# UNIVERSITI MALAYSIA SARAWAK

94300 Kota Samarahan

Sarawak

**Faculty of Cognitive Sciences & Human Development**


**KMK3433 : Mobile Computing (G01)**

**Pair Assignment – 10%**

*Design and Development of Calculator App*

**Semester –** 2, 2021/2022

**Lecturer –** Dr Rehman Ullah Khan

**Date –** 16 April 2021


**Full name & Matric No –**

Chiang Rui Bin 65642

Joanne Ling Li Yi 66281

# Introduction

Android studio is a well-integrated platform for developers worldwide to develop the android-based application. This environment's foundation is based on a Java integrated development, named IntelliJ IDEA ("Android studio", n. d.). It is multifunctional, providing users with a variety of useful features like visual layout editor, giving them the capability to build application for all devices, instant application run and most importantly, smart code editor ("Top 10 features of Android Studio", 2018). This includes Java programming language whereby it is an object-oriented programming language whereby it provides a fast environment for application development with high reliability ("What is Java?", n. d.). Due to its not-so-complex syntax, it is considered to be high-level language as humans are able to understand and execute it easier (Leahy, 2019). Despite the famously utilized Java programming language, Kotlin has recently been introduced to Android Studio. In accordance with the claim by Heller (2020), Kotlin serves as a general-purpose programming language that is free, open-source, and it focuses more on clarity as well as tooling support. Depending on the users' preference on the language used for application development, both do serve their pros and cons but the code must be run on an emulator before the publication in order to test its stability and whether it is workable. This Gradle-based build system is a virtual device that communicates with other components or widgets users have included in the designated layout like, for instance, buttons, text view, switch, and others ("Emulator", n. d.). Doing so, it saves users' time on connecting their smart android devices to the computer over USB since it is faster to send the data to emulator than through USB. Also, it comes with a pre-defined configuration for those devices.

In this project, an android-based calculator is development in the Java programming language. The calculator can perform a basic arithmetic operation including addition, subtraction, multiplication as well as division. Also, it supports clear operation where users can swipe off the screens with just a single click. It provides a sense of transparency for users are able to view what actions are made, what buttons are clicked (Input), and the output of it (Result) with great clarity.

# Description of the Application

## 1. Overall Appearance

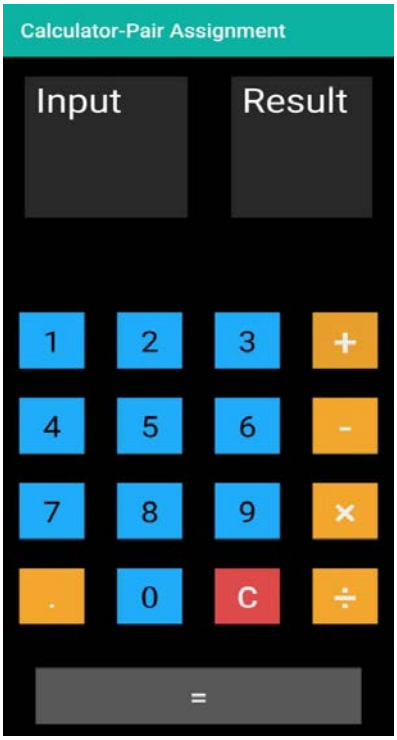



*Figure 1 shows the default screen of the calculator*

activity_main.xml | MainActivity.java | AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.pairassignment">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Calculator-Pair Assignment"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="MissingClass">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

In accordance with Figure 1, it shows the overall default screen of the calculator. At the top of the screen, the calculator is given a title “Calculator-Pair Assignment”. This is done by adding a label in the “*AndroidManifest.xml*” file at line 9. This allows the users to customize the title that will be displayed on the screen.

```
android:label="Calculator-Pair Assignment"
```

AndroidManifest.xml file locates in the root directory, acting as a foundation or empty canvas of the application before the developers are able to do the programming task. It contains all the essential fundamental components as the users pack and build the APK, including services, activities, content providers (Manifest), and broadcast receivers (Intent Filter). If one of these components go missing, the application will not run. Within the manifest elements, the users are given the choice to customize the icons as well as the theme of the application and this is done by declaring sub-elements of the application element (<application>), starting from line 6 to line 20.

