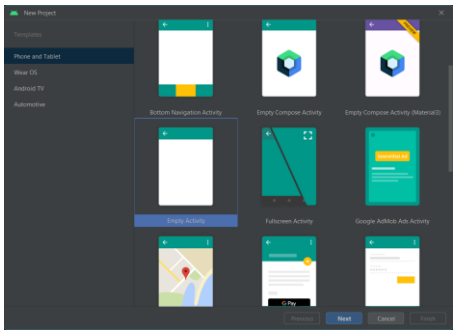
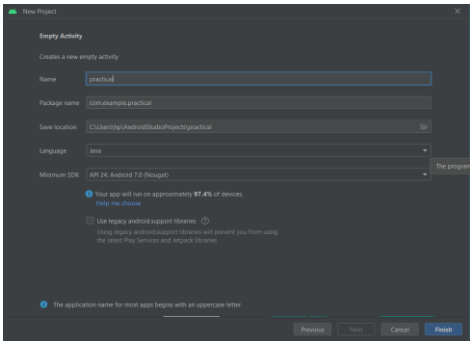
PRACTICAL 1

AIM: Introduction to Android, Introduction to Android Studio IDE, Application Fundamentals: Creating a Project, Android Components, Activities, Services, Content Providers, Broadcast Receivers, Interface overview, Creating Android Virtual device, USB debugging mode, Android Application Overview. Simple “Hello World” program.

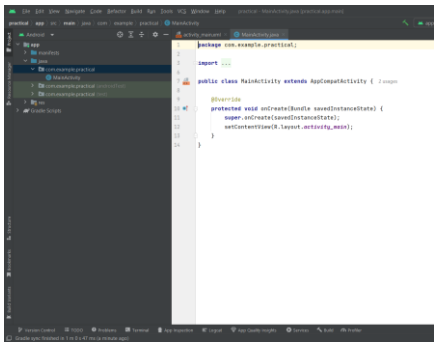
STEP 1: Creating a new project



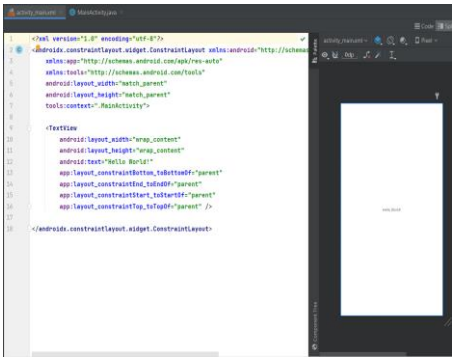
STEP 2: Give the application name



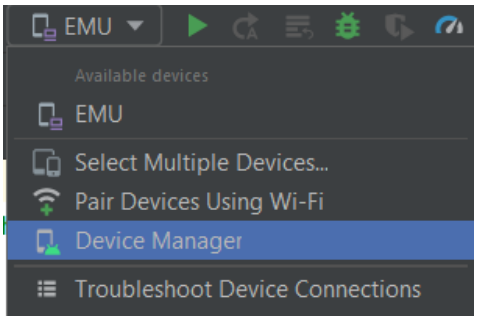
STEP 3: New Project created

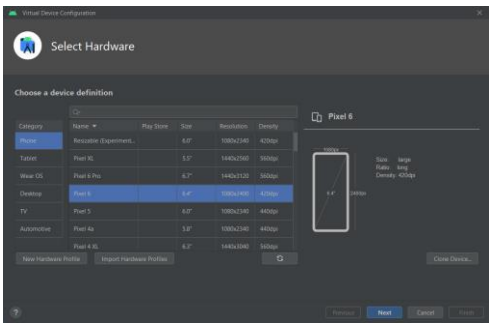


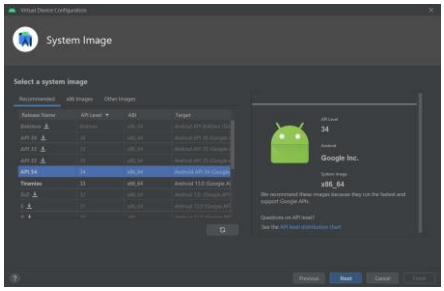
STEP 4: “Hello World”

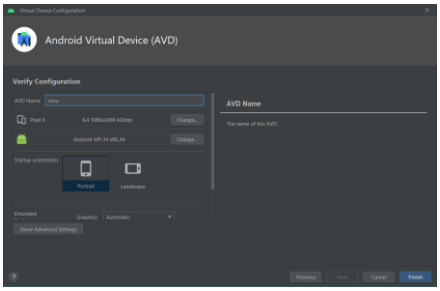


STEP 5: Creating AVD (Android Virtual Machine) as per your name



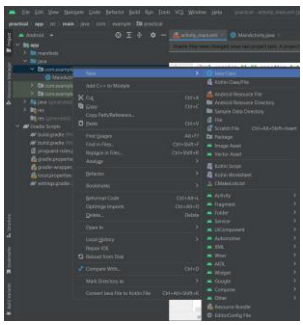


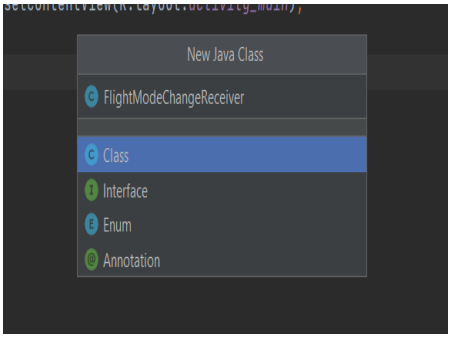


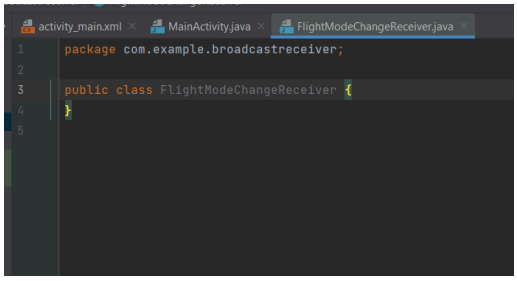


STEP 6: Output will be as per your name on the virtual device

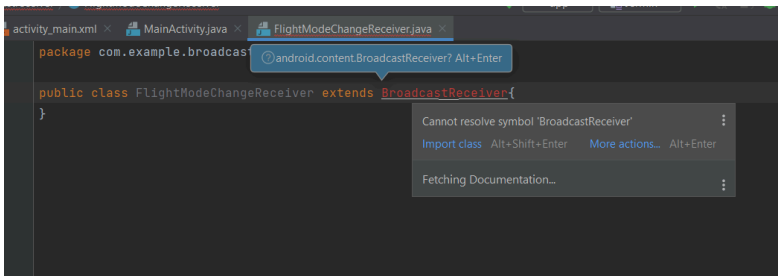
STEP 7: Now we will set “Broadcast Receiver” in Android Studio.Create a Java Class name “FlightModeChangeReceiver”



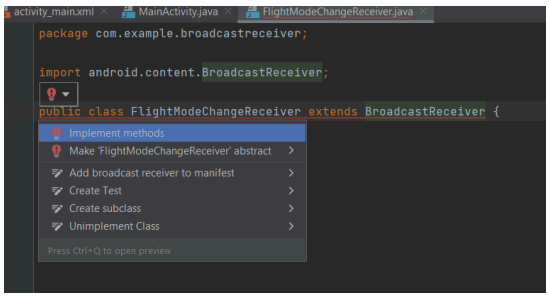




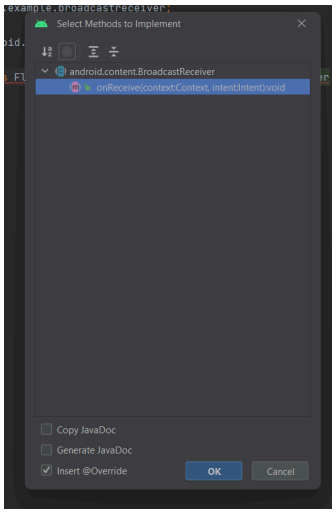
STEP 9: Extend the class, type “BroadcastReceiver” and then import class.



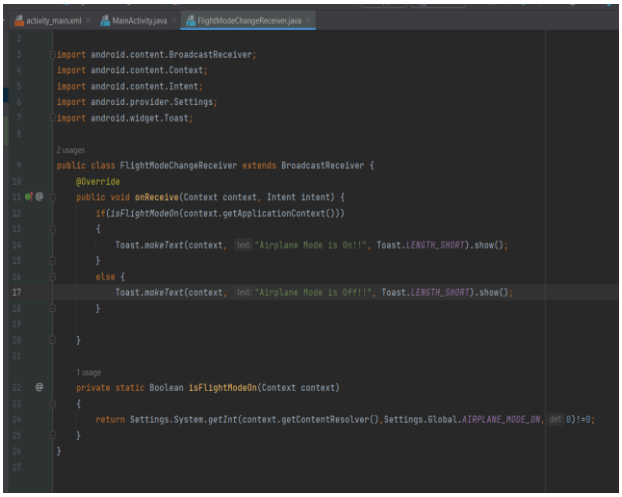
STEP 10: Afterward, go to the red icon, select the first option “Implement methods” and hit enter.



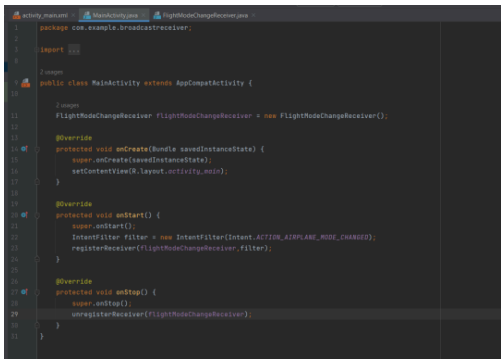
STEP 11: Click OK



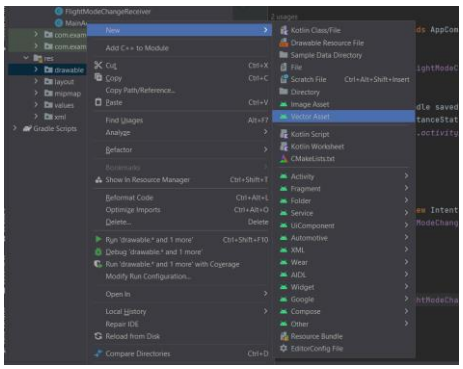
STEP 12: Type the complete code in the “FlightModeChangeReceiver.java” file



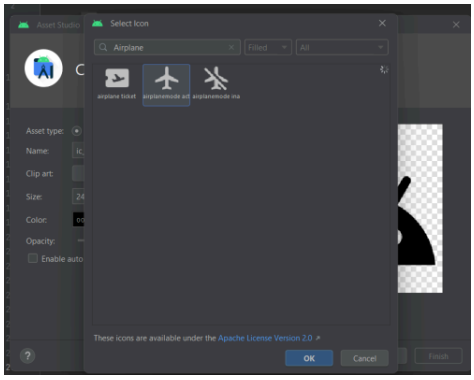
STEP 13: Then go to the “MainActivity.java” file and call the object that you have created in the “FlightModeChangeReceiver.java” file and then use the activity lifecycle to start and stop the airplane mode using onStart() and onStop() methods.

