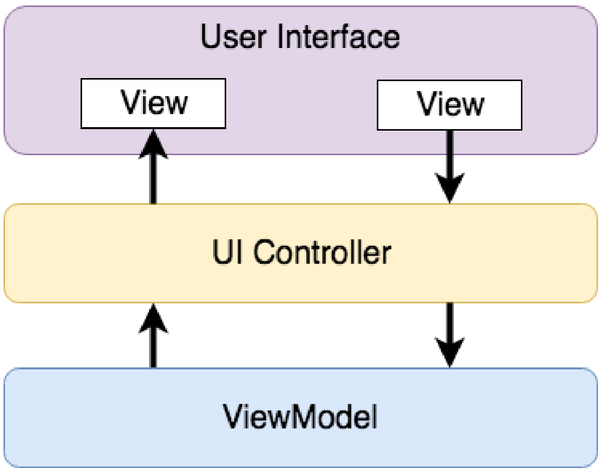
LAB 6 – Modern Android Architecture

At the end of this lab, students should be able to:

- Make use of the ViewModel component.

ViewData Model

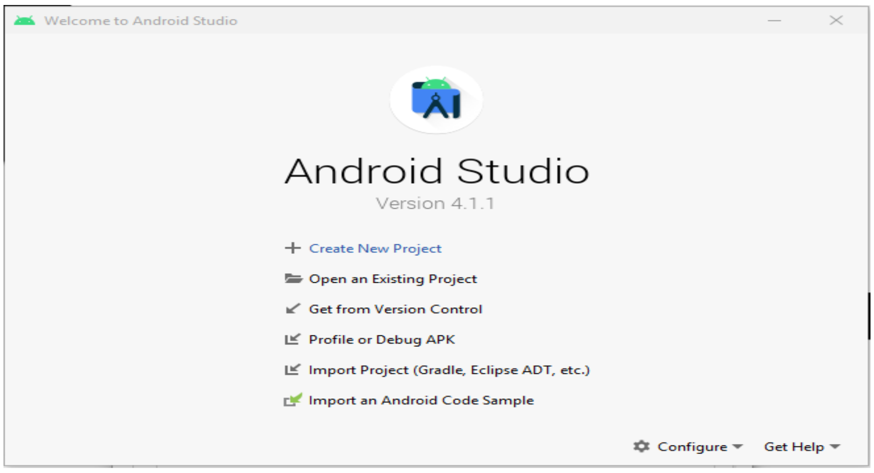
Until the last year, Google has tended not to recommend any particular approach to structuring an Android app.
That has now changed with the introduction of Android Jetpack which consists of a set of tools, components, libraries and architecture guidelines. Google now recommends that an app project be divided into separate modules, each being responsible for a particular area of functionality otherwise known as “separation of concerns”.
In particular, the guidelines recommend separating the view data model of an app from the code responsible for handling the user interface. In addition, the code responsible for gathering data from data sources such as web services or databases should be built into a separate repository module instead of being bundled with the view model.
Android Jetpack includes the Android Architecture Components which have been designed specifically to make it easier to develop apps that conform to the recommended guidelines.



ViewModel Tutorial in Android Studio- An Example

a) Creating the ViewModel Example Project

i. Go to File -> New Project



ii. Choose **Fragment + ViewModel** template, then click Next

iii. Enter *ViewModelDemo* into the Name field and specify com.ebookfrenzy.viewmodeldemo as the package name.
Change the Minimum API level setting to API26: Android 8.0(Oreo) and the Language menu to Java.
Click Finish.

