Android Developer

WORKING WITH GUI AND MULTIMEDIA ELEMENTS

Chapter 1

Access views

- In Android development, you need to define the views or components in the layout XML file and assign them an ID.
- Then, you can access these views in your Java file by referencing their IDs using the **findViewById()** method.

- import necessary Android packages such as **EditText, Button**, and **TextView** etc.

```java
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        // Get reference to the EditText field
        EditText editText = findViewById(R.id.editText);

        // Get reference to the Button
        Button button = findViewById(R.id.button);

        // Now you can use editText and button as needed
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Enter text here" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click me" />

</RelativeLayout>
```

Interactive and EventHandler

- Simple text changing event
- Designing a Click Listeners

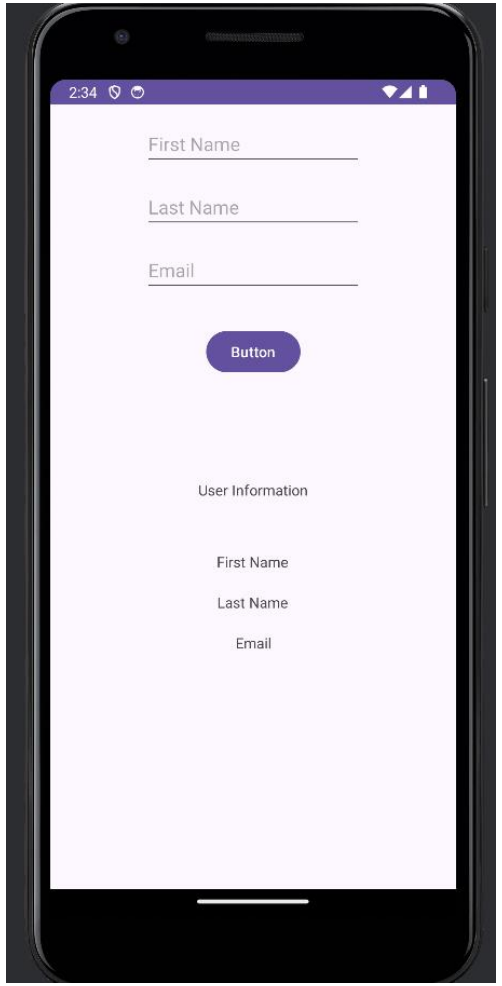
```
TextView helloMessage = findViewById(R.id.txtLabel1);
helloMessage.setText("I LOVE MSU");
```

```
// define your method here
1 usage
public void onButtonClick(View view) {
    // Handle button click event here
    // For example, show a toast message
    Toast.makeText(getApplicationContext(), text: "Button clicked", Toast.LENGTH_SHORT).show();
    /*...*/

    // you can also this task or many other tasks
    TextView helloMessage = findViewById(R.id.txtLabel1);
    helloMessage.setText("I LOVE MSU Hospital");
}
```

Toast

```
public void onButtonClick(View view) {
    // Handle button click event here
    // For example, show a toast message
    Toast.makeText(getApplicationContext(), "Button clicked", Toast.LENGTH_SHORT).show();
    /*You must import the following
    import android.widget.Toast;
    import android.view.View;
    */

    // you can also this task or many other tasks
    TextView helloMessage = findViewById(R.id.txtLabel1);
   // helloMessage.setText("I LOVE MSU Hospital");

    // another task
    EditText editTxtName = findViewById(R.id.editTxtName);
    String myNmae = editTxtName.getText().toString();

    // You can pass value in either of these ways
    //helloMessage.setText("Hello " + editTxtName.getText().toString());
    helloMessage.setText("Hello " + myNmae);

}
```

Registration Form




```java
public void onRegisterBtnClick(View view){

    //GETT THE VALUES FOR THE EDIT TEXT VIEW
    EditText editTxtFname = findViewById(R.id.editTxtFname);
    String getFirstName = editTxtFname.getText().toString();

    EditText editTxtLname = findViewById(R.id.editTxtLname);
    String getLastName = editTxtLname.getText().toString();

    EditText editTxtEmail = findViewById(R.id.editTxtEmail);
    String getEmail = editTxtEmail.getText().toString();

    //Create TextView objects TO GET THE VALUES FROM THE TEXTVIEWS
    TextView txtFirstName = findViewById(R.id.txtFirstName);
    TextView txtLastName = findViewById(R.id.txtLastName);
    TextView txtEmail = findViewById(R.id.txtEmail);

    //sET THE TEXT TO THE LABELS

    txtFirstName.setText("First Name: " + getFirstName);
    txtLastName.setText("Last Name: " + getLastName);
    txtEmail.setText("Email: " + getEmail);

}
```

- `setContentView(R.layout.activity_main);`
- Is a method responsible for setting the UI layout for an activity.
  - It displays its UI to the user.