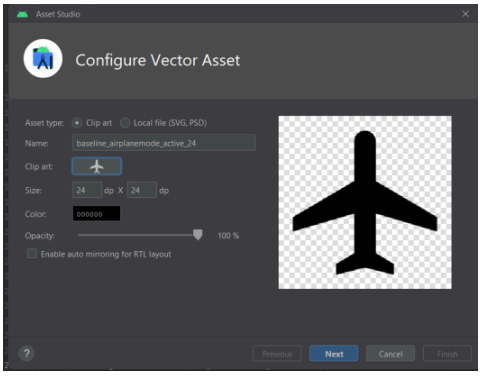


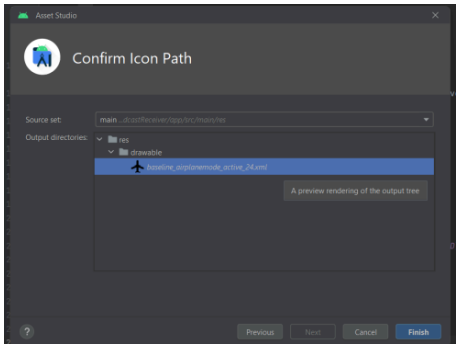
STEP 14: Once you are done with the logic part then you can add airplane mode icon. Go to drawable folder -> select new -> select Vector Asset



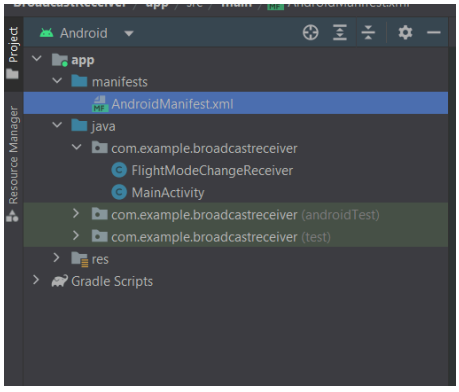
STEP 15: Click on the “clip art” option and type “Airplane”



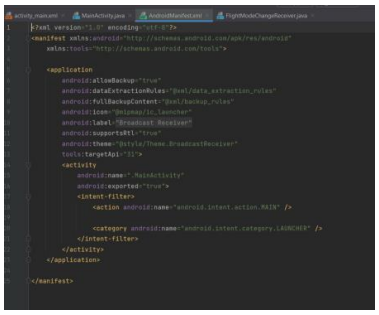




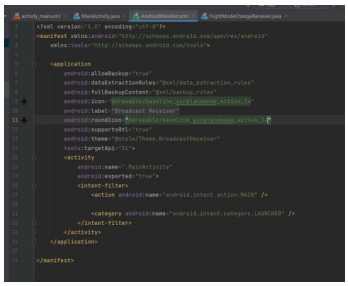
STEP 16: Now you have added the airplane mode icon in the drawable folder then, Go to the “AndroidManifest.xml” file



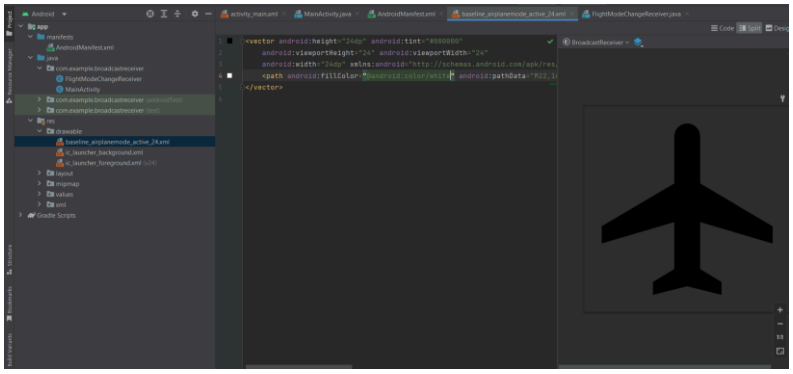
STEP 17: Then inside the “android: icon” property add your icon name which we have selected from the vector asset.



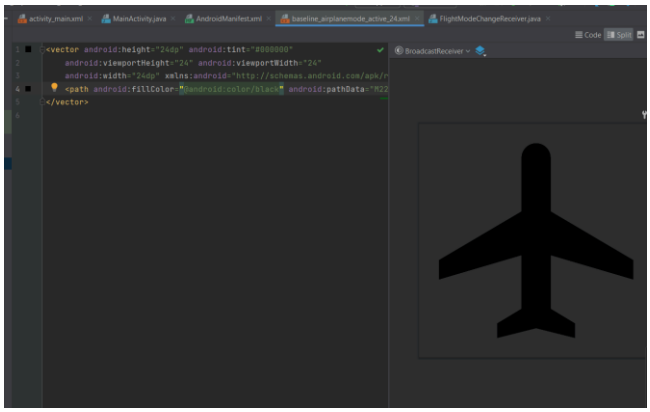
STEP 18: Also you add a new property which is “android:round icon”



STEP 19: To make the icon visible go to the icon which we have added in the drawable folder



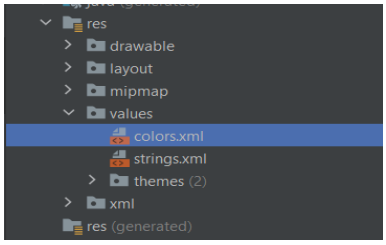
STEP 20: Then change the color to black, now the icon will be visible

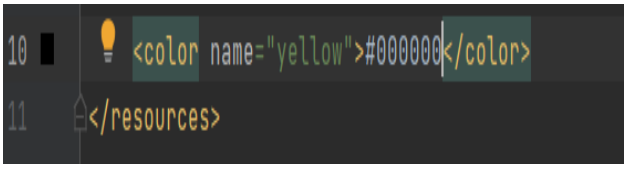


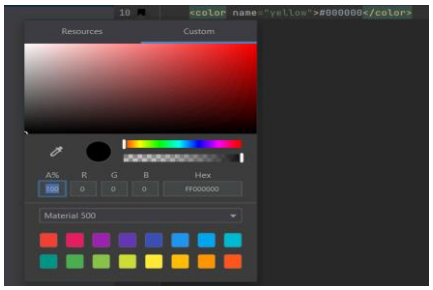
PRACTICAL 2

AIM: Android Resources: (Color, Theme, String, Drawable, Dimension, Image).

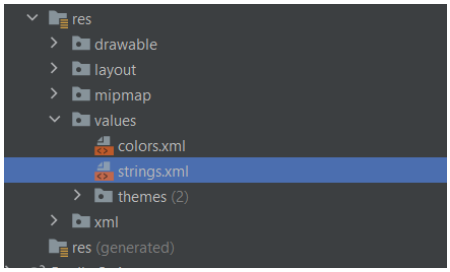
a. COLOR

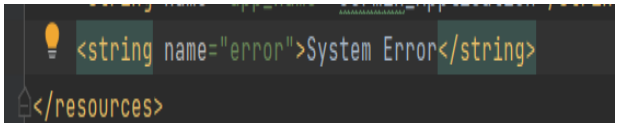




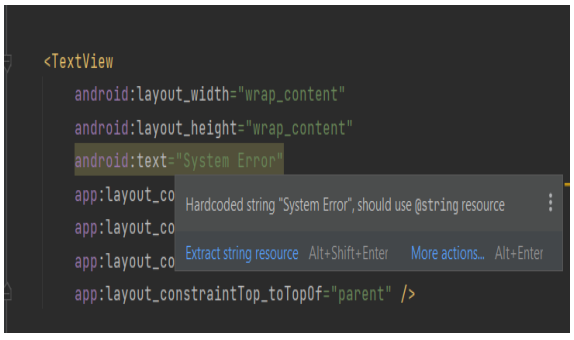


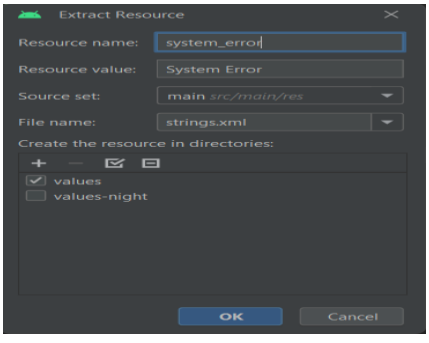
b. STRING

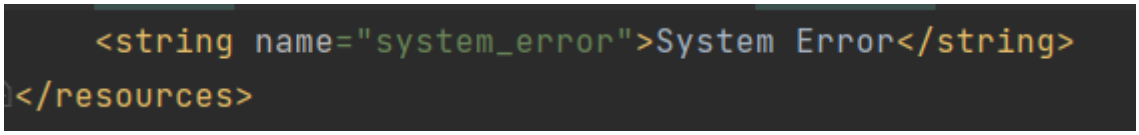




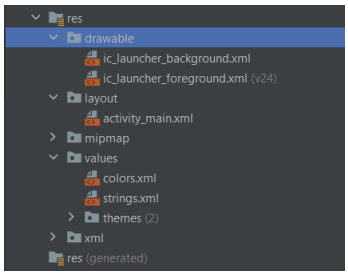
Convert Hardcoded to Softcoded string



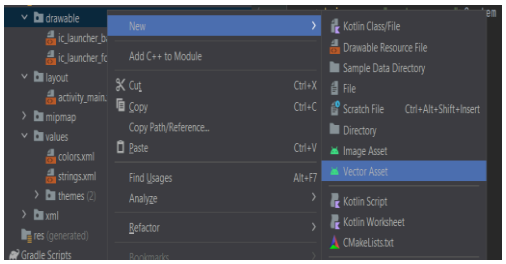


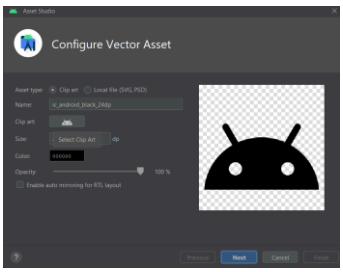


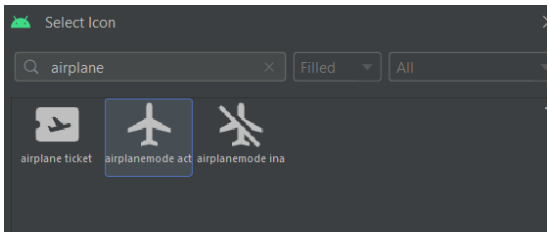
c. DRAWABLE: (3 methods used in drawable folder)

