CSCI 448 – Lab 03B
Wednesday, February 01, 2023
LAB IS DUE BY **Tuesday, February 14, 2023 11:59 PM**!!

With Quizler now in a functional form, the Little Green Games development team is getting ready for the big release party! The Chief Technology Officer comes into the room right after the dev team heads out to buy streamers. The CTO then tells you the app isn't quite ready for launch yet. The QA Team has detected a bug in the app but needs more logging to help reproduce and trace the error.

Step 1 – Log the Activity

Knowing the Activity is the life blood of our app, we will log its lifecycle and interactions with the Operating System.

Begin by creating the static `companion object` to initialize a private constant named `LOG_TAG`. Set the value to be `448.MainActivity`. Now, when overriding `onCreate()` and after the `super.onCreate()` has been called, log a debug message stating that the function had been called.

```
Log.d(LOG_TAG, "onCreate() called")
```

Do the same for the following lifecycle methods:

- `onStart()`
- `onResume()`
- `onPause()`
- `onStop()`
- `onDestroy()`

To gain even more insight to the process, also override and log the following Activity callbacks as well:

- `onContentChanged()`
- `onPostCreate(Bundle?)`
- `onPostResume()`
- `onAttachedToWindow()`
- `onEnterAnimationComplete()`
- `onDetachedFromWindow()`

When a device is connected to Android Studio, open the Logcat tab. Ensure the filter is setup for your package and the prefix for your tag.

```
package:mine tag:448.
```

Build, deploy, and run the app. Follow the flow of the Activity lifecycle as you open and close the app via the device.

Next, inspect the composables.

Step 2 – Log the Composables

For all of our composables listed below, create a top level private constant named LOG_TAG. The value will begin with the 448. Prefix and end with the name of the composable.

```
// replace XYZ with composable name

private const val LOG_TAG = "448.QuestionXYZ"

@Composable
fun QuestionXYZ(...) {
  Log.d(LOG_TAG, "Composing ABC") // replace ABC with information
  ...
}
```

Part 2.I – `QuestionScreen`

In `QuestionScreen`, simply log that the screen is being composed.

Part 2.II – `QuestionScoreText`

In `QuestionScoreText`, log the score that is being displayed.

Part 2.III – `QuestionDisplay`

In `QuestionDisplay`, log the question object that is being displayed.

Part 2.IV – `QuestionButton`

In `QuestionButton`, log the button text that is being displayed.

Part 2.V – View the Logs

For all the above, feel free to add finer logging down each composable tree.

Again build, deploy, run. This time when viewing the logs, interact with the buttons to change the state. Observe the recomposition flow.

(*Note: depending on how you have set your composables, modifiers, and potentially completed Lab03A XC then the recomposition flow may vary from implementation to implementation. Seemingly stable composables are still recomposed since the sizing/placement needs to be recomputed.*)

Currently, the Activity lifecycle and View composables are being logged. The flow of state downward is logged during recomposition. Now to track the flow of events upwards.

Step 3 – Log the View Model

Add the `companion object` with the private constant `LOG_TAG` to `QuestionViewModel`. Log the following events:

- Add an `init` block for when the View Model is created
- Log every member function (public or private)

Again again build, deploy, run. This time when viewing the logs, interact with the buttons to trigger events. Observe the event callback and state recomposition unidirectional flow.

Step 4 – Log the Model

For completion, in `QuestionRepo` add an `init` block to log when the repository is initialized. Note, since the repository is already a singleton (and thus static) there is no `companion object`. Create the `LOG_TAG` as a private constant.

Again again again build, deploy, run. This time when viewing the logs, interact with the buttons to trigger events, rotate the device, leave the app, come back to the app, destroy the app, restart the app – all interactions you can think of. Observe everything.

Step 5 – Deploy Your App

With the log trace in place, you send Quizler back to the QA Team to create the next ticket. Though, you already have an idea of what they'll tell you. Make your prediction in the Canvas survey **Lab03B QA Team Prediction** with the provide access code of **`LOG_TAG`**.

LAB IS DUE BY **Tuesday, February 14, 2023 11:59 PM**!!