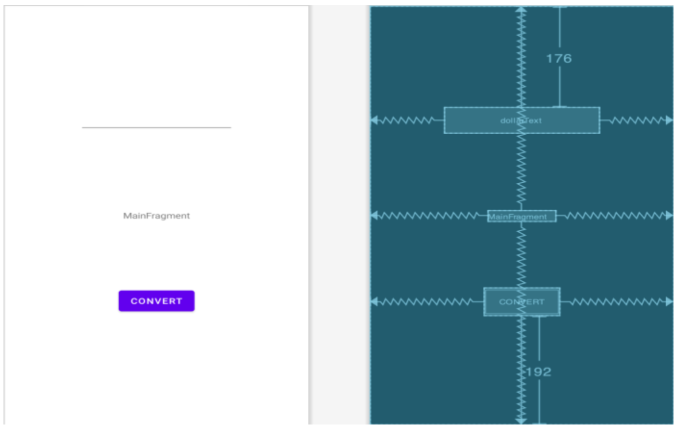
b) Designing the Fragment Layout

The next step is to design the layout of the fragment. Locate the *main_fragment.xml* file in the Project tool window and double click on it to load it into the layout editor.

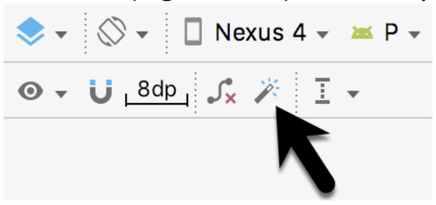
i. Select the select the existing TextView widget and use the Attributes tool window to change the id property to *resultText*.

ii. Drag a Number (Decimal) view from the palette and position it above the existing TextView. With the view selected in the layout refer to the Attributes tool window and change the id to *dollarText*.

iii. Drag a Button widget onto the layout so that it is positioned below the TextView, and change the text attribute to read “Convert”. With the button still selected, change the id property to *convertButton*. At this point, the layout should resemble that illustrated in Figure below:



iv. Click on the *Infer constraints* button (Figure 33-4) to add any missing layout constraints:



v. Finally, click on the warning icon in the top right-hand corner of the layout editor and convert the hardcoded strings to resources.

c) Implementing the View Model

With the user interface layout completed, the data model for the app needs to be created within the view model.

i. Open *MainViewModel.java* file and modify the class so that it reads as follows:

```
package com.ebookfrenzy.viewmodeldemo.ui.main;
import androidx.lifecycle.ViewModel;
public class MainViewModel extends ViewModel {
private static final Float usd_to_eu_rate = 0.74F;
private String dollarText = "";
private Float result = 0F;

public void setAmount(String value) {
this.dollarText = value;
result = Float.parseFloat(dollarText)*usd_to_eu_rate;
}
public Float getResult()
{
return result;
}
}
```

The class declares variables to store the current dollar string value and the converted amount together with getter and setter methods to provide access to those data values. When called, the *setAmount()* method takes as an argument the current dollar amount and stores it in the local *dollarText* variable. The dollar string value is converted to a floating point number, multiplied by a fictitious exchange rate and the resulting euro value stored
in the *result* variable. The *getResult()* method, on the other hand, simply returns the current value assigned to the *result* variable.

d) Modifying the Fragment

The fragment class now needs to be updated to react to button clicks and to interact with the data values stored in the ViewModel. The class will also need references to the three views in the user interface layout to react to button clicks, extract the current dollar value and to display the converted currency amount.

i. Open MainFragment.java, odify the *onActivityCreated()* method to obtain and store references to the three view objects as follows:

```
.
.
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import com.ebookfrenzy.viewmodeldemo.R;
public class MainFragment extends Fragment {
private MainViewModel mViewModel;
private EditText dollarText;
private TextView resultText;
private Button convertButton;
.
.
@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
super.onActivityCreated(savedInstanceState);
mViewModel = new ViewModelProvider(this).get(MainViewModel.class);
dollarText = getView().findViewById(R.id.dollarText);
resultText = getView().findViewById(R.id.resultText);
convertButton = getView().findViewById(R.id.convertButton);
}
.
.
```