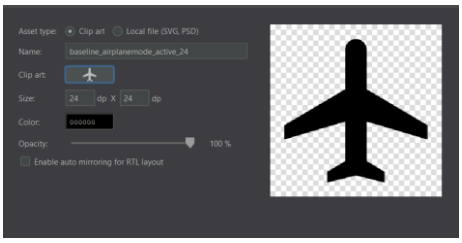


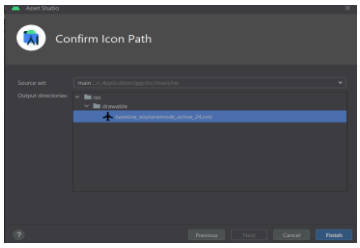
METHOD 1: Vector Asset

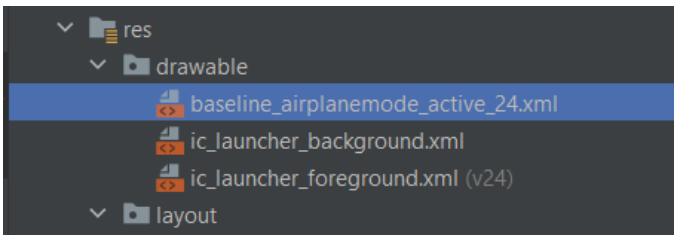






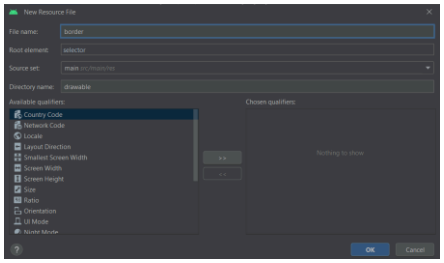


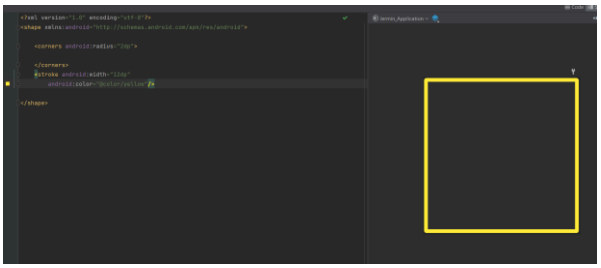


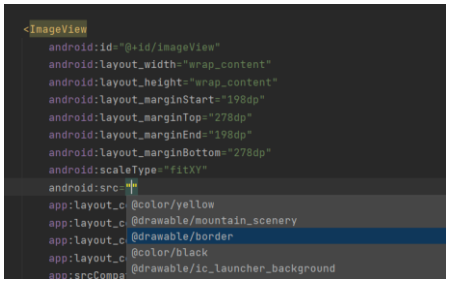


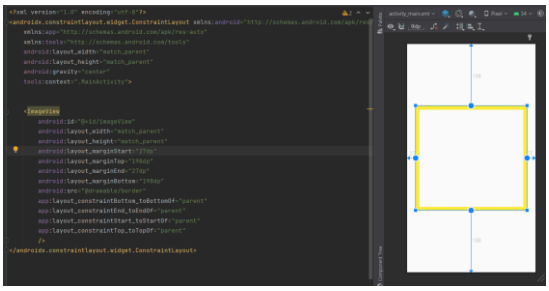
METHOD 2: (Download the image from Google, copy and paste it inside the drawable folder)

METHOD 3: Create your image (eg: Border)

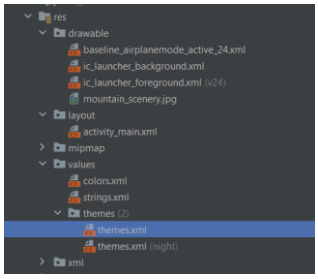




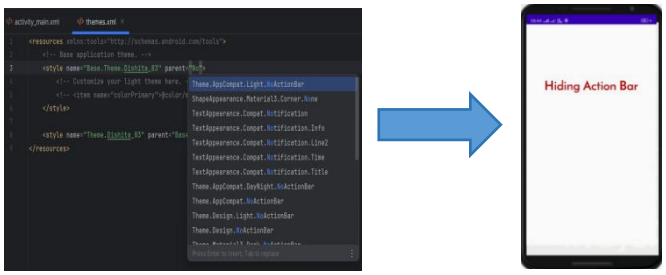




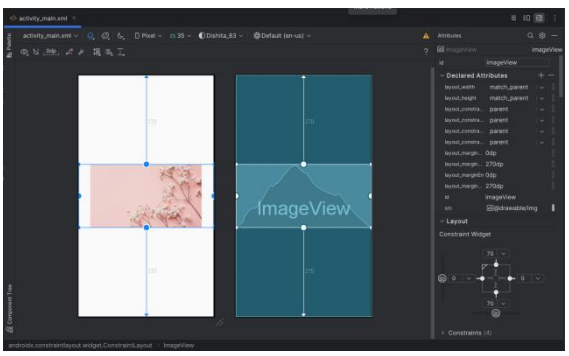
d. THEMES (light mode and dark mode)

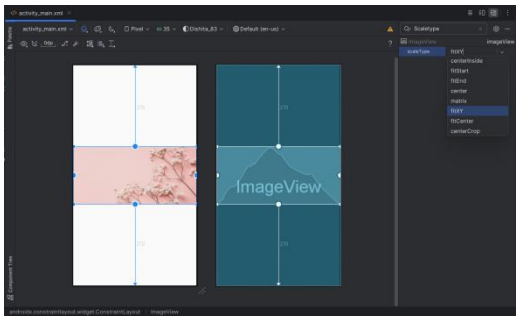


NoActionBar will not display the name of your project during the execution



e. DIMENSION



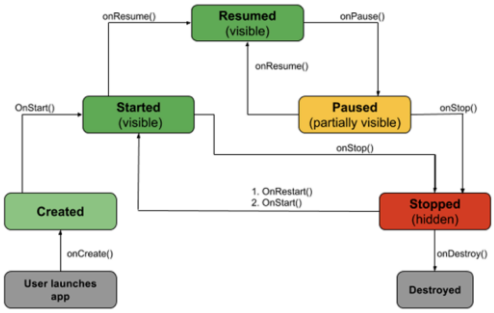


PRACTICAL 3

Programming Activities and fragments

AIM: Activity Life Cycle, Activity methods, Multiple Activities, Life Cycle of fragments and multiple fragments.

Activity Lifecycle:



onCreate(): Called by the OS when the activity is first created. This is where you initialize any UI elements or data objects. You also have the savedInstanceState of the activity that contains its previously saved state, and you can use it to recreate that state.

onStart(): Just before presenting the user with an activity, this method is called. It's always followed by onResume(). In here, you generally should start UI animations, audio-based content or anything else that requires the activity's contents to be on screen.

onResume(): As an activity enters the foreground, this method is called. Here you have a good place to restart animations, update UI elements, restart camera previews, resume audio/video playback, or initialize any components that you release during onPause().

onPause(): This method is called before sliding into the background. Here you should stop any visuals or audio associated with the activity such as UI animations, music playback, or the camera. This method is followed by onResume() if the activity returns to the foreground or by onStop() if it becomes hidden.

onStop(): This method is called right after onPause() when the activity is no longer visible to the user, and it's a good place to save data that you want to commit to the disk. It's followed by either onRestart(), if this activity is returning to the foreground, or onDestroy() if it's being released from memory.

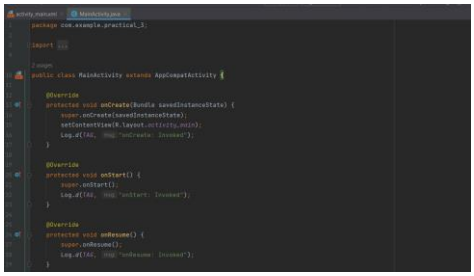
onRestart(): Called after stopping an activity, but just before starting it again. It's always followed by onStart().

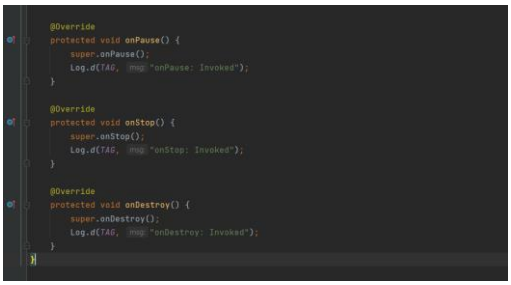
onDestroy(): This is the final callback you'll receive from the OS before the activity is destroyed. You can trigger an activity's destruction by calling finish(), or it can be triggered by the system when the system needs to recoup memory. If your activity includes any background threads or other long-running resources, destruction could lead to a memory leak

if they're not released, so you need to remember to stop these processes here as well.

STEP 1: Create a New Project

STEP 2: Write a code for activity lifecycle in "MainActivity.java"





Output :

```
W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
D/OpenGLRenderer: Swap behavior 0
D/EGL_emulation: eglCreateContext: 0x94d4d8a0: maj 2 min 0 rcv 2
D/EGL_emulation: eglMakeCurrent: 0x94d4d8a0: ver 2 0 (tinfo 0xa8e59f60)
D/EGL_emulation: eglMakeCurrent: 0x94d4d8a0: ver 2 0 (tinfo 0xa8e59f60)
D/Main Activity: onPause
D/EGL_emulation: eglMakeCurrent: 0x94d4d8a0: ver 2 0 (tinfo 0xa8e59f60)
D/Main Activity: onStop
```

```
D/Main Activity: onStop
D/Main Activity: onRestart
D/EGL_emulation: eglMakeCurrent: 0x94d4d8a0: ver 2 0 (tinfo 0xa8e59f60)
D/Main Activity: onStart
D/Main Activity: onResume
```

PRACTICAL 4

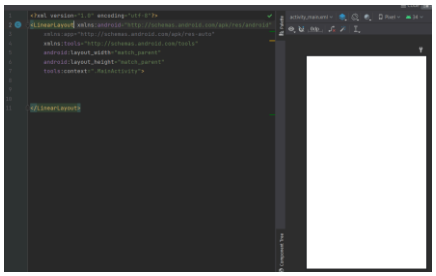
Programs related to different Layouts

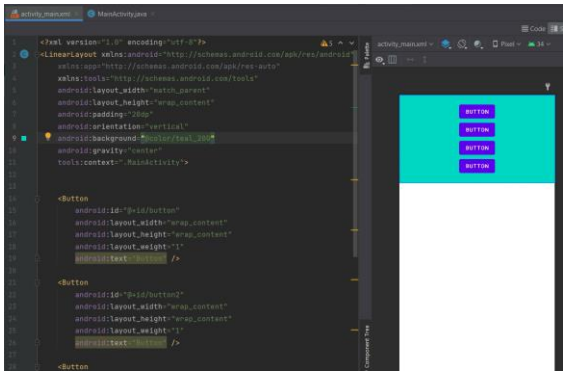
AIM: Coordinate, Linear, Relative, Table, Absolute, Frame, List View, Grid View

1. Linear Layout

STEP 1: Create a New Project

STEP 2: Go to “activity_main.xml” and design a UI using “LinearLayout”



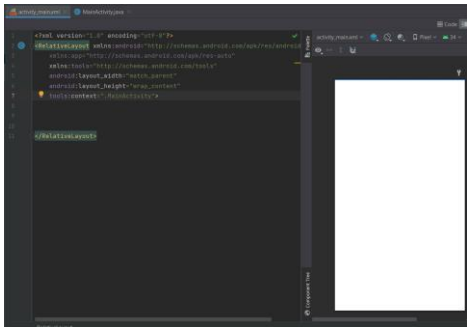


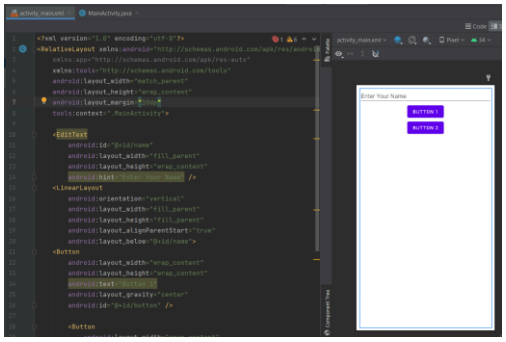
```
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />
</LinearLayout>
```