Methods for EditText class

- In addition to the **getText**(), **setText**(), and **setHint**() methods, there are several other useful methods available for the EditText class in Android. Here are some commonly used ones:

1.Selection Methods:

1. **setSelection**(int index): Sets the cursor position in the EditText to the specified index.
2. **setSelection**(int start, int stop): Sets the selection range in the EditText from the specified start index to the stop index.

2.Transformation Methods:

1. **setInputType**(int type): Sets the input type for the EditText, controlling the keyboard behavior and input validation.
2. **setTransformationMethod**(TransformationMethod method): Sets the transformation method applied to the text, such as hiding passwords with asterisks.

3.Cursor Control Methods:

1. **clearFocus**(): Removes focus from the EditText, hiding the keyboard if it's currently visible.
2. **requestFocus**(): Requests focus for the EditText, showing the keyboard if necessary.

Conti....

1. **Text Change Listener Methods:**
    1. addTextChangedListener(TextWatcher watcher): Adds a TextWatcher to listen for changes in the text content of the EditText.
    2. removeTextChangedListener(TextWatcher watcher): Removes a previously added TextWatcher.
2. **Selection Control Methods:**
    1. selectAll(): Selects all the text in the EditText.
    2. extendSelection(int index): Extends the current selection from the current cursor position to the specified index.
3. **Cursor Movement Methods:**
    1. moveCursorToVisibleOffset(): Moves the cursor to the first visible position in the EditText if it's currently out of view.
4. **Error Handling Methods:**
    1. setError(CharSequence error): Sets an error message that will be displayed below the EditText if the input is invalid.
    2. setError(CharSequence error, Drawable icon): Sets an error message along with an icon.

Setting Click Listener for Button Using Button ID

- ```
  // this is another way of setting a click listener
  Button btnHello = findViewById(R.id.btnOneHello);

  btnHello.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
          System.out.println("Hello I am a click listener method");
          Toast.makeText(getApplicationContext(), "Button clicked",
  Toast.LENGTH_SHORT).show();
      }
  });
  ```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Button"
    android:layout_centerInParent="true"
    android:id="@+id/btnOneHello"/>
```

- Inside the **onClick**() method, the desired actions to be performed when the button is clicked are specified.
- Find the **hello world project**
```

Click listener By id

```
myButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Handle button click event here
        Toast.makeText(getApplicationContext(), text: "Button clicked", Toast.LENGTH_SHORT).show();

        // Perform the other tasks
        TextView helloMessage = findViewById(R.id.txtLabel1);
        EditText editTxtName = findViewById(R.id.editTxtName);

        // Get the name entered in the EditText
        String myName = editTxtName.getText().toString();

        // Set the text to the TextView
        helloMessage.setText("Hello " + myName);
    }
```

Button myButton = findViewById(R.id.*myButton*

Another way

```
public class MainActivity extends AppCompatActivity
implements View.OnClickListener {

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case btnOneHello:
                System.out.println("Hello I am a click listener
method");
                Toast.makeText(getApplicationContext(), "Button
clicked", Toast.LENGTH_SHORT).show();
                break;

            default:
                break;

        }
    }
```

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

    // this is another way of setting a click listener

    Button btnHello = findViewById(btnOneHello);
    btnHello.setOnClickListener(this);
}
```

Click listener for EditText

```
// creating a clisner for a editText view
EditText edtTxtName = findViewById(R.id.edtTxtName);
edtTxtName.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, "You are typing",
Toast.LENGTH_SHORT).show();
    }
});
```

Check boxes checked change listener

```
ckBox6 = findViewById(R.id.ckBox6);
ckBox6.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {
        if(isChecked){
            Toast.makeText(MainActivity.this, "Answered 3",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(MainActivity.this, "Not Answered 3",
Toast.LENGTH_SHORT).show();
        }
    }
});
```