## 2. Visual Layout

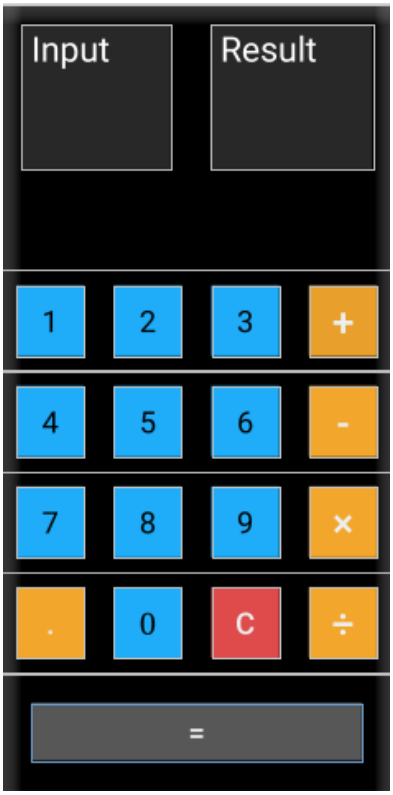
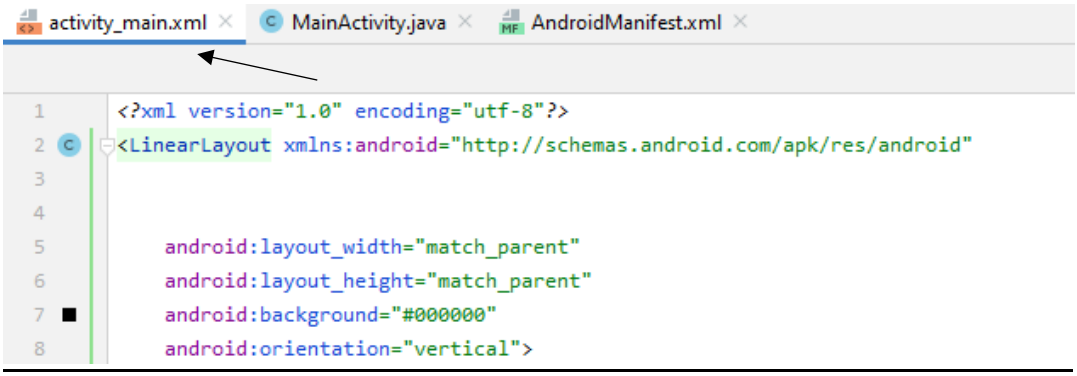



*Figure 2 shows the layout of the screen*




In order to determine the layout as well as the position and orientation of the others components in a single screen, the application makes use of **linear layout** at line 2. Linear layout allows the components to be oriented in a single direction, either horizontal or vertical. This can be found in the file named "*activity_main.xml*".

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

To specify the layout direction, orientation attribute has to be called and in this call, the calculator application is positioned in a vertical manner overall (Line 8). This implies that the whole screen will have only one child per row.

```
android:orientation="vertical"
```

At line 5 and 6, the users are allowed to customise their layout's width and height, either match_parent or wrap_content. In this case, the view is set to be as big as the parent screen (match_parent). This implies that the child view shares the same amount of space with the vertical parent screen. Otherwise, wrap_content will only be used when the users specify a particular amount of space they want the components to be fit in or occupy only.

```
android:layout_width="match_parent"
android:layout_height="match_parent"
```

By default, the layout's background is set to be white but our application has changed it to black. This is done by calling **android:background="#000000"** while the numbers represent as the color code. Whilst **#FFFFFF** stands for white code.

```
11      <LinearLayout
12          android:layout_width="match_parent"
13          android:layout_height="243dp"
14          android:layout_weight="0.3"
15          android:orientation="horizontal">
```

What differs with the parent layout is that the **layout height** is customised, which is to be *243dp* (Line 13). It is to ensure that there is enough space between the text-views and the components situated at the button of them, consisted of the functional buttons. Coming along with the option of determining **layout weight**, it allows users to assign value to the screen space, meaning that the value affects how much space it should occupy and how many children attributes it can be fit into (Line 14). The larger the value, the more child views can occupy. On the other hand, the **width** of this layout remains to be match_parent so that the input or the result does not go beyond the viewable screen (Line 12).

```
android:layout_width="match_parent"
android:layout_height="243dp"
android:layout_weight="0.3"
```