2. Relative Layout

STEP 1: Create a New Project

STEP 2: Go to “activity_main.xml” and design a UI using “RelativeLayout”



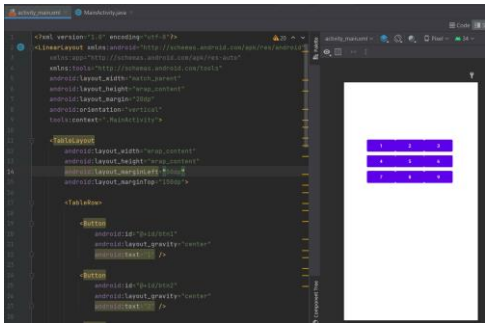


```
        android:layout_below="@+id/name">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1"
        android:layout_gravity="center"
        android:id="@+id/button" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 2"
            android:layout_gravity="center"
            android:id="@+id/button2" />
    </LinearLayout>
</RelativeLayout>
```

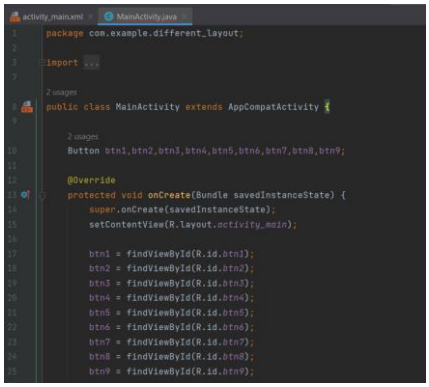
3. Table Layout

STEP 1: Create a New Project

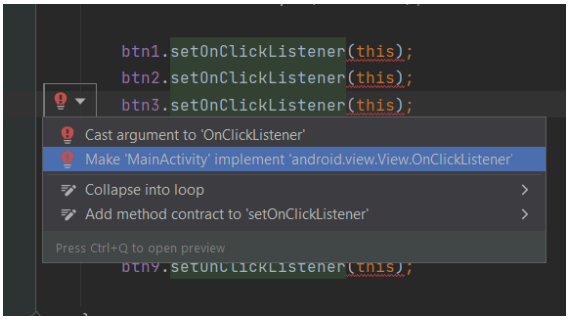
STEP 2: Go to “activity_main.xml” and design a UI using “TableLayout”



```
    <TableRow>

        <Button
            android:id="@+id/btn7"
            android:layout_gravity="center"
            android:text="7" />

        <Button
            android:id="@+id/btn8"
            android:layout_gravity="center"
            android:text="8" />

        <Button
            android:id="@+id/btn9"
            android:layout_gravity="center"
            android:text="9" />
    </TableRow>
</TableLayout>

</LinearLayout>
```



```
package com.example.different_layout;

import ...

2 usages
public class MainActivity extends AppCompatActivity {

    2 usages
    Button btn1,btn2,btn3,btn4,btn5,btn6,btn7,btn8,btn9;

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

        btn1 = findViewById(R.id.btn1);
        btn2 = findViewById(R.id.btn2);
        btn3 = findViewById(R.id.btn3);
        btn4 = findViewById(R.id.btn4);
        btn5 = findViewById(R.id.btn5);
        btn6 = findViewById(R.id.btn6);
        btn7 = findViewById(R.id.btn7);
        btn8 = findViewById(R.id.btn8);
        btn9 = findViewById(R.id.btn9);
```



```
btn1.setOnClickListener(this);
btn2.setOnClickListener(this);
btn3.setOnClickListener(this);
```

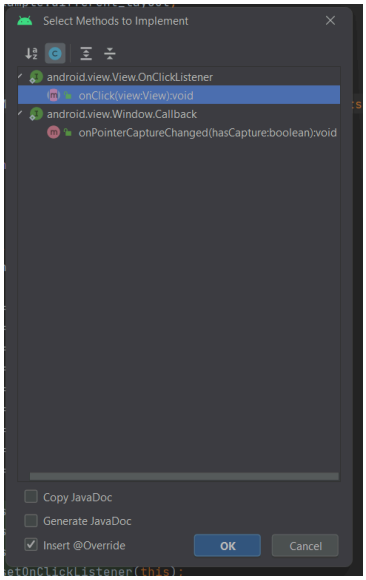
Cast argument to 'OnClickListener'

Make 'MainActivity' implement 'android.view.View.OnClickListener'

Collapse into loop

Add method contract to 'setOnClickListener'

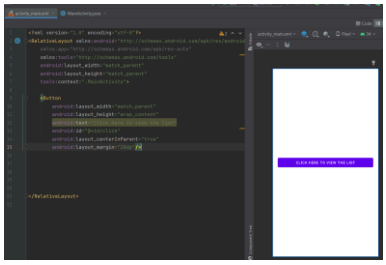
Press Ctrl+Q to open preview



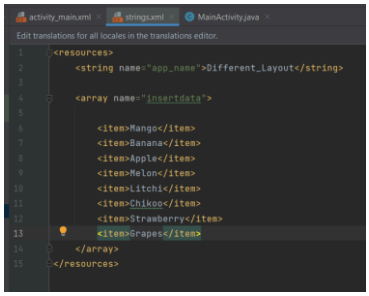
4. List View Layout

STEP 1: Create a New Project

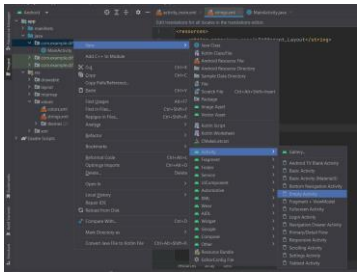
STEP 2: Go to “activity_main.xml” and design a UI using “ListViewLayout”

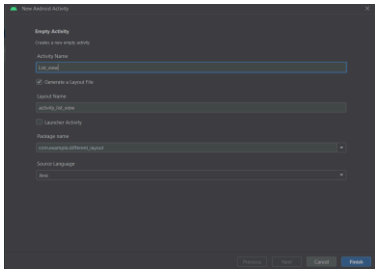


STEP 3: Go to the res folder, click on the values folder, and select “strings”

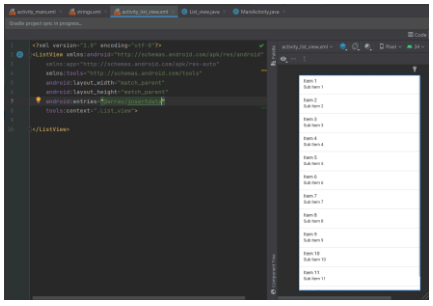


STEP 4: Go to the java folder and click and select a new empty activity

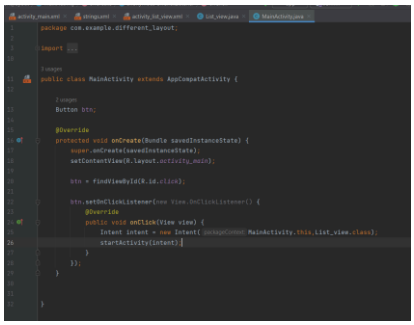




STEP 5: Go to “activity_list_view.xml” and insert the data which we have created inside the strings



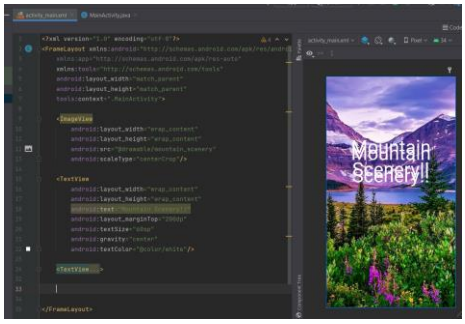
STEP 6: Go to “MainActivity.java” and write the logic



5. Frame Layout

STEP 1: Create a New Project

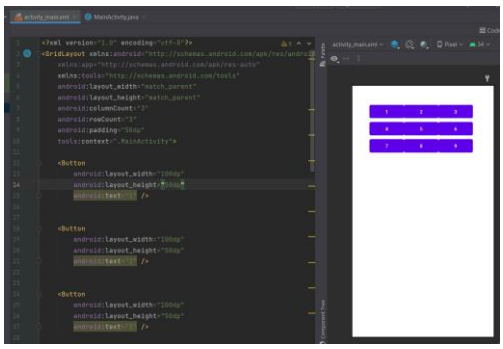
STEP 2: Go to “activity_main.xml” and design a UI using “FrameLayout”



6. Grid View Layout

STEP 1: Create a New Project

STEP 2: Go to “activity_main.xml” and design a UI using “GridLayout”



PRACTICAL 5

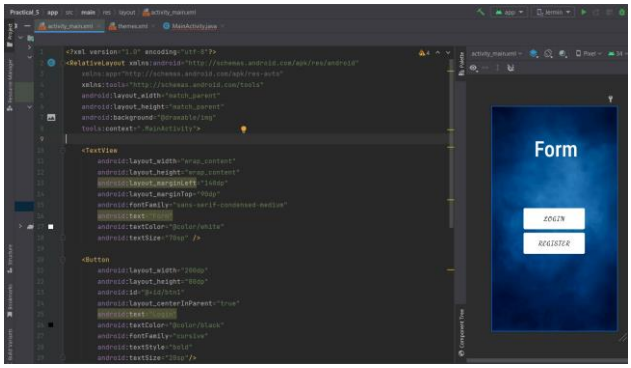
Programming UI elements

AIM: Design App With UI:

STEP 1: Create a Login and Register Page

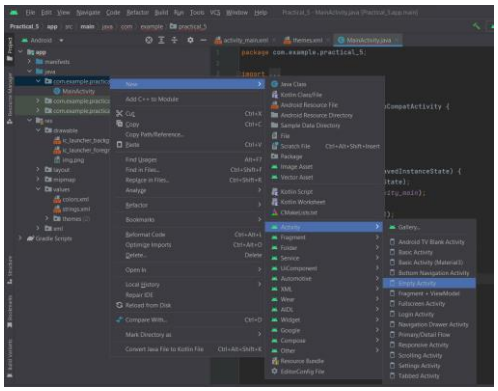
STEP 2: Create a New Project

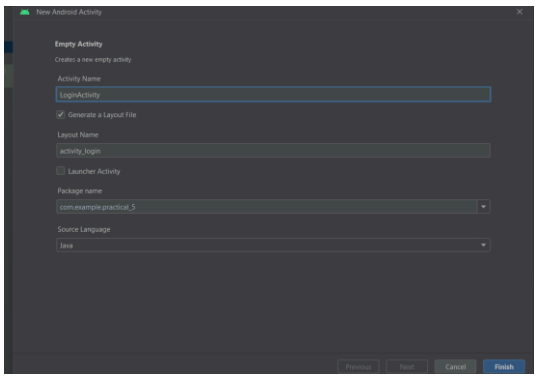
STEP 3: Design UI and write the code in “activity_main.xml”

