**Retrofit and Coroutines**

* RetrofitPractice: uses **Retrofit**
* Old **original way** of Retrofit.
* Uses **Jason Array**
* JSONApp: uses **Retrofit**
* Uses **Old Original way**
* Uses @**Header**
* Uses **JsonObject** to call the data
* Uses **keySet** to get all Keys from JsonObject
* CoroutinesApp: uses **Coroutines**
* Uses **Delay**
* **Make the data updated from the internet infinite way** util the user press Stop
* Coroutines\_app-main: uses **Coroutines**
* **Simple example** of how to use Coroutines
* CurrencyConverter-master: uses **Retrofit**
* Uses **Old Original way**
* Uses @**Header**
* getJSON-master: uses **Coroutines**
* Uses **getString(“name”)** to get specific value
* Uses **.keys()** for JSONObject to get all keys name
* Uses **2 Ways of calling data**. My way and teacher way
* My Way is to call all the data as JSONObject and use keys to access each
* Teacher way is to access the data that he chooses directly
* RecipeApp: uses Ritrofit
* Uses new short way
* Call data from starting array
* Uses get, to download data
* Uses post, to upload data
* Uses data class and parameters
* DebugChallenge3: Uses Coroutines
* Get the data from started array
* Get the data from array inside Json Object
* Lots of arrays in the data
* Uses TOAST inside the catch to show errors
* HeadsUp! & Prep: Uses Retrofit
* Uses super new short way
* Uses data class and variables as constraint
* Uses GET to call the data
* Uses Post to Add (Upload) new Data
* Uses Put to Update the Data (Change server data)
* Uses Delete to Delete Data (Delete from Server)
* Uses device rotation to update the data

**Activity Lifecycle**

* ActivityLifecycleApp:
* Uses **all 6 functions** to check user location
* Uses **Log.d()**
* CallbacksApp:
* Uses **all 6 functions** to check user location
* Uses **Log.d** and Toast
* Uses **3 classes** to track
* DebuggingChallenge2:
* Uses **onStop to save the data**

**Recycler View**

* ActivivityLifecycleApp: Uses **Binding** way
* **Simple** way
* Using **onItemClickListener**
* Uses **2 array as parameters**
* Uses **cardView**
* DebuggingChallenge2: **didn’t use any way**
* **Wasn’t created by me**
* Uses **onItemClick**
* Uses **separate class** to initialize UI elements for Recycler View
* Uses **ArrayList<ArrayList<Any>>**
* getJSON-master: using **Binding** Way
* Uses 2 RecyclerViews
* RecipeApp: using Binding Way
* Uses onItemClickLestener
* Take 2D array
* Show data that got from internet
* Uses CardView
* GuessThePhrase & 2: using Extensions way
* Pass Array and Int variables
* Color the TextView
* Uses Smooth Scroll Position to go to the last of the RecyclerView
* DebugChallenge3: Using Binding Way
* Use fun inside the RVAdapter to update the data
* Uses 2D array
* HeadsUp! & Prep: uses Binding Way
* HeadsUp! Uses onItemClickListener (interface inside the Adapter class)
* Uses 2D Array of 5 Elements
* EncoderDecoder: uses Binding Way
* Color the entered text

**Toast and Snackbar**

* ActivivityLifecycleApp: **Using Both**
* Using **Toast inside override fun** and pointing to the class.
* ButtonApp: Uses **Snackbar**
* Uses **setPadding**
* CallbacksApp: Uses **Toast**
* DebugOne: Uses **Toast**
* RecipeApp: Uses **Toast**
* HeadsUp! & Prep: Uses Toast

**Alert Dialog**

* AlertDialog: Simple Tutorial for alert dialog
* Uses **input**
* Uses **positive and negative buttons**
* DebuggingChallenge2:
* Uses **separate class to initialize Entry UI** for the dialog
* Uses **Pair method** to connect 2 xml files
* GuessThePhrase & 2:
* Uses positive button
* Uses cancelable = false
* Uses reCreate inside the positive button
* HeadsUp! & Prep: Uses Progress Dialog
* Title changes continually with the timer

**onSaveInstanceState, onRestoreInstaceState**

**and SharedPreferences**

* Calculator: Uses **onSave, onRestore**
* DebuggingChallenge2: Uses **SharedPreferences**
* Saving and Calling **ArrayList** using For loop
* GuessThePhrase & 2: **Uses onSave, onRestore** but **2 uses both**
* Restore data inside onCreate fun
* Save Int, String, Char array and String array list

**Intent**

* CallbacksApp: **Simple use** of how Intent works
* RecipeApp
* Uses intent to pass data to another class
* Uses getString to receive data
* Pass 2D array as String
* HeadsUp! & Prep
* Uses intent to pass data to another class
* Uses getString to receive data
* Pass 2D array of 5 Elements as String and one will change to Int

**Spinner**

* JSONApp
* Uses **MutableSet** converted to array to show the data
* CurrencyConverter-master
* Uses **Simple Array** to show the data

**Libraries and Others**

* EncoderDecoder:
* Uses Encoding and Decoding Characters in String by shifting 13 letter

HellowWorldApp2Practice