Based on Figure 2, it is deemed that there are a total of 6 horizontal layout. This implies that the components or attributes cannot be aligned in a vertical manner no matter how spacious the defined areas are. To achieve this, every linear layout has a sub-element named android: orientation and this is the place where developers can define its alignment. For example, the first horizontal layout contains two text-view components where the numbers appear to be once the users click the respective buttons (Line 15).

```
android:orientation="horizontal">
```

```
48      <LinearLayout
49          android:layout_width="match_parent"
50          android:layout_height="wrap_content"
51          android:layout_weight="0.2"
52          android:orientation="horizontal">
```

Other than the horizontal layout for the text-views, the rest of them are for buttons (Each row contains four buttons except the horizontal layout of the total button). The common features across all of them are that their width matches with their parent view *(match_parent at line 49)*, sharing with a floating value of 0.2 for the layout (Line 51). If the **layout weight** is 0, it means that the view cannot be stretched. Since the **layout height** is set to be *wrap_content*, hence the buttons shouldn't be placed outside this limited space (Line 50).

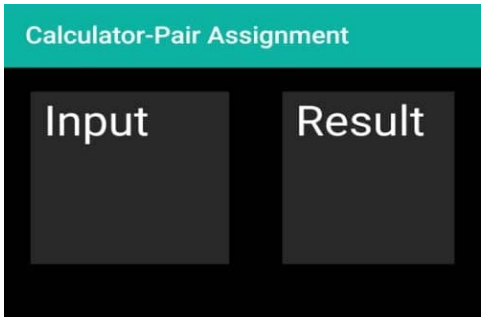



*Figure 3 shows the text-views for input and result*

```
17      <TextView
18          android:id="@+id/display"
19          android:layout_width="150dp"
20          android:layout_height="145dp"
21          android:layout_margin="20dp"
22          android:background="#292828"
23          android:hint="Input"
24          android:paddingLeft="10dp"
25          android:textColor="#FFFFFF"
26          android:textColorHint="#FFFFFF"
27          android:textSize="35sp"
28          android:typeface="sans" />
```

Under the text-view attribute, there are a variety of sub-elements in order to customize it. At Line 18, **android:id="@+id/display"** functions in a way that the text-view is assigned with a specific id so that it can be referred in the *"MainActivity.Java"* file. It is where all the functionalities are coded. In this case, the input text-view is assigned with ***display*** id and this

view can be found and assigned by calling **`edittext1 = (TextView) findViewById(R.id.display);`**

At line 21, the **margin** of the layout is defined. This implies that this rectangular view, together with the content inside, is pushed afar from the boarder of its horizontal layout that matches with its parent layout. The larger the margin value, the further the view it is from the other views.

```
android:layout_margin="20dp"
```

To allow the users to have a better understanding regarding what the text-views do, it is given with a hint instead of text (Line 23). It is because hint will not affect the users as they try to input the values, meanwhile they have to manually clear of the text if *setText()* function is not utilized. In this case, the input text-view is hinted with "Input".

```
android:hint="Input"
```

Talking about the **background colour** of the input text-view, it is coded with #292828 at line 22.On the other hand, the text as well as the hint is given a new parameter call **padding**. It is used to position them in a standard place so that they will not go out of the order. For example, the input text-view is given *10dp* for the padding left parameter (Line 24), meaning that the content inside is pushed 10dp towards its centre and the movement starts from the left side. It is similar with margin, just that they are different in where the change in position happens with respective to that text-view. Regarding the **colour** of the text and hint (Line 25 and 26), they are assigned with the white colour code of #FFFFFF and the **text size** is set to be *35sp* (Line 27). The unit *sp* implies that the pixel is scaleale whilst *dp* means that the pixel of the typeface depends on the density of the screen. At line 28, the **typeface** sub-elements is called where the users can specified the type of the text or hint. In this case, we set it to *Sans font*.

```
android:paddingLeft="10dp"
android:textColor="#FFFFFF"
android:textColorHint="#FFFFFF"
android:textSize="35sp"
android:typeface="sans" />
```

```
30      <TextView
31          android:id="@+id/display2"
32          android:layout_width="158dp"
33          android:layout_height="145dp"
34          android:layout_margin="20dp"
35          android:layout_weight="1"
```

Overall, the input and result text-views share the similar features except the fact that a sub-element regarding the **layout weight** is called, carrying a value of 1 (Line 35). This exerts more restriction on the amount of space the content can occupy. Also, the main reason of the restriction is that the input text-view normally needs more space, supporting users to key in the numeric value with a longer length.