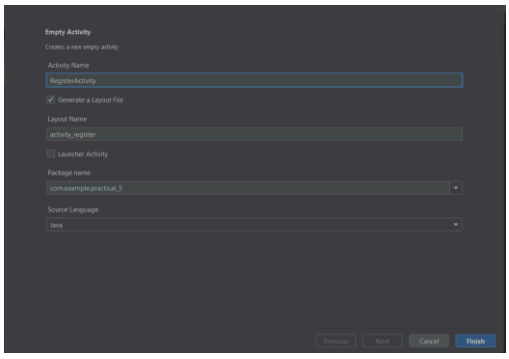


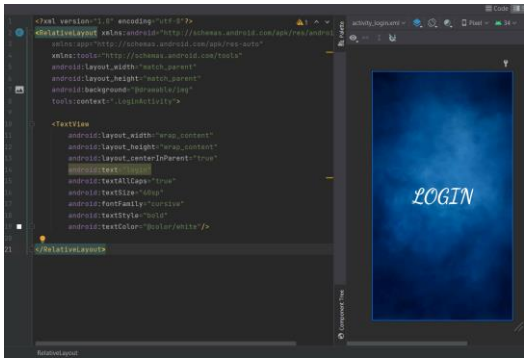

```
    <Button
        android:layout_width="200dp"
        android:layout_height="80dp"
        android:id="@+id/btn2"
        android:layout_centerInParent="true"
        android:text="Register"
        android:textStyle="bold"
        android:layout_below="@id/btn1"
        android:textColor="@color/black"
        android:fontFamily="cursive"
        android:textSize="20sp"/>

</RelativeLayout>
```

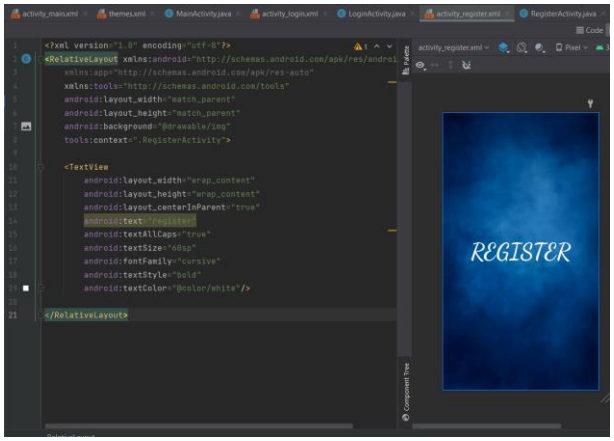
STEP 4: Create a new empty activity for the login and register page



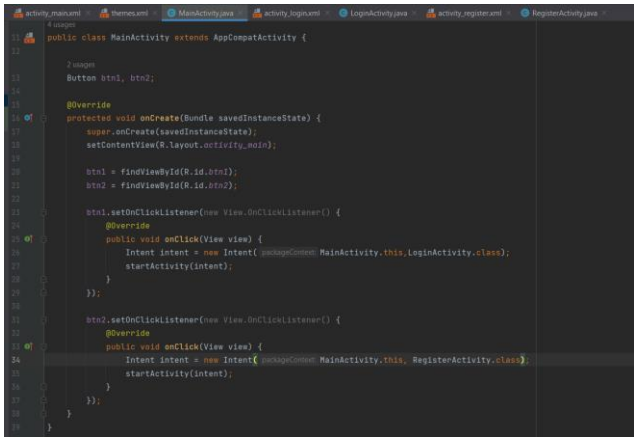








STEP 5: Write logic for the login and register page in“MainActivity.java”



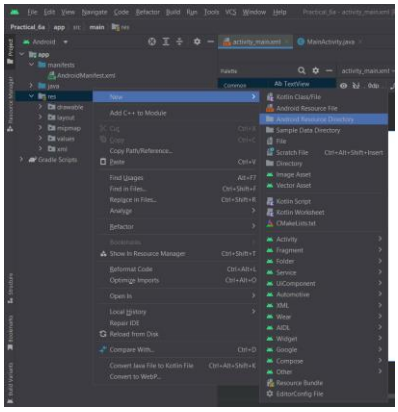
PRACTICAL 6

Programming menus, dialog, dialog fragments

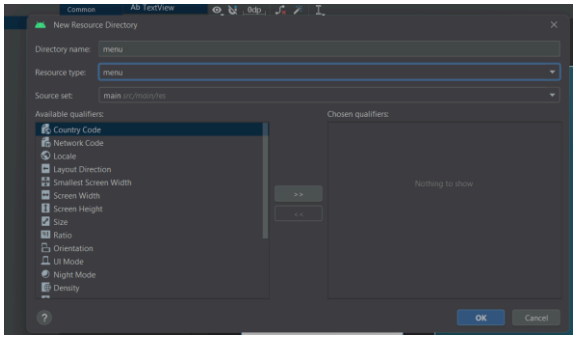
1. MENU OPTIONS

STEP 1: Create a New Project

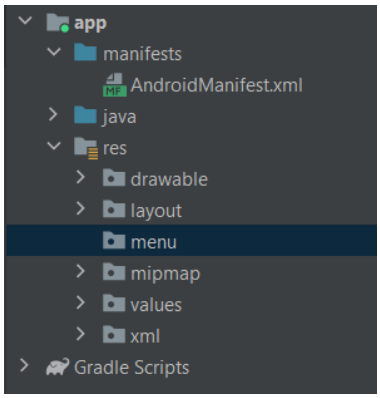
STEP 2: Go to the “res” folder, right-click, select new and then “Android Resource Directory”



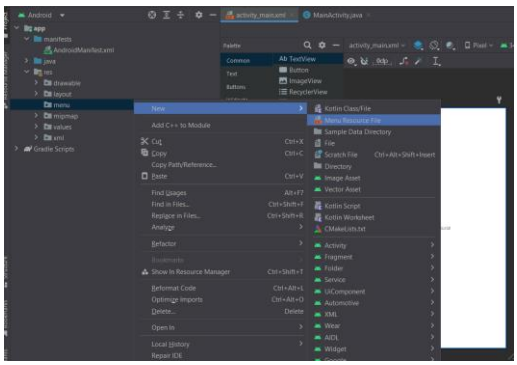
STEP 3: Create a New Project



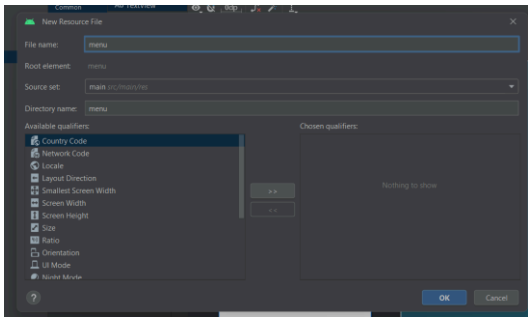
STEP 4: Go to the “res” folder, right-click, select new and then “Android Resource Directory”



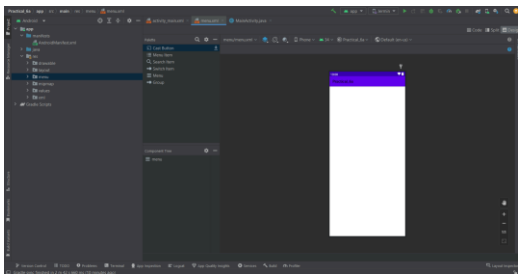
STEP 5: Right-click on the menu folder, select new and then click on “Menu Resource File”



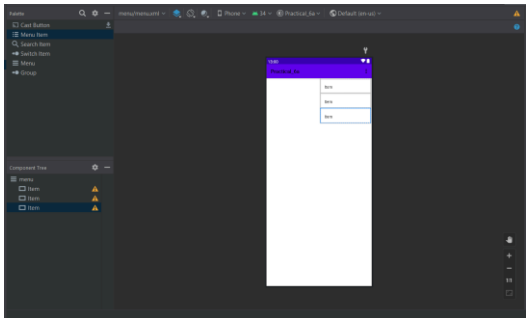
STEP 6: Give the file name as “menu” and click ok



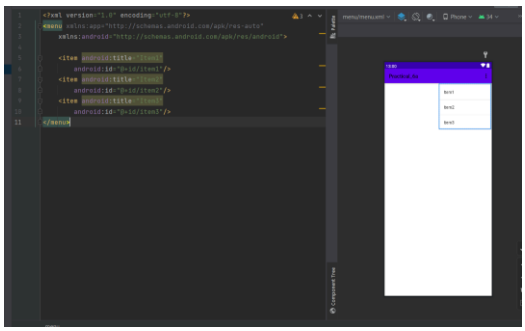
STEP 7: Then, the menu activity is ready



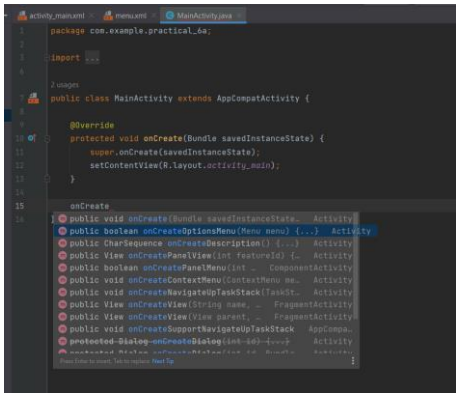
STEP 8: Drag-drop 3 menu items on the activity



STEP 9: Go to split mode and change the item name to item1, item2, and item3 also give the id to all the items

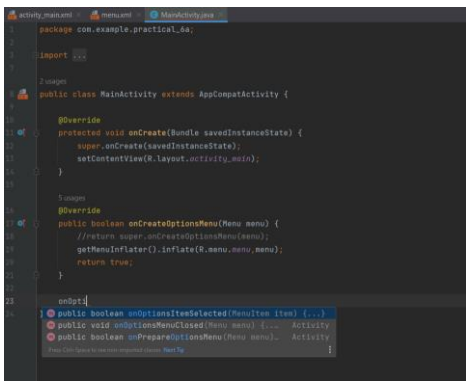


STEP 10: Go to “MainActivity.java” and write the logic

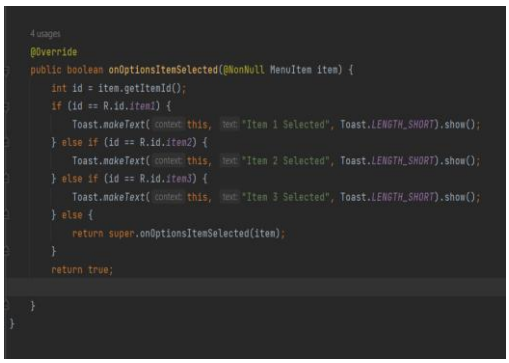


```
package com.example.practical_6a;

import ...

2 usages
public class MainActivity extends AppCompatActivity {

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

    onCreate
    public void onCreate(Bundle savedInstanceState,  Activity
    public boolean onCreateOptionsMenu(Menu menu) {...}  Activity
    public CharSequence onCreateDescription() {...}  Activity
    public View onCreatePanelView(int featureId) {...}  Activity
    public boolean onCreatePanelMenu(int ...  ComponentActivity
    public void onCreateContextMenu(ContextMenu me...  Activity
    public void onCreateNavigateUpTaskStack(TaskSt...  Activity
    public View onCreateView(String name, ...  FragmentActivity
    public View onCreateView(View parent, ...  FragmentActivity
    public void onCreateSupportNavigateUpTaskStack  AppCompa...
    protected Dialog onCreateDialog(int id) {...}  Activity
```

```
package com.example.practical_6a;

import ...

2 usages
public class MainActivity extends AppCompatActivity {

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

    5 usages
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //return super.onCreateOptionsMenu(menu);
        getMenuInflater().inflate(R.menu.menu,menu);
        return true;
    }

}
```