In the previous lab, the onClick property of the Button widget was used to designate the method to be called when the button is clicked by the user. Unfortunately, this property is only able to call methods on an Activity and cannot be used to call a method in a Fragment. To get around this limitation, we will need to add some code to the Fragment class to set up an onClick listener on the button. The code to do this can be added to the *onActivityCreated()* method of the *MainFragment.java* file as follows:

```
@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
super.onActivityCreated(savedInstanceState);
mViewModel = new ViewModelProvider(this).get(MainViewModel.class);
dollarText = getView().findViewById(R.id.dollarText);
resultText = getView().findViewById(R.id.resultText);
convertButton = getView().findViewById(R.id.convertButton);
convertButton.setOnClickListener(new View.OnClickListener()
{
```

```
                @Override
                public void onClick(View v) {
                }
        });
        }
```

With the listener added, any code placed within the *onClick()* method will be called whenever the button clicked by the user.

e) Accessing the ViewModel Data

When the button is clicked, the *onClick()* method needs to read the current value from the EditText view, confirm that the field is not empty and then call the *setAmount()* method of the ViewModel instance. The method will then need to call the ViewModel's *getResult()* method and display the converted value on the TextView widget.
Since LiveData is not yet being used in the project, it will also be necessary to get the latest result value from the ViewModel each time the Fragment is created.

i. Remaining in the *MainFragment.java* file, implement these requirements as follows in the *onActivityCreated()* method:

```
@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
super.onActivityCreated(savedInstanceState);
mViewModel = new ViewModelProvider(this).get(MainViewModel.class);
dollarText = getView().findViewById(R.id.dollarText);
resultText = getView().findViewById(R.id.resultText);
convertButton = getView().findViewById(R.id.convertButton);
resultText.setText(mViewModel.getResult().toString());
convertButton.setOnClickListener(new View.OnClickListener()
{
@Override
public void onClick(View v) {
if (!dollarText.getText().toString().equals("")) {
mViewModel.setAmount(dollarText.getText().toString());
resultText.setText(mViewModel.getResult().toString());
} else {
resultText.setText("No Value");
}
}
});
}
```

f) Testing the Project

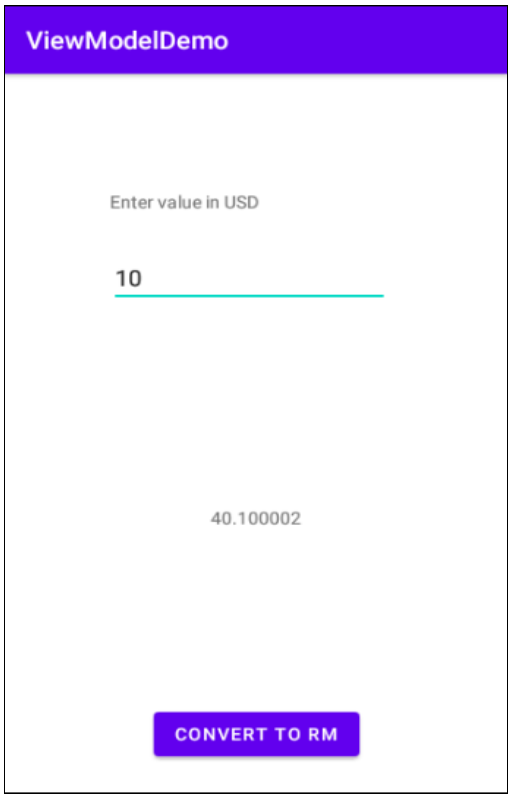
With this phase of the project development completed, build and run the app on the simulator or a physical device, enter a dollar value and click on the Convert button. The converted amount should appear on the TextView indicating that the UI controller and ViewModel re-structuring appears to be working as expected.

i. When the original AndroidSample app was run as in previous Lab, rotating the device caused the value displayed on the *resultText* TextView widget to be lost. Repeat this test now with the ViewModelDemo app and note that the current euro value is retained after the rotation. This is because the ViewModel remained in memory as the Fragment was destroyed and recreated and code was added to the *onActivityCreated()*

method to update the TextView with the result data value from the ViewModel each time the Fragment re-started.

EXERCISE

i. Change the conversion from USD to EU to USD to RM
ii. Update the layout as below
iii. Please screenshot your output and submit in MMLS


ViewModelDemo
Enter value in USD
10
40.100002
CONVERT TO RM