```
ckBox5 = findViewById(R.id.ckBox5);
ckBox5.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {
        if(isChecked){
            Toast.makeText(MainActivity.this, "Answered 2",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(MainActivity.this, "Not Answered 2",
Toast.LENGTH_SHORT).show();
        }
    }
});
```

RadioGroup and RadioButton

```
<RadioButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Night"
   android:checked="true"
   android:id="@+id/rdBtnNight"
   android:layout_below="@+id/ckBox4"
   android:layout_marginTop="50dp"/>

<RadioButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Day"
   android:checked="true"
   android:id="@+id/rdBtnDay"
   android:layout_below="@+id/ckBox5"
   android:layout_alignStart="@id/ckBox5"
   android:layout_marginTop="50dp"
   android:layout_toRightOf="@+id/rdBtnNight"/>
```

```
<RadioGroup
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_below="@+id/ckBox4"
   android:layout_marginTop="50dp"
   android:layout_centerHorizontal="true">

   <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Night"
      android:checked="true"
      android:id="@+id/rdBtnNight"/>

   <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Day"
      android:checked="true"
      android:id="@+id/rdBtnDay"/>

</RadioGroup>
```

Check them without clicking them

```
int checkedRdBtn = rdGrpDayNight.getCheckedRadioButtonId();

if (checkedRdBtn == R.id.rdBtnNight) {
    Toast.makeText(MainActivity.this, "It is a Night", Toast.LENGTH_SHORT).show();
} else if (checkedRdBtn == R.id.rdBtnDay) {
    Toast.makeText(MainActivity.this, "It is a Day", Toast.LENGTH_SHORT).show();
}
else if (checkedRdBtn == R.id.rdBtnNoneOFthem) {
    Toast.makeText(MainActivity.this, "It is None of Them", Toast.LENGTH_SHORT).show();
}
```

Uibasics2 **project**

Progress Bar

```
<ProgressBar
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/progressBar1"
   android:layout_centerInParent="true"
   android:layout_below="@+id/rdGrpDayNight"
   android:layout_marginTop="50dp"
   android:visibility="visible">

</ProgressBar>
```

Type one

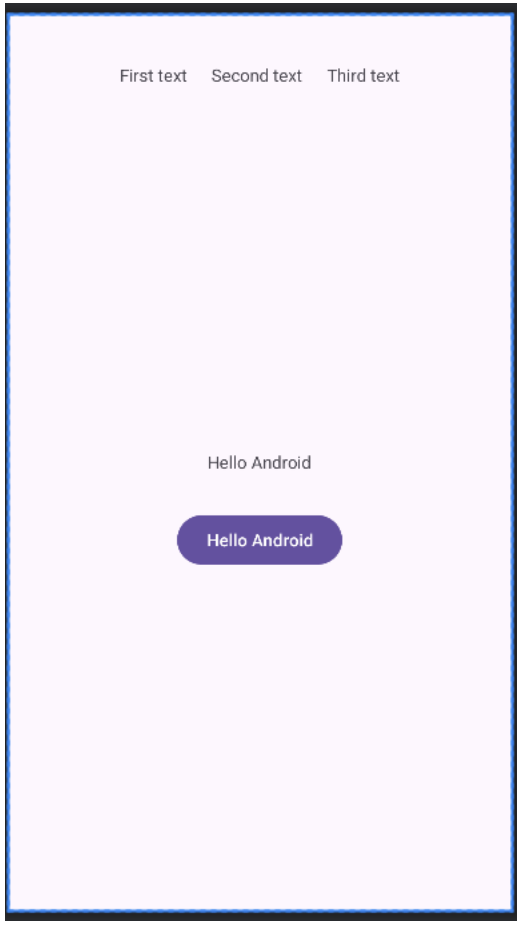
```
<ProgressBar
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:id="@+id/progressBar2"
   android:layout_centerInParent="true"
   android:layout_below="@+id/progressBar1"
   android:layout_marginTop="50dp"
   android:visibility="visible"
   style="@style/Widget.AppCompat.ProgressBar.Horizontal"
   android:progress="50"
   android:max="100"
   android:layout_marginLeft="20dp"
   android:layout_marginRight="20dp">