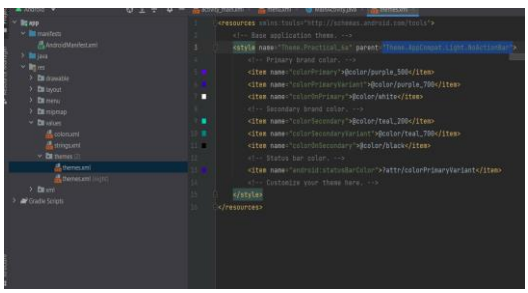


```
package com.example.practical_6a;

import ...

2 usages
public class MainActivity extends AppCompatActivity {

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

    5 usages
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //return super.onCreateOptionsMenu(menu);
        getMenuInflater().inflate(R.menu.menu,menu);
        return true;
    }

    onOpti
    public boolean onOptionsItemSelected(MenuItem item) {...}
    public void onOptionsMenuClosed(Menu menu) {...  Activity
    public boolean onPrepareOptionsMenu(Menu menu)...  Activity
```

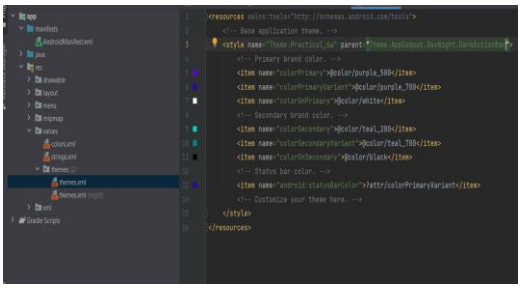


```
4 usages
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.item1) {
        Toast.makeText( context: this,  text: "Item 1 Selected", Toast.LENGTH_SHORT).show();
    } else if (id == R.id.item2) {
        Toast.makeText( context: this,  text: "Item 2 Selected", Toast.LENGTH_SHORT).show();
    } else if (id == R.id.item3) {
        Toast.makeText( context: this,  text: "Item 3 Selected", Toast.LENGTH_SHORT).show();
    } else {
        return super.onOptionsItemSelected(item);
    }
    return true;

}
}
```

STEP 11: Once the logic part is done, go to the “themes” folder and check whether the theme is given as light mode or dark mode if not then make it because if the theme is given as “NoActionBar” then you won’t be able to see the items.



```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.Practical_6a" parent="Theme.MaterialComponents.Light.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

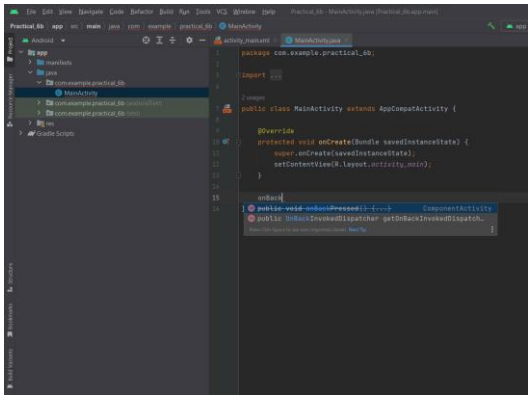


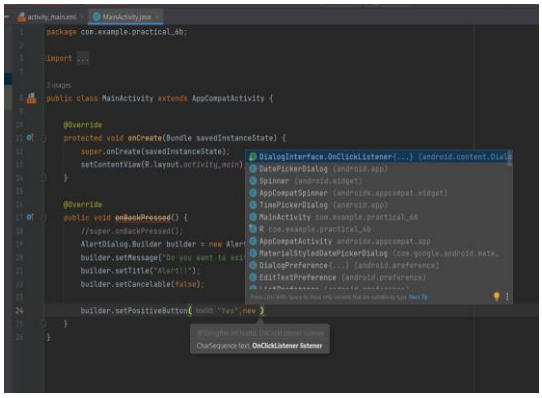
```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.Practical_6a" parent="Theme.AppCompat.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

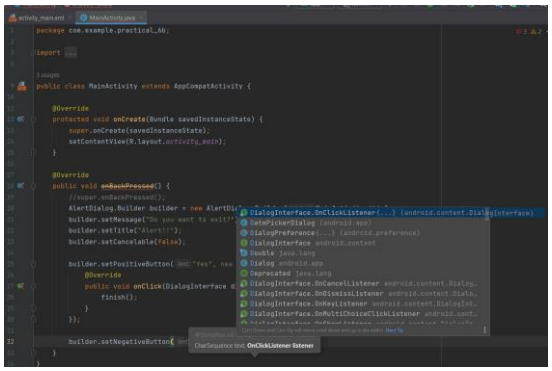
2. ALERT DIALOG BOX

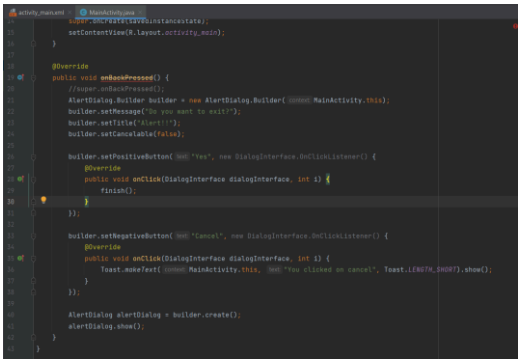
STEP 1: Create a New Project

STEP 2: Go to “MainActivity.java” and create an AlertDialog Box









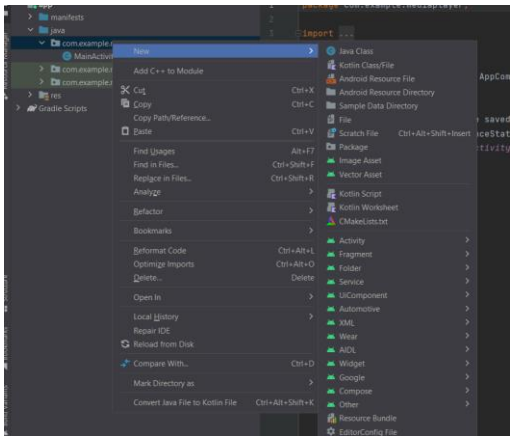
PRACTICAL 7

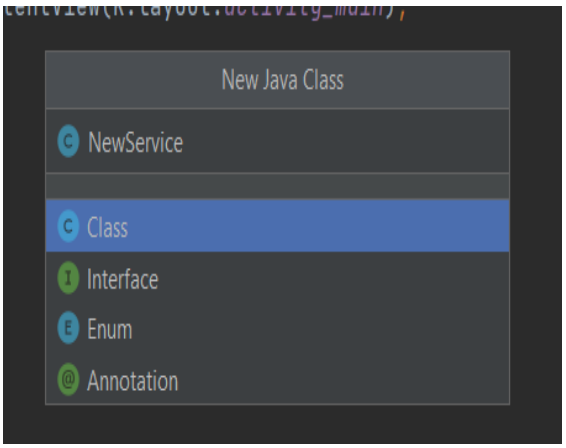
Programs on Services, notifications, and broadcast receivers

PART 1: SERVICES

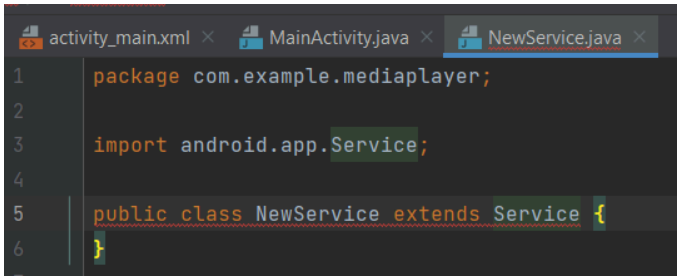
STEP 1: Create a New Project

STEP 2: Create a New Java class name “NewService”

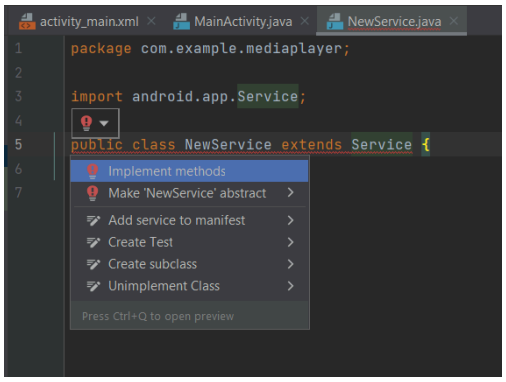




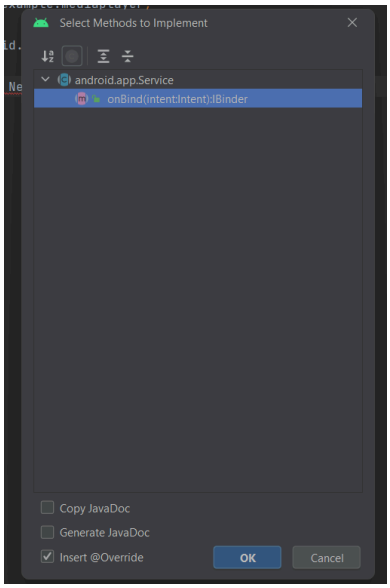
STEP 3: Type extends “Service”

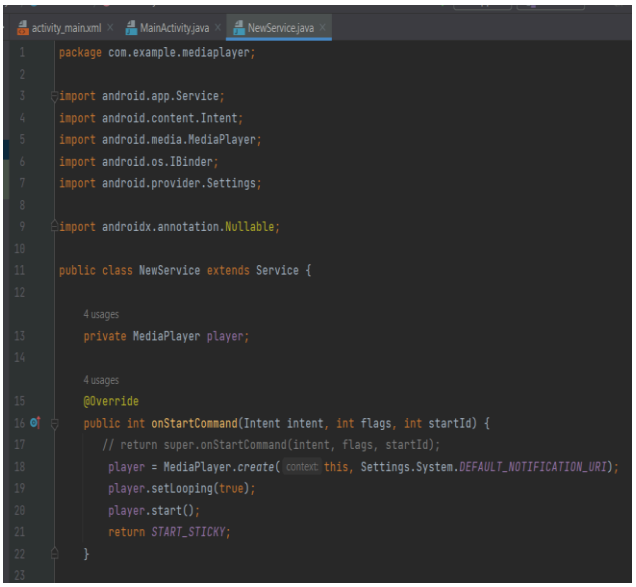


STEP 4: Click on the red icon select the first option and hit enter



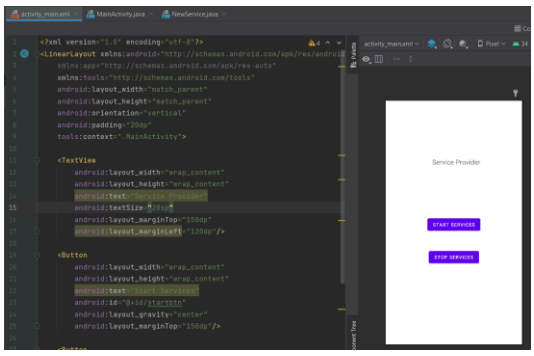
STEP 5: Click Ok and then start writing the code in the “NewService.java” file

