Part 2

1. In a standard Model-View-ViewModel architecture, what role does each part play? What is the Model? What is the View? What is the ViewModel?
   1. Model:
      1. Represents application’s data and business logic
      2. Responsible for managing and manipulating the data, performing calculations and enforcing business rules.
   2. View:
      1. Represents the user interface (UI) or the presentation layer of the application.
      2. Responsible for displaying the data from the model to the user and capturing user input.
      3. Do not contain much business logic..
      4. Observes the ViewModel to stay synchronized with data.
   3. ViewModel
      1. Intermediary between the Model and the View.
      2. Serves as a bridge that connects the data and business logic of the Model with the View
      3. Contains presentation logic that formats and prepares data from the Model for display in the View.
      4. Makes data and commands available for the View to use
      5. Handles user input and translates it into actions that affect application components.
2. The [documentation for Android's ViewModel](https://developer.android.com/topic/libraries/architecture/viewmodel) says, "The ViewModel class is designed to store and manage UI-related data in a lifecycle conscious way." What is meant by the term "lifecycle conscious way?" What lifecycle is the ViewModel aware or conscious of?
   1. The ViewModel is conscious of the dynamic environment that is held by the device the application is held in, and will handle data that gets destroyed by the main activity when, for example, the user rotates the phone. During this, the application destroys and recreates the ‘onCreate()’ method, the data that was in use before it was destroyed is held by the ViewModel and fills the View with the necessary data upon recreation.
3. [MutableLiveData](https://developer.android.com/reference/androidx/lifecycle/MutableLiveData" \t "_blank)provides two methods to update the value, setValue() and postValue(). When should one be used rather than the other?
   1. The choice between these two methods depends on the context in which the data is being set. setValue() should be used when being called from the main-Activity or the UI componenet, and postValue() should be used when setting the value from a non-UI element. setValue() runs whenever the main method runs, and postValue() runs the next time the main method runs.
4. in the [MainActivity code for Part 1](https://gist.github.com/tgibbons-css/9d6492bbc6e56a039d5a2b2279ce5aab" \t "_blank), an observer is set on the ViewModel's order status. Describe what this code does, including what method is called when the observer notices a change in the order status.
   1. The code observes the orderStatus MutableLiveData String variable of the ViewModel class. The onChanged() method is called when data inside that variable is changed and the method takes what was changed and adds it to the existing text that is in the textOrder field separated by a newline character.