Lab: Stopwatch

Lab Objectives:

- Create a stopwatch ViewModel object
- Create a stopwatch layout
- Use a Handler to periodically update the stopwatch
- Use a LiveData object to update the view

What to Turn in:

- Demonstrate your stopwatch app to a TA.

Stopwatch Behavior

A stopwatch is a tool that keeps track of elapsed time. A stopwatch has a **display** that shows the current elapsed time. A stopwatch can be **started**, **stopped**, and **reset**. When a stopwatch is started, the elapsed time begins increasing. When a stopwatch is stopped, the elapsed time stops increasing. When a stopwatch is reset, the elapsed time is set to zero. For our purposes, resetting the stopwatch does not stop it if it is running. Note that starting the stopwatch does not reset the stopwatch (i.e. stopping and starting without resetting simply "pauses" the stopwatch).

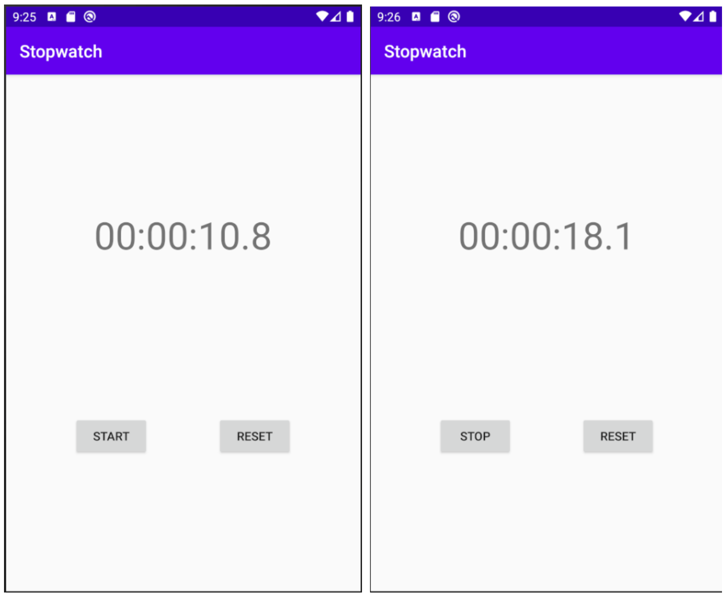
Assignment

Create a stopwatch app that:

- (2 pts.) Displays the current elapsed time in the format "HH:MM:SS.T", where HH is the elapsed hours displayed as two digits, MM is minutes, SS is seconds, and T is tenths of a second. For example, "00:05:17.4" means five minutes and 17.4 seconds have elapsed.
- (2 pts.) Presents Start and Reset buttons when the stopwatch is stopped, and Stop and Reset buttons when the stopwatch is running. These buttons should function as described in the stopwatch behavior section above.
- (3 pts.) Uses ViewModel to persist the stopwatch state and UI across configuration changes such as a device rotation.
- (3 pts.) Uses LiveData to update the elapsed time display every 10 ms.

Sample UI

The below figures show a sample UI. You are free to customize this, but your UI must meet the requirements listed above.



Suggestions and Hints

- Use a Handler to handle the timing. We particularly recommend postAtTime(), which accepts a Runnable (an object that implements the Runnable interface). This will call the run() method of the Runnable at the specified time.
- If going this route, use SystemClock.uptimeMillis() to get the current system time.
- To use a Handler to periodically perform an action, you can post the Runnable again from within the Runnable's run() method.
- To stop the stopwatch, you can remove all callbacks from the Handler (canceling pending Runnables), or you can check the state of the stopwatch before posting the next Runnable.
- Use a MutableLiveData<Long> for the elapsed time.
- To translate a length of time in milliseconds to hours/minutes/seconds, use simple division.
- Use a formatted string resource to display the elapsed time.
- While Jetpack/androidx is versioned separately from the Android API Level, it still has versions. The documentation on ViewModels assumes you are using androidx.lifecycle version 2.2.0 or higher. Your project may load an older version by default. If you get an error with the ViewModelProvider constructor, try following the steps at https://developer.android.com/jetpack/androidx/releases/lifecycle#declaring_dependencies to make sure Gradle grabs the newest version of androidx.lifecycle from the Google Maven repository.