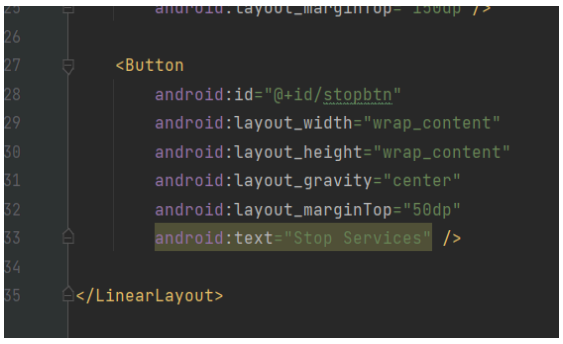




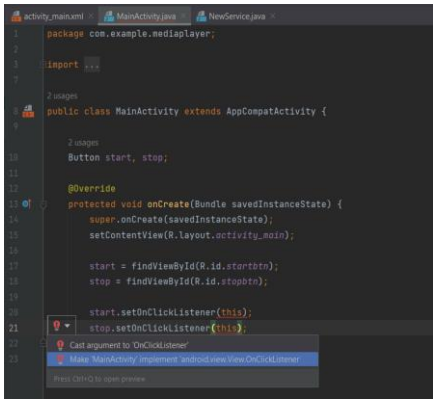
STEP 6: We are using activity lifecycle here to start and stop the service

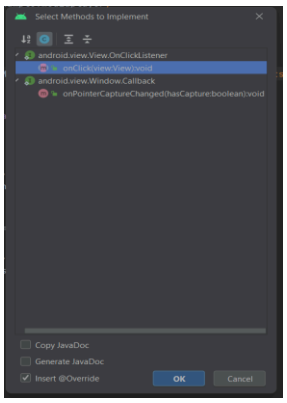
STEP 7: Then we will design UI in the “activity_main.xml” file

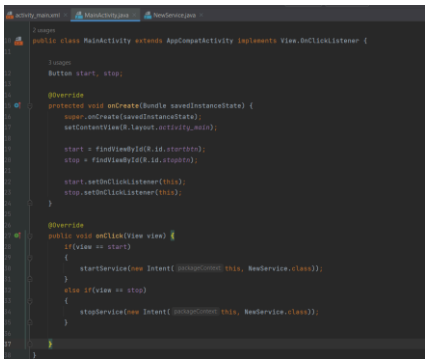




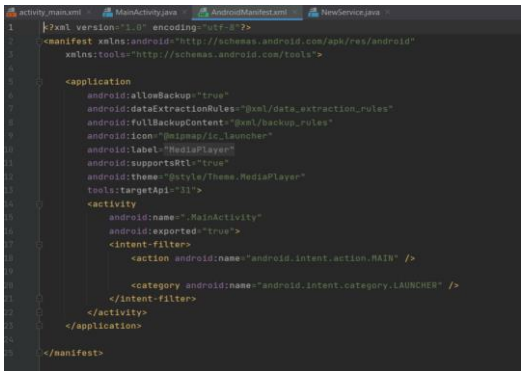
STEP 8: Once we are done with the UI part then we will write the logic behind the button which will be setOnClickListener .Then we will use this pointer for both buttons select the second option and hit enter

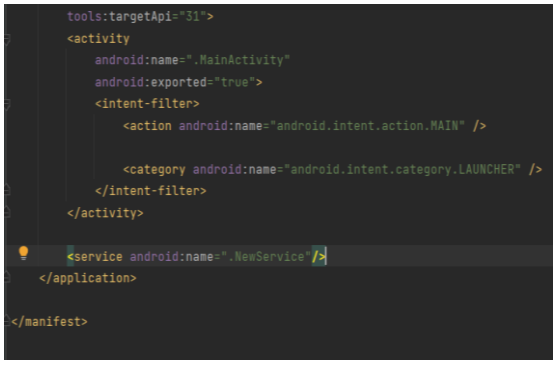






STEP 9: After that, we will go to the “AndroidManifest.xml” file and inside the activity, we will create a service and add the “NewService” file name

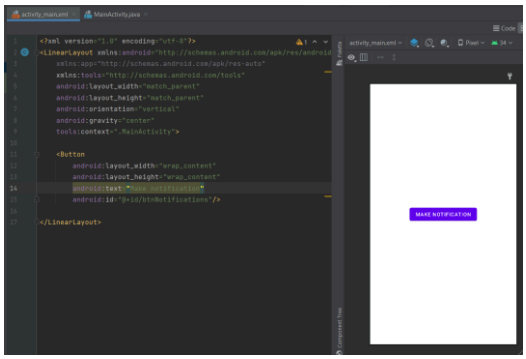




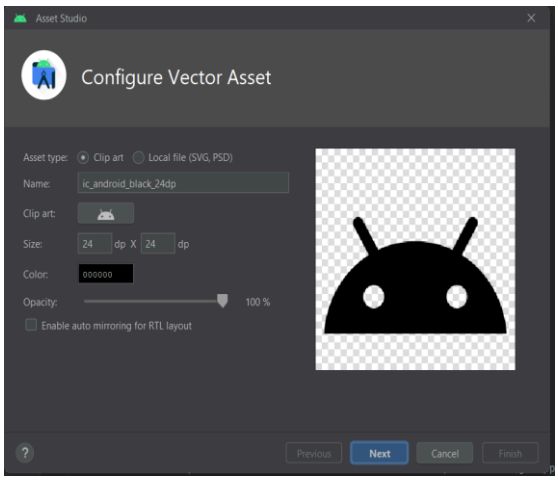
PART 2: NOTIFICATION

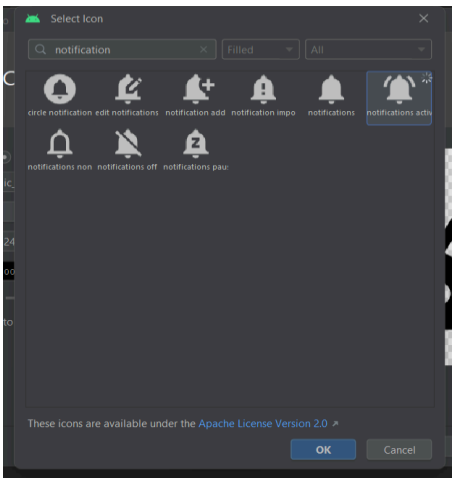
STEP 1: Create a New Project

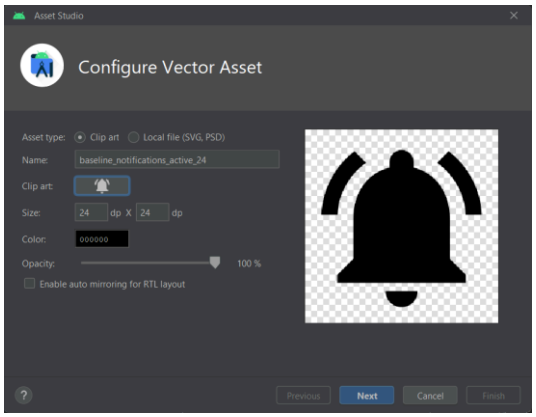
STEP 2: Go to the “activity_main.xml” file and design the UI

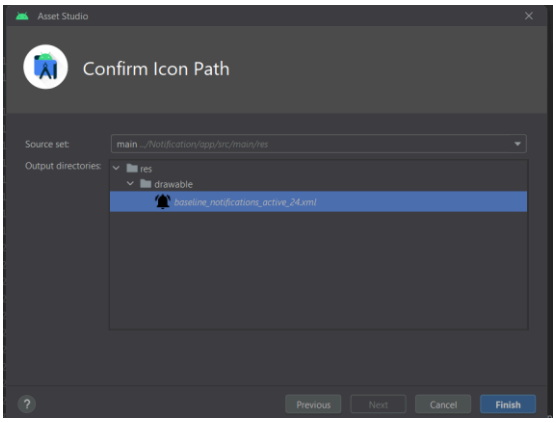


STEP 3: Go to the drawable folder click and select vector asset and select the notification icon

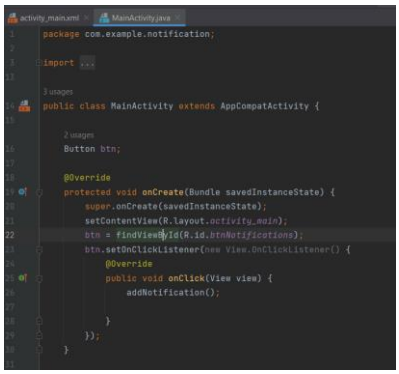




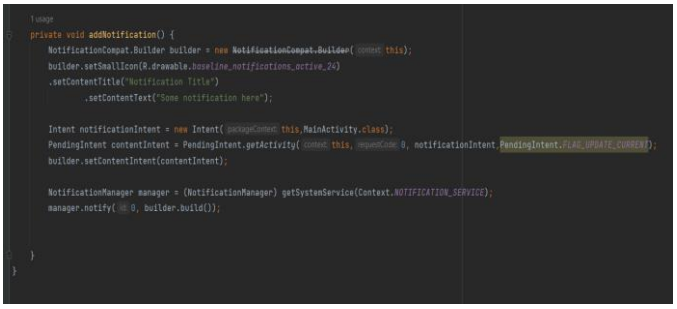




STEP 4: Go to the “MainActivity.java” file and write the logic



```
package com.example.notification;

import ...

public class MainActivity extends AppCompatActivity {

    Button btn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = findViewById(R.id.btnNotifications);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                addNotification();
            }
        });
    }
```

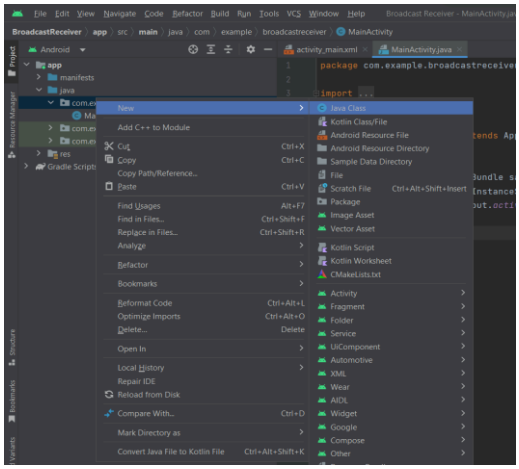


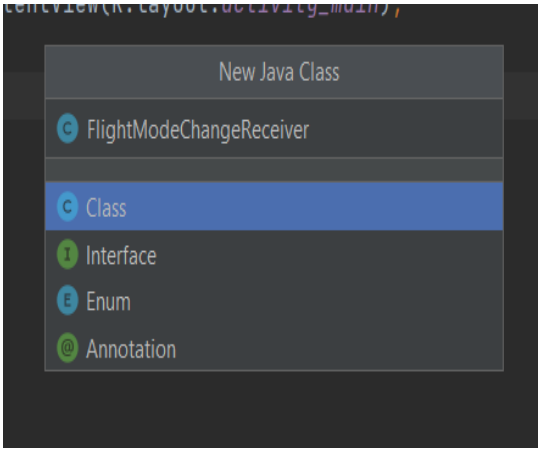
```
private void addNotification() {
    NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this);
    builder.setSmallIcon(R.drawable.baseline_notifications_active_24)
    .setContentTitle("Notification Title")
            .setContentText("Some notification here");

    Intent notificationIntent = new Intent( packageContext: this,MainActivity.class);
    PendingIntent contentIntent = PendingIntent.getActivity( context: this, requestCode: 0, notificationIntent,PendingIntent.FLAG_UPDATE_CURRENT);
    builder.setContentIntent(contentIntent);

    NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    manager.notify( id: 0, builder.build());

}
```

