**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
* Flickr\_Browser\_App-master: Uses Retrofit
* Uses Json Plugin
* Uses Get to call Data
* Uses Url to add variables to the link
* FlickrBrowserApp: Uses Coroutines
* Get data as Json Object
* FlickrBrowserAppRetrofit: Uses Retrofit
* Uses Get to call Data
* Uses Url to add variables to the link
* Uses Query to add variable to the link
* KotlinWeatherApp-master: Uses **Coroutines**
* Uses Date Format to Call and format the Date from the internet
* ParsingLocalJSONFile: Don’t use any
* Parsing Json File from created in assets local file
* POSTRequestAppRevisited: 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)
* POSTRequestPractice: Uses Retrofit
* Uses @SerializedName in the data class
* Uses data class and variables as constraint
* Uses GET to call the data
* Uses Post to Add (Upload) new Data
* RetrofitDemo-master: Uses Retrofit
* Uses GET to call the data
* Get Only 2 String from Long JSON Object (List Of Json Object)
* SingletonPractice: Uses Retrofit
* Uses Old Log Weird Way
* Get Currency from API
* Uses **JsonObject** to call the data

**XML**

* Reddit\_Xml\_App-main: Uses Retrofit
* Read Data From internet
* Read Data as XML File
* Uses Long Way
* Uses SimpleXmlConverterFactury to Handle the Data
* FlickrBrowserAppXML: Uses Retrofit to get Data as XML
* Uses Get to call Data
* Uses Url to add variables to the link
* Uses ElementList
* Uses Path to go throw Element
* Uses Attribute to get the Data from inside the Tag
* Uses SimpleXmlConverterFactury to Handle the Data
* RSSFeedPractice:
* Uses 2 way of parsing the data
* Uses Almin Way
* Uses Ranjith Way
* Uses One Way for Fetching the Data (AsyncTask<Void, Void, ArrayList<Group>>()) for Both Ways
* Uses Interface class and another class and fun to get array from another class

**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**
* StudyApp
* Uses onResume to refresh the recycler view data
* onResume work when the other class uses finish() and back to this class

**Recycler View and Grid**

* 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
* Flickr\_Browser\_App-master: Uses **Binding Way**
* Uses onClickListener for the RV inside the adapter class
* Uses Start Activity inside the RV adapter class
* FlickrBrowserApp, FlickrBrowserAppRetrofit and FlickrBrowserAppXML
* Uses **RV Binding Way**
* Uses **Grid**
* Both have **on item Click Listener**
* Both Have **Favorite CheckBox**
* FlickerBrowserApp: Uses Custom Toast inside the Adapter
* ParsingLocalJSONFile:
* Uses onClickListener
* HeadsUpSQLiteSaveOnly
* 2 classes share same RV
* Both use onClickListener
* StudyApp and NotesAppRoom and StudyAppRoom **:**
* StudyApp and StudyAppRoom**:** Start activity from RV Adapter class
* StudyApp and StudyAppRoom: **Uses Extensions way**
* Uses Toast inside Adapter class
* Getting Context in adapter class
* Uses image on click listener inside adapter class
* StudyAppRoom Uses 2 RV Adapter with 1 Xml file
* NotesAppRoom and StudyAppRoom: Delete Data in SQLite inside RV Adapter
* RSSFeedPractice:
* Uses 2 Recycler View
* Uses **Search View**
* NotesAppViewModel:
* **Passes Main Activity in the adapter class so it can call functions from adapter located in main**

**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
* NumbersGameApp: Uses Snackbar
* **Custom Toast**:
* NotesAppRoom
* StudyAppRoom
* StudyApp
* HeadsUpSQLiteSaveOnly
* HeadsUpRoom
* SignUpAndSignIn

**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
* HeadsUpSQLiteSaveOnly
* Uses Another XML that used to edit and delete user as Alert Dialog
* Uses regular alert dialog inside custom alert dialog

**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
* FlickrBrowserApp, FlickrBrowserAppRetrofit and FlickrBrowserAppXML
* Uses intent to pass data to another class
* Uses getString to receive data