</ProgressBar>
```

Type two

Layouts

Layouts: Relative




```
<RelativeLayout
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:id="@+id/relativeLayout1"
  android:layout_centerHorizontal="true"
  android:layout_marginTop="40dp">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="First text"
    android:id="@+id/txtFirsttext"/>
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Second text"
    android:id="@+id/txtSecondtext"
    android:layout_toRightOf="@+id/txtFirsttext"
    android:layout_marginLeft="20dp"/>

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Third text"
    android:id="@+id/txtThirdtext"
    android:layout_toRightOf="@+id/txtSecondtext"
    android:layout_marginLeft="20dp"/>

</RelativeLayout>
```

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Hello Android"
  android:layout_centerInParent="true"
  android:id="@+id/txtHelloAndroid"/>

<Button
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Hello Android"
  android:id="@+id/btnHelloAndroid"
  android:layout_below="@+id/txtHelloAndroid"
  android:layout_centerHorizontal="true"
  android:layout_marginTop="30dp"/>
```

RelativeLayout

- is a one of the layouts in android that arran**ges its child views relative to each other** or **relative to the parent container**.
- It's **flexible** because **you can position elements based on their relationships**, such as aligning one view to the right of another, or positioning a view in the center of the parent.

RelativeLayout

- **Positioning Relative to Parent**:
- You can align a view relative to its parent container (the ***RelativeLayout*** itself).
- Common attributes include:
  - *android:layout_alignParentTop*: Aligns the view at the top of the parent.
  - *android:layout_alignParentBottom*: Aligns the view at the bottom of the parent.
  - *android:layout_centerInParent*: Centers the view horizontally and vertically within the parent.

RelativeLayout

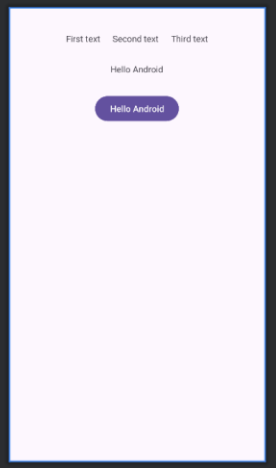
- **Positioning Relative to Other Views**:
- Views can also be aligned relative to other sibling views in the layout.
- Common attributes include:
  - *android:layout_toRightOf*: Positions the view to the right of another view.
  - *android:layout_below*: Positions the view below another view.
  - *android:layout_alignBaseline*: Aligns the baseline of the view with another view’s baseline (useful for aligning text).

```
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10     <RelativeLayout...>
39
40
41     <TextView
42         android:layout_width="wrap_content"
43         android:layout_height="wrap_content"
44         android:text="Hello Android"
45         android:layout_below="@+id/relativeLayout1"
46         android:layout_centerHorizontal="true"
47         android:layout_marginTop="30dp"
48         android:id="@+id/txtHelloAndroid"/>
49
50     <Button
51         android:layout_width="wrap_content"
52         android:layout_height="wrap_content"
53         android:text="Hello Android"
54         android:id="@+id/btnHelloAndroid"
55         android:layout_below="@+id/txtHelloAndroid"
56         android:layout_centerHorizontal="true"
57         android:layout_marginTop="30dp"/>
58
59 </RelativeLayout>
```

Component Tree

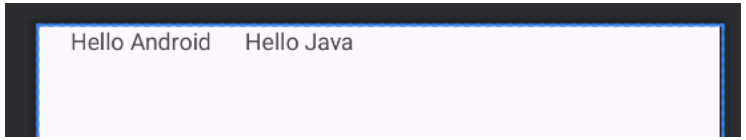
- main
  - relativeLayout1
    - txtFirsttext "First text"
    - txtSecondtext "Second text"
    - txtThirdtext "Third text"
  - txtHelloAndroid "Hello Android"
  - btnHelloAndroid "Hello Android"