**`android:layout_weight="1"`**

Generally, the difference that is shown among the buttons includes its ID and background colour, text size, text colour, and the text itself.

```
54      <Button
55          android:id="@+id/b1"
56          android:layout_width="wrap_content"
57          android:layout_height="match_parent"
58          android:layout_margin="15dp"
59          android:layout_weight="0.25"
60          android:background="#1FACF8"
61          android:text="1"
62          android:textColor="#000000"
63          android:textSize="30sp"
```

```
90      <Button
91          android:id="@+id/badd"
92          android:layout_width="wrap_content"
93          android:layout_height="match_parent"
94          android:layout_margin="15dp"
95          android:layout_weight="0.25"
96          android:background="#E89F2C"
97          android:text="+"
98          android:textColor="#ecf0f1"
99          android:textSize="40sp" />
```

```
275     <Button
276         android:id="@+id/buttoneql"
277         android:layout_width="375dp"
278         android:layout_height="58dp"
279         android:layout_margin="30dp"
280         android:layout_weight="0.25"
281         android:background="#E2636363"
282         android:text="="
283         android:textColor="#ecf0f1"
284         android:textSize="30sp" />
```

### 3. Main Activity (Java Code)

To make the application functional, the main function source code was contained in the .java source files of the project. By default, upon creating a project, it includes a MainActivity.java source file which acts as an activity class that runs when running the application.

The first few lines of the MainActivity.java is about the package objects for this application. Importing and declaring packages that contain information for implementation and specification of the package is done.

```
1   package com.example.pairassignment;
2
3   import androidx.appcompat.app.AppCompatActivity;
4   import android.os.Bundle;
5   import android.view.View;
6   import android.widget.Button;
7   import android.widget.TextView;
8
9   import java.text.DecimalFormat;
10  import java.util.Locale;
11
12  import static java.lang.String.*;
```

The declaration of the main function is starting at the Line 15. The main class function contains the declaration of the variables used for the button and function of the application and text viewer function. The main class function can be written as,

```
public class MainActivity extends AppCompatActivity {
```

Followed by the declaration of variables which will be used to map into the layout design of the application.

```
17      double in1 = 0, i2 = 0;
18      TextView edittext1,edittext2;
19      boolean Add, Sub, Multiply, Divide, deci, dot_clicked;
20      Button button_0, button_1, button_2, button_3, button_4, button_5, button_6, button_7, button_8, button_9, button_Add, button_Sub,
21              button_Mul, button_Div, button_Equ, button_Del, button_Dot;
22
```

Moving on to the next line, **@Override** indicates that a method declaration is intended to override a method declaration in a supertype. This method is subsequently written with the line 23 to 26,

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
```

The **onCreate** method is called when Activity class is first created. The **savedInstanceState** bundle contains information recorded by the Android framework that the super method will then use to restore state. For instance, any **EditText** with an android:id attribute will automatically save whatever the user has typed into it, and that information will be inside the **savedInstanceState** bundle. If you pass null, this automatic restoration is impossible. As for the **setContentView(R.layout.activity_main)** gives information about our layout resource which is defined in **activity_main.xml** file.

At line 29, **DecimalFormat output = new DecimalFormat("##.####");** is written to declare the format and length of the decimal in the application when keyed in by the user. The button that created in the layout file (**activity_main.xml**) is linked and mapped to the responding variable created. For example, **button_0 = (Button) findViewById(R.id.b0);** where the **findViewById(R.id.b0)** is a class for view widgets that are defined with the name or specific ID declared.

In addition, to set the assigned value to the variable declared, for example, if the button "1" is clicked, a "1" value is saved to the **edittext1** variable, **setOnClickListener** function is used. It is a method that used with buttons to invoke callback function and gives override method inherited from the super class. As a result, the system executes the code written in **onClick(View v)** after the user presses the button. This function can be seen from Line 52 to Line 263.

