## Purpose

The purpose of homework 2 was to create an application (Android or iOS) that allows a user to create both digital and analog clocks. The application shall be designed utilizing a Model View Controller (MVC) design. The MVC is intended to split up the functions of the program to allow them to be interchangeable and increase future usability. It also implements abstract classes in order to have a "cascading" effect on the children functions, they are all connected via the "Controller". In addition to creating Analog and Digital clocks, the user shall be able to change the attributes of the clocks (Second, Minute, and Hour). These effects should replicate throughout all clocks, setting all the clocks to the new updated times. The user shall also be able to alter the Year, Month, and Day of the program as well. Implementation of altering clock/date information will be done by utilizing incrementation and decrementation of the date information by use of + and - buttons.
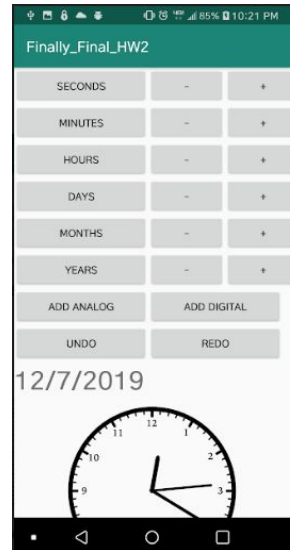
Model View Controller

The application was created with the Model View Controller design in mind, breaking up the program into module sections. The clock information (Time/Date) are controlled by a single class called ClockModel. Once the application is initiated, the ClockModel is called, this creates a Calendar object from which all time and date for the clocks are referenced from. All clocks are controlled from the ClockController function, this function is responsible for communicating with the ClockModel. ClockController will communicate with the ClockModel and pass information as to what the new Calendar object will look like after the incrementing or decrementing has taken place. Finally, the UndoRedo class is where all changes to the Calendar object are copied and put into an ArrayList<Calendar>. If a change to the date or time takes place, the UndoRedo class copies that Calendar object and puts it into the ArrayList. This is done for use when the undo function is called, which pops off the most recent Calendar object and sets the clocks to that Calendar information.
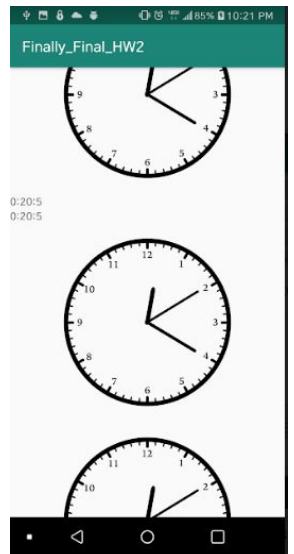
Model View Controller Experience

This was my first experience utilizing the MVC and to be quite honest I was a little lost at first. But after slowing down and understanding what each part of the design was supposed to it became much clearer about how I should build this project design. The hardest part of this application was figuring out the breakup of the MVC and how all the sections interact with each other. That took me by surprise, but after looking at the code for a good amount of time (20 hours) I was able to understand the interaction between the different parts of the design. This style of programming is absolutely different than anything that I have experienced at WSU, but it absolutely was worth understanding for future use in projects.

UML Diagram

MainActivity
- ArrayList<CV> Name
- ClockLayout Handle
- LinearLayout myLayout
- TextView theDate

- void onCreate()
+ void setCal()
+ void newAnalog()
+ void minusSec()
+ void addSec()
+ void minusMin()
+ void addMin()
+ void minusHour()
+ void addHour()
+ void minusDay()
+ void addDay()
+void minusMonth()
+ void addMonth()
+ void minusYear()
+ void addYear()
+ void updateClocks()
+ void undoer()
+ void redoer()

User

ClockControllers
- ClockModel CM
- Context context
- UndoRedo undoredo

+ ClockViewer gimmeAClock()
+ Calendar returnCal()
+ void undo()
+ void redo()
+ void addSecond()
+ void subSecond()
+ void addMinute()
+ void subMinute()
+ void addHour()
+ void subHour()
+ void addDay()
+ void subDay()
+ void addMonth()
+ void subMonth()
+ void addYear()
+ void subYear()

Java.util.Calendar

<<Abstract>>
ClockViewer
- Context context

+ ClockViewer()
+ View getClockView()
+ void refreshTime()

DigitalClock
- TextView tv

+ DigitalClock()
+ View getClockView()
+ Void refreshTime()

Extends

VectorAnalogClock

AnalogHelper
- AnalogClock analclock

+ View getClockView()
+ void setTimes()
+ void refreshTime()

Extends

AnalogClock

+ AnalogClock()
+ void init()

Extends

Java.util.Calendar

ClockModel
- Calendar cal

+ ClockModel()
+ Calendar getCal()
+ void setCal()

UndoRedo
- Stack<Calendar> calStack
- Stack<Calendar> redoStack

+ UndoRedo()
+ void pushStack()
+ void popStack()
+ void redoMe()