activity_main.xml



Layouts: Linear

- They don’t have center in horizontal and vertical and center in parent, layout below.
- If the parent layout is linear and you want to use these, create a relative layout as child and put elements that you want to give such attributes.
- In linear everything is being placed next to each other.
- The orientation attribute this layout takes is either horizontal or vertical.




```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

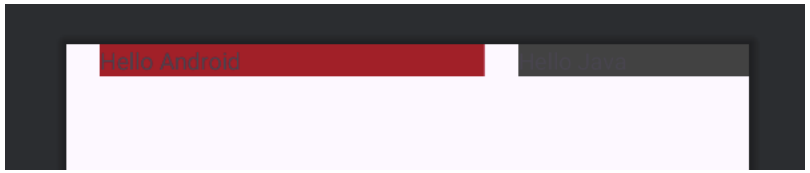
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello Android"
      android:id="@+id/txtHelloAndroid"
      android:layout_marginLeft="20dp"/>

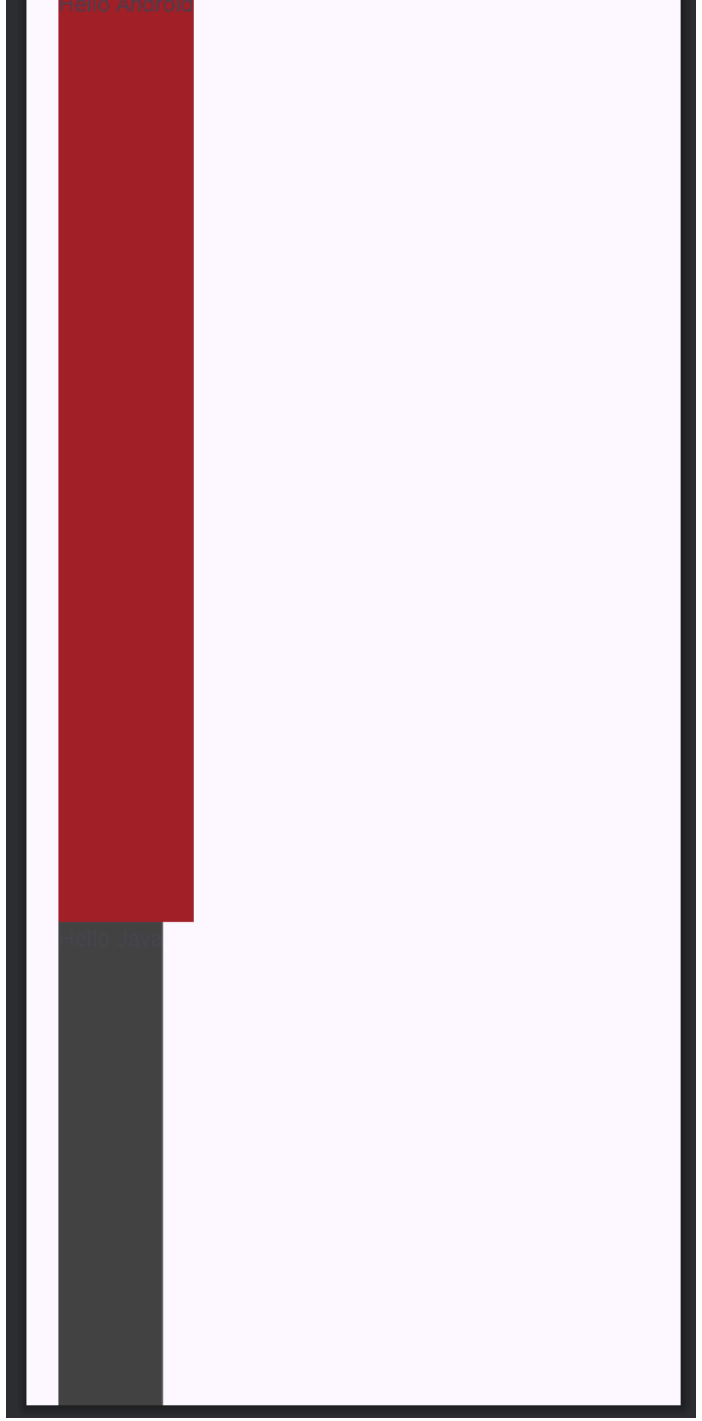
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello Java"
      android:id="@+id/txtHelloJava"
      android:layout_marginLeft="20dp"/>
</LinearLayout>
```

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello Android"
      android:id="@+id/txtHelloAndroid"
      android:layout_marginLeft="20dp"/>