```
52      button_1.setOnClickListener(new View.OnClickListener() {
53          @Override
54          public void onClick(View v) {
55              edittext1.setText(edittext1.getText() + "1");
56
57          }
58      });
59
60      button_2.setOnClickListener(new View.OnClickListener() {
61          @Override
62          public void onClick(View v) { edittext1.setText(edittext1.getText() + "2"); }
65      });
66
```

On the other hand, the operators that used in the Calculator application such as addition, subtraction, multiplication, and division function operators have a more detailed function and logic**. If else** function is utilized too. For example, the addition operators,

```
if (edittext1.getText().length() != 0) {
                    in1 = Float.parseFloat(edittext1.getText() + "");
                    edittext2.setText("+");
                    Add = true;
                    deci = false;
                    edittext1.setHint(null);
                    edittext1.setText(null);
                }
```

The **in1** is an instance declared to hold the first input value by using the **getText()** function. As for the **edittext2.setText("+");**, it is used to display the addition symbol on the screen of

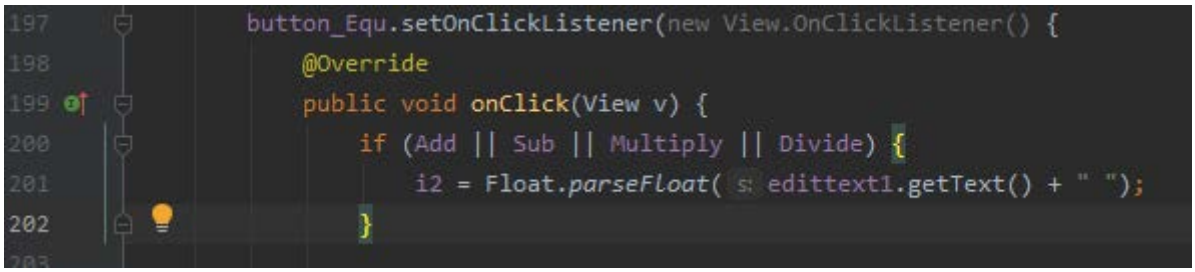
application. Following by the **true and false logical operator,** this is used to assign the "True" value to the "Add" variable when pressed the button. This logic statement is the same format for the operators of addition, subtraction, multiplication, and division.

```
140         button_Sub.setOnClickListener(new View.OnClickListener() {
141             @Override
142             public void onClick(View v) {
143
144                 dot_clicked = false;
145
146                 if (edittext1.getText().length() != 0) {
147                     in1 = Float.parseFloat( s: edittext1.getText() + "");
148                     edittext2.setText("-");
149
150                     Sub = true;
151                     deci = false;
152                     edittext1.setHint(null);
153                     edittext1.setText(null);
154                 }
155             }
156         });
```

```
158         button_Mul.setOnClickListener(new View.OnClickListener() {
159             @Override
160             public void onClick(View v) {
161
162                 dot_clicked = false;
163
164                 if (edittext1.getText().length() != 0) {
165                     in1 = Float.parseFloat( s: edittext1.getText() + "");
166                     edittext2.setText("*");
167
168                     Multiply = true;
169                     deci = false;
170
171                     edittext1.setHint(null);
172                     edittext1.setText(null);
173                 }
174             }
175         });
```

```
177         button_Div.setOnClickListener(new View.OnClickListener() {
178             @Override
179             public void onClick(View v) {
180
181                 dot_clicked = false;
182
183                 if (edittext1.getText().length() != 0) {
184                     in1 = Float.parseFloat( s: edittext1.getText() + "");
185                     edittext2.setText("/");
186
187                     Divide = true;
188                     deci = false;
189
190                     edittext1.setHint(null);
191                     edittext1.setText(null);
192                 }
193             }
194         });
```

To calculate the inputted value of the application, the result will be generated upon pressing the equal "=" button. From Line 197 to Line 234, this button includes **if else statement** due to the operators that pressed by the user will be different from each time.




If there any calculation operators are pressed, the new value inputted will be mapped to the **i2** variable which is the second input value. After that, the calculation will start according to the calculation operators. For example, if addition button is pressed, **edittext2.setText(output.format(in1 + i2));** is used to set and display the new calculated value as text output of the result. The "+" symbol is used for the function of addition of both in1 and in2 value. This logic is applied same for the subtraction, multiplication, and division calculation operator function.