**Spinner**

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

**SQLite and Room and View Model**

* NotesApp & HeadsUpSQLiteSaveOnly & GuessThePhrase2:
* Save and Getting
* Uses lazy
* Uses Version 2
* Update and Delete: NotesApp, HeadsUpSQLiteSaveOnly
* NotesApp-master
* **New Different Way**
* Save and Getting
* Using Companion object
* SavingData
* Save and Getting
* Getting Specific Location from Someone Name
* Update
* Delete
* Version 2
* SignUpAndSignIn
* Save and Get
* Uses Different Ways of Getting Data
* Uses 2 Data Check at the same Time (UserName – Password)
* Tried To Encrypt password but failed But the encrypting class still there
* StudyApp
* Save, Get, Edit and Delete
* Save Data from Set Data Array
* Delete from inside RV Adapter Class
* NotesAppRoom: **Uses Room**
* **Uses Mr.Ranjith Way**
* Uses allowMainTheadQueries to do function without Coroutines
* Save, Get, Edit and Delete
* Delete and Edit from inside RV Adapter Class
* Don’t Use Coroutines And Working good
* HeadsUpRoom: **Uses Room**
* **Uses Mr.Almin Way**
* New versions destroy last version
* Save, Get, Edit and Delete
* **Uses Coroutines** for functionality
* **Uses CoroutinesScope and async**
* Uses **one Data Class for API Data and SQLite Data**
* Call Data from API and Save it in SQLite
* StudyAppRoom **:Uses Room**
* **Uses Mr.Almin Way**
* Save Data from Set Data Array
* New versions destroy last version
* Save, Get, Edit and Delete
* **Uses Coroutines** for functionality
* **Uses CoroutinesScope and async**
* **Create 2 tables**
* **Uses 2 Data classes**
* **Complicated**
* **RecipeAppRoom: Add ViewModel**
* **Uses Mr.Almin Way in Room**
* **Uses My Super Short Way In View Model**
* New versions destroy last version
* Save, Get, Edit and Delete
* **Uses Coroutines** for functionality
* **Uses CoroutinesScope and async**
* Uses **one Data Class for API Data and SQLite Data**
* Call Data from API and Save it in SQLite
* Saved more that 160 data from api
* Uses auto generate pk
* Getting the pk from the API to save it to avoid recopy again
* **FlickrBrowserApp: Uses Room**
* **Uses Mr.Almin Way**
* New versions destroy last version
* Save, Get and Delete
* **Uses Coroutines** for functionality
* **Uses CoroutinesScope and async**
* Uses **one Data Class for API Data and SQLite Data**
* Only Save favorite list onStop fun
* Delete the unfound image in the new favorite list from database
* Add new favorite image to database
* **NotesAppViewModel: Uses Room** with **View Model** and **Live View**
* Uses My short Way
* Uses Lazy to Call ViewModel class
* Uses Observe for live data
* SQLiteDemo-0dd: Uses Both SQLiteHelper and Room

**Libraries and Others**

* EncoderDecoder:
* Uses Encoding and Decoding Characters in String by shifting 13 letter
* Flickr\_Browser\_App-master:
* **Uses Menu** as Down Bar Menu
* **Uses DiffUtil**
* **Uses Object Class**
* CompanionObjectsApp:
* **Uses Companion Object**
* Glide Library: To Download images from internet and display it
* Flickr\_Browser\_App-master
* FlickrBrowserApp
* FlickrBrowserAppRetrofit
* FlickrBrowserAppXML
* ParsingLocalJSONFile
* FlickrBrowserApp, FlickrBrowserAppRetrofit and FlickrBrowserAppXML
* Uses Menu
* 2 items inside Menu
* Item inside menu always shown
* One item for favorite, second item to change view
* Change item icon when menu clicked
* YouTube Library: To Play Videos from YouTube
* KotlinYouTube-master
* Libraries:
* Uses Styleable Toast Library
* Uses Slid Activity Library: For Sliding the activity to close it
* Uses Alert Library: New Way to Show Alert with icon
* Uses Color Picker Library: To Let the user choose the color he wants
* POSTRequestAppRevisited
* Uses onConfigurationChanged to check device rotation

SQLiteRoomNameLocation-master