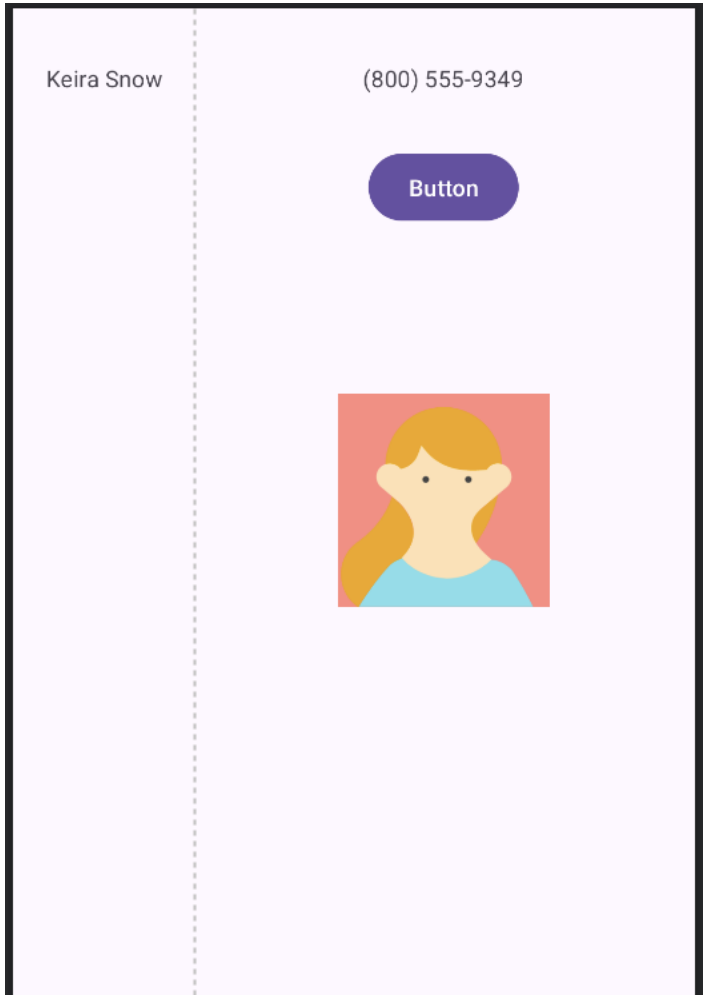
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello Java"
      android:id="@+id/txtHelloJava"
      android:layout_marginLeft="20dp"/>
</LinearLayout>
```






```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Android"
        android:id="@+id/txtHelloAndroid"
        android:layout_marginLeft="20dp"
        android:background="@color/design_default_color_error"
        android:layout_weight="60"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Java"
        android:id="@+id/txtHelloJava"
        android:layout_marginLeft="20dp"
        android:background="@color/cardview_dark_background"
        android:layout_weight="30"/>

</LinearLayout>
```

Layouts: Constraint



ImageView


```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:text="Hello World!"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="214dp"
    android:layout_height="209dp"
    android:layout_marginTop="32dp"
    app:layout_constraintEnd_toEndOf="@+id/textView"
    app:layout_constraintStart_toStartOf="@+id/textView"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:srcCompat="@mipmap/userimage"
    />
```

Change the App

- Go to the manifest file

```
android:icon="@mipmap/userimage"

android:roundIcon="@mipmap/userimage"
```

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.IMAGES"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category
android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

ListView

Go to the listview1 projects

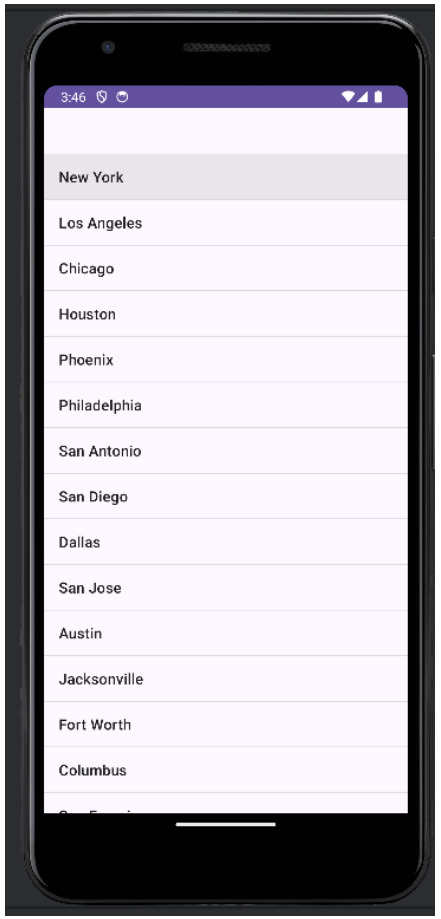
```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:id="@+id/cityListView"/>
```