```
205                 if (Add) {
206
207                     edittext2.setText(output.format( number: in1 + i2));
208                     edittext1.setText(" ");
209                     Add = false;
210                 }
211
212                 if (Sub) {
213
214                     edittext2.setText(output.format( number: in1 - i2));
215                     edittext1.setText(" ");
216                     Sub = false;
217                 }
218
219                 if (Multiply) {
220
221                     edittext2.setText(output.format( number: in1 * i2));
222                     edittext1.setText(" ");
223                     Multiply = false;
224                 }
225
226                 if (Divide) {
227
228                     edittext2.setText(output.format( number: in1 / i2));
229                     edittext1.setText(" ");
230                     Divide = false;
231                 }
```

Besides that, there is a delete or clear function button is added to this calculator. It is used to clear the inputted value and set it back to null value or 0.

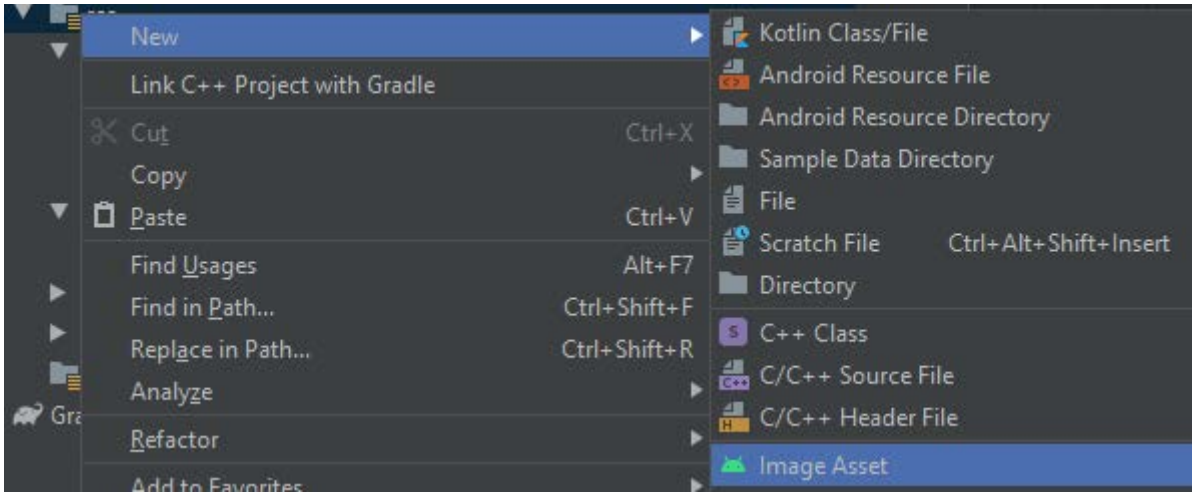
```
236         button_Del.setOnClickListener(new View.OnClickListener() {
237             @Override
238             public void onClick(View v) {
239                 edittext1.setText(null);
240                 edittext1.setHint("Input");
241
242                 edittext2.setText(null);
243                 edittext2.setHint("Result");
244
245                 dot_clicked = false;
246                 in1 = 0.0;
247                 i2 = 0.0;
248             }
249         });
```

Last but not least, a decimal function is also included in this calculator application for a better flexibility. There is a conditional statement for this decimal function such as if the user does not input any value before pressing the decimal button, it will set the text value to “0.”.