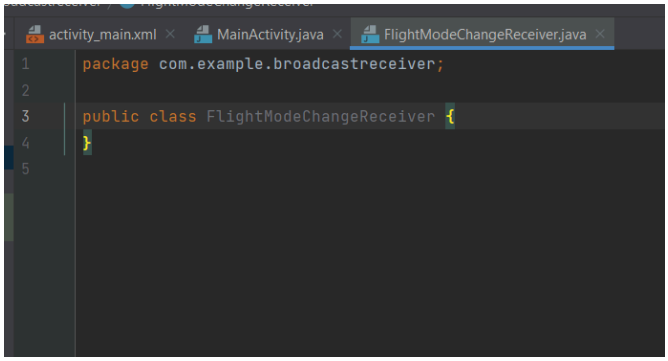
PART 3: BROADCAST RECEIVER

STEP 1: Create a New Project

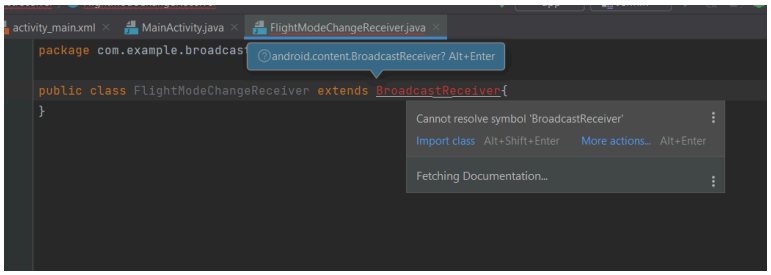
STEP 2: Now we will set “Broadcast Receiver” in Android Studio Create a Java Class name “FlightModeChangeReceiver”



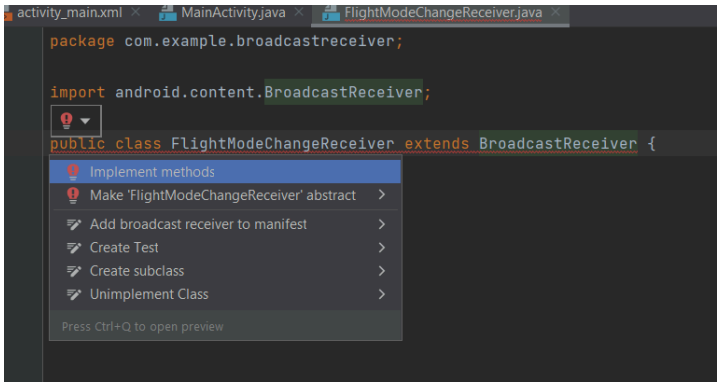




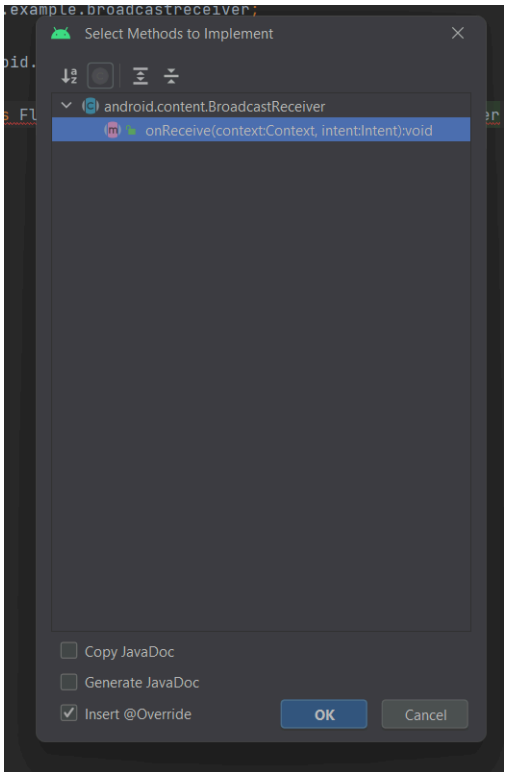
STEP 3: Extend the class, type “BroadcastReceiver” and then import class.



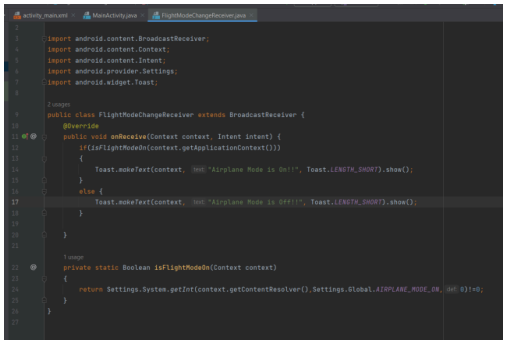
STEP 4: Afterward, go to the red icon, select the first option “Implement methods” and hit enter.



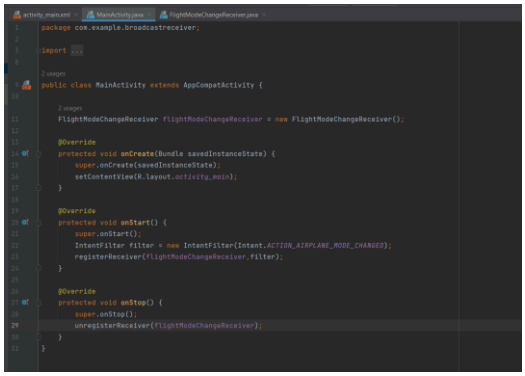
STEP 5: Click OK



STEP 6: Type the complete code in the “FlightModeChangeReceiver.java” file

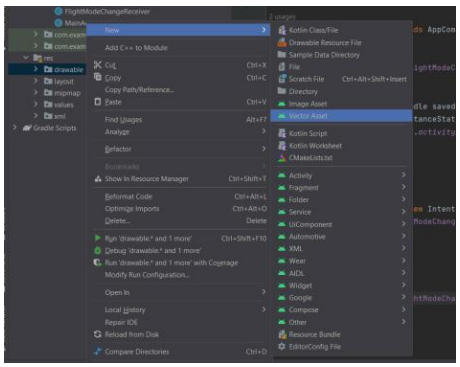


STEP 7: Then go to the “MainActivity.java” file and call the object that you have created in the “FlightModeChangeReceiver.java” file and then use the activity lifecycle to start and stop the airplane mode using onStart() and onStop() methods.

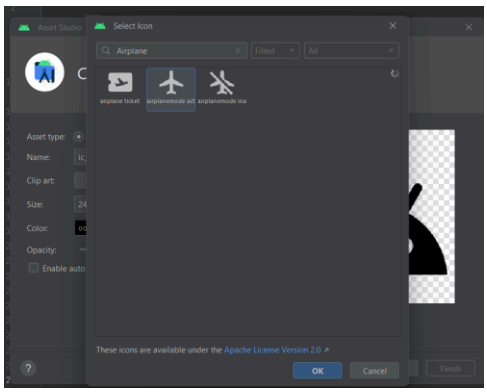


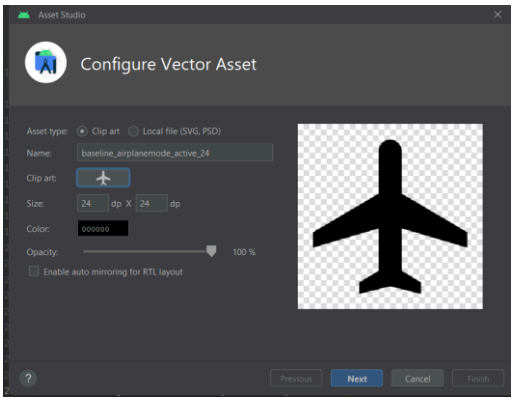
STEP 8: Once you are done with the logic part then you can add airplane mode icon.

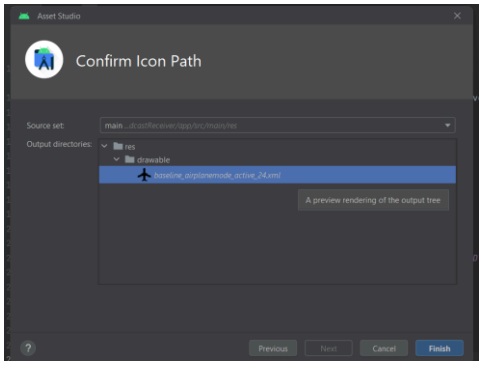
Go to drawable folder -> select new -> select Vector Asset



STEP 9: Click on the “clip art” option and type “Airplane”

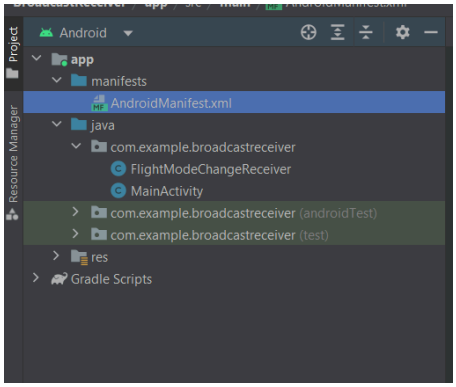




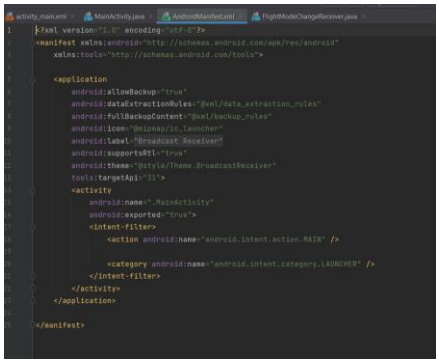


STEP 10: Now you have added the airplane mode icon in the drawable folder then,

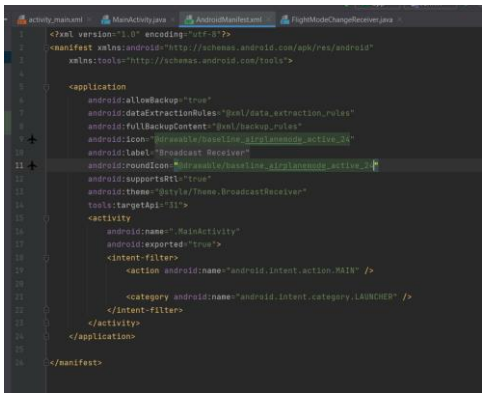
Go to the “AndroidManifest.xml” file



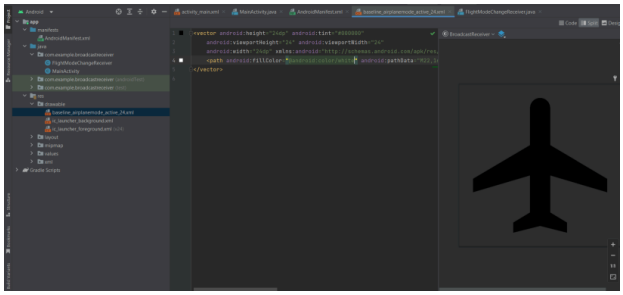
STEP 11: Then inside the “android: icon” property add your icon name which we have selected from the vector asset.



STEP 12: Also you add a new property which is “android:round icon”



STEP 13: To make the icon visible go to the icon which we have added in the drawable folder



STEP 14: Then change the color to black, now the icon will be visible