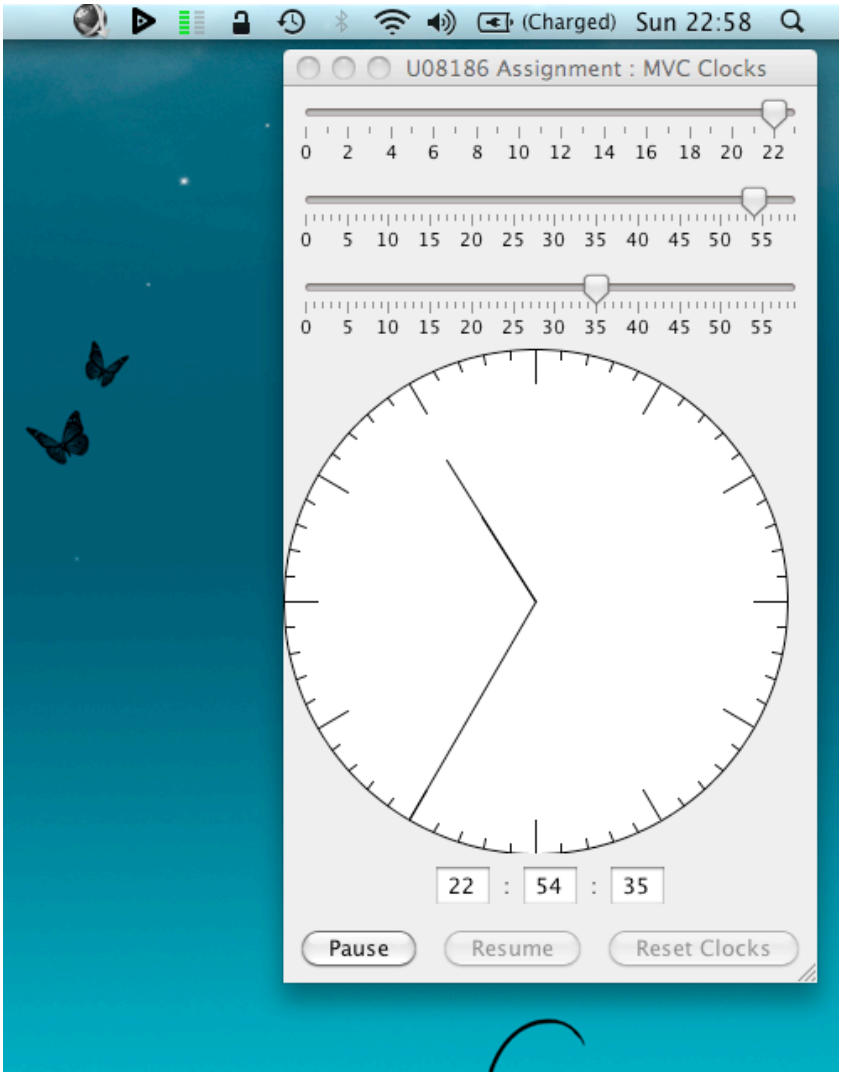
5.2 “Pause” button is Clicked.

- “Pause” button is disabled.
- “Resume” button is enabled.
- “Reset Clocks” button is enabled.



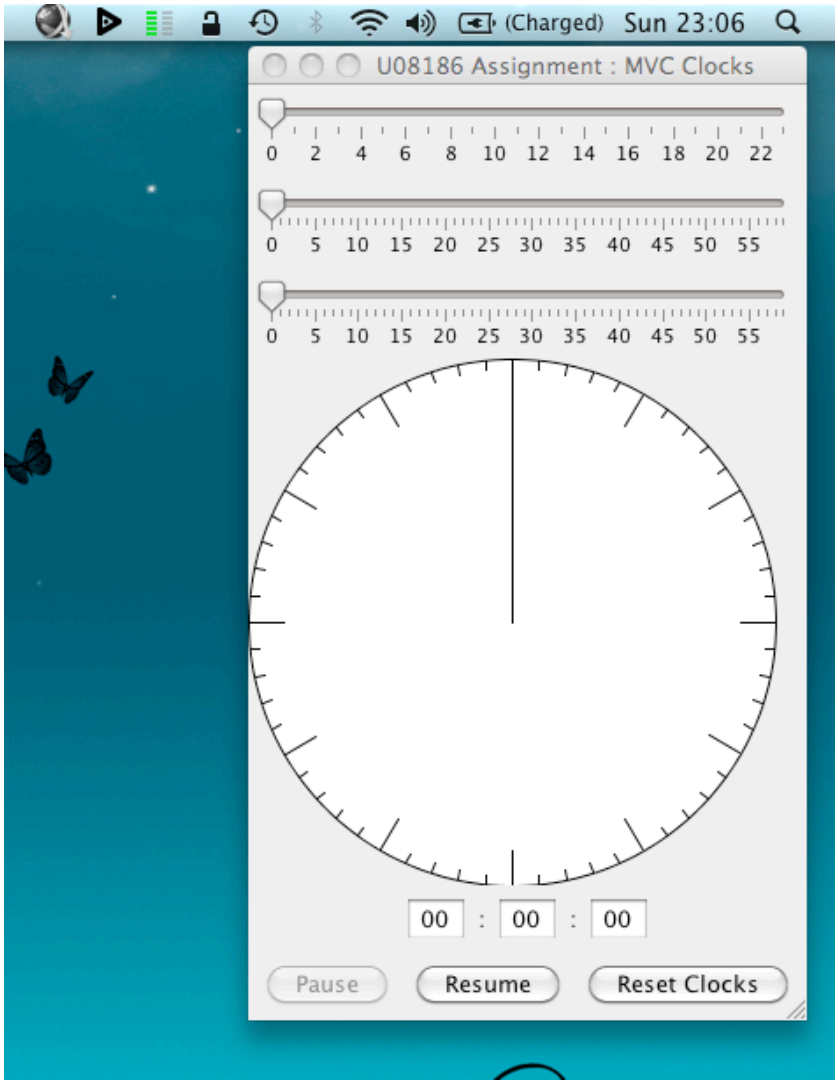
5.3 “Resume” button is clicked

- “Resume” button is disabled.
- “Reset Clocks” button is disabled.
- “Pause” button is enabled.
- All Clocks continue from where they stopped.



5.3 “Reset Clocks” button is Clicked.

- All Clocks set their time to 00:00:00.



6 CONCLUSION

This program has been written perfectly and it executes without any errors, either during compilation or execution.

This program has no system limitations and no program bugs were identified at the time of testing. Testing has been limited as to the Test Plan provided in the earlier chapters. If any program errors or bugs are to arise during the testing of this program, then I must declare that it is out of my abilities to solve it.

This is a simple program about different clocks but it has complex implementation of the Observer pattern and then MVC architecture.
Due to the research performed to be able to complete this assignment successfully on time, I have come to gain great knowledge in the Java programming language, Design Patterns and the MVC architecture.