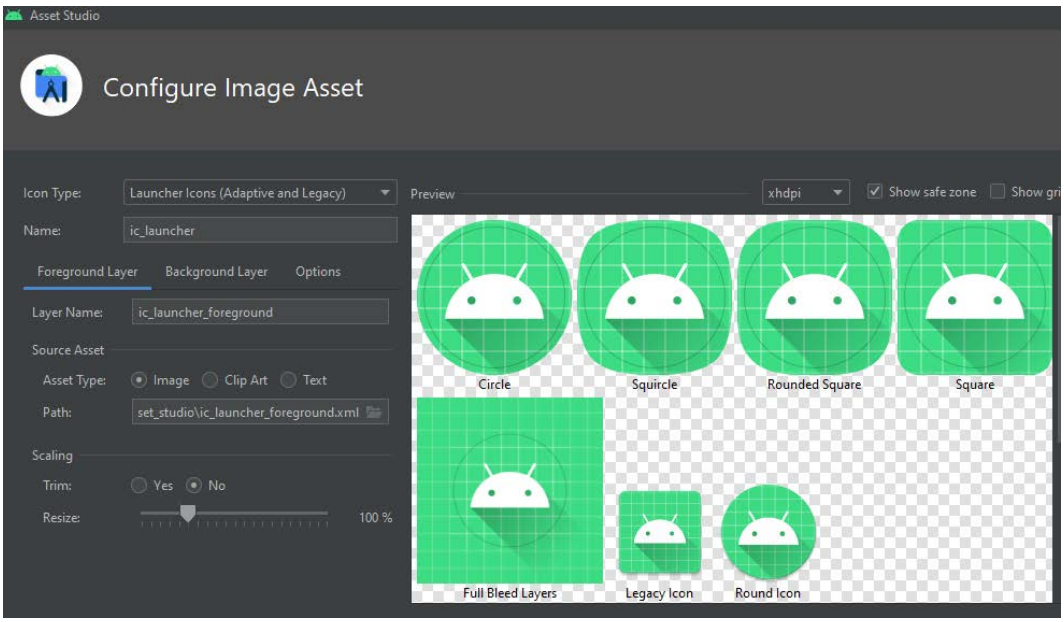
## 4. Change Icon Launcher Design

As a new project created, the icon launcher for the application is set to default icon with the default image and background. It can be changed by creating a new Image Asset. Firstly, right clicking the "res" folder of the project in the app file.

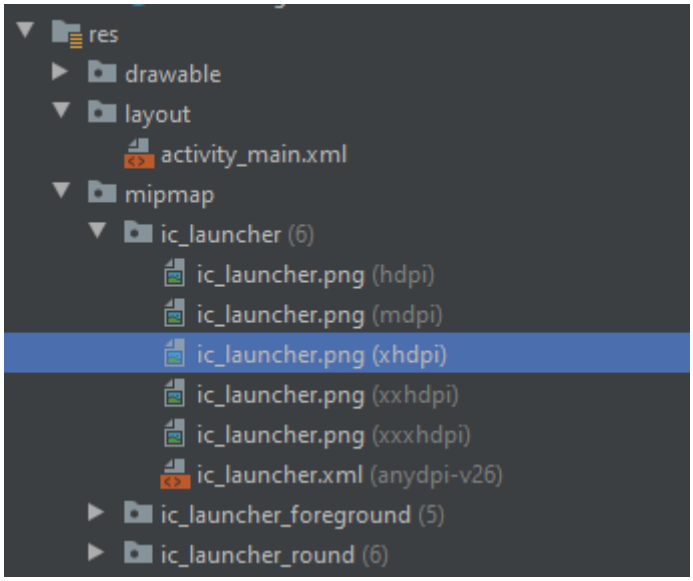

Then, select "New" and select "Image Asset" as shown in the figure.




A new window will be popped up and there is selection to apply own image from the Source Asset tab. From changing the icon logo, size and the background layer. If done, press "Next" and "Finish".



A new icon launcher file will be replaced and created in the /res/mipmap/ic_launcher.



**Reference**


Android studio. (n. d.). Retrieved from https://searchmobilecomputing.techtarget.com/definition/Android-Studio

Emulator. (n. d.). Retrieved from https://whatis.techtarget.com/definition/emulator?_gl=1*psta4z*_ga*MTg0MTQ3NDY0OS4xNjE3MDkwMDU4*_ga_RRBYR9CGB9*MTYxODIzMzQ3OC4yLjEuMTYxODIzNDM1MS4w&_ga=2.122200322.840037868.1618233478-1841474649.1617090058

Heller, M. (2020, Mar 23). *What is Kotlin? The Java alternative explained*. Retrieved from https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html

Leahy, P. (2019, July 3). *What is Java?* Retrieved from https://www.thoughtco.com/what-is-java-2034117

Top 10 features of android studio of developers not to miss. (2018, July 17). Retrieved from https://www.admecindia.co.in/miscellaneous/top-10-features-android-studio-developers-not-miss/

What is Java? Definition, meaning & features of Java platforms. (n. d.). Retrieved from https://www.guru99.com/java-platform.html