```
private ListView cityListView;
```

```
    cityListView = findViewById(R.id.cityListView);
    ArrayList<String> listOfCities = new ArrayList<>();
```

```
    listOfCities.add("New York");
    listOfCities.add("Los Angeles");
    listOfCities.add("Seattle");
    listOfCities.add("Denver");
    listOfCities.add("Washington");
    listOfCities.add("Boston");
```

```
    // create adaoptor that gives this data to the listview
    ArrayAdapter<String> cityListAdapter = new ArrayAdapter<>(
        this,
        android.R.layout.simple_list_item_1,
        listOfCities
    );
    cityListView.setAdapter(cityListAdapter);
```



Click Listener for the ListView Items

```
cityListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
        //Toast.makeText(MainActivity.this, "The position is " +
position + " The id is " + id, Toast.LENGTH_SHORT).show();
        Toast.makeText(MainActivity.this, listOfCities.get(position),
Toast.LENGTH_SHORT).show();
    }
});
```

Spinner

Designed for displaying each item in a dropdown list

```
<Spinner
    android:id="@+id/cityListSpineer1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="15dp"/>
```

```
private Spinner countryListSpineer1;

// Spinner
countryListSpineer1 = findViewById(R.id.countryListSpineer1);

ArrayList<String> listOfCountries = new ArrayList<>();
listOfCountries.add("USA");
listOfCountries.add("Canada");
listOfCountries.add("Australia");
listOfCountries.add("United Kingdom");
listOfCountries.add("Germany");

// create Array Adapter
ArrayAdapter<String> countryListAdapter = new
ArrayAdapter<>(
        this,
        android.R.layout.simple_spinner_dropdown_item,
        listOfCountries
);

countryListSpineer1.setAdapter(countryListAdapter);
```

Spinner Item Selected Listener

```
// selected listner
countryListSpineer1.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
        Toast.makeText(MainActivity.this, "Selected " +
listOfCountries.get(position), Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

Create String-array in the strings.xml

Is a way of passing data statically.
Keeping the data in a static file instead of dynamically creating in the java file or run time.

```
<Spinner
    android:id="@+id/countryListSpineer1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="15dp"
    android:entries="@array/cars"/>
```

activity_main.xml

```
<string-array name="cars">
    <item>Toyota</item>
    <item>BMW</item>
    <item>Nissan</item>
    <item>Tesla</item>
    <item>Proton</item>
</string-array>
```

strings.xml

```
// selected listner
countryListSpineer1.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View
view, int position, long id) {
        Toast.makeText(MainActivity.this, "Selected " +
countryListSpineer1.getSelectedItem().toString(),
Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

MainActivity.java Java file

Different XML files

MainActivity.java Java file

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Hello World!"
   app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toTopOf="parent"
   android:layout_centerInParent="true"
   android:id="@+id/textView"/>
```

```
private TextView textView, textView2;

textView = findViewById(R.id.textView);
textView2 = findViewById(R.id.textView2);

textView.setText("My Application"); // it has warning
textView2.setText(getString(R.string.myApp)); // no warning
```

activity_main.xml

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_below="@+id/textView"
   android:layout_centerInParent="true"
   android:layout_marginTop="50dp"
   android:id="@+id/textView2"/>
```

```
<string name="myApp">My Application 2</string>
```

strings.xml

Color.xml file

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="black">#FF000000</color>
   <color name="white">#FFFFFFFF</color>
   <color name="green">#FF34A853</color>
</resources>
```

Intent

- Intent is an object used to facilitate communication between different components of an app or even between different apps.
- It allows you to **request an action**, such as **starting a new activity**, **sending data**, or **invoking system services**.

```
// Create an Intent to navigate to SecondActivity
Intent intent = new Intent(MainActivity.this, MainActivity2.class);

// Optionally, pass data to SecondActivity
//intent.putExtra("firstName", getFirstName);
//intent.putExtra("lastName", getLastName);
//intent.putExtra("email", getEmail);

// Start SecondActivity
startActivity(intent);
```

intent.putExtra("firstName", firstName);

- **1. First Argument ("firstName"): The Key**
  - **Purpose**: The first argument is a **key** (or identifier) that acts as a label to identify the data you're passing.
  - This key is a String and must be unique if you're passing multiple pieces of data.
  - In your example, "firstName" is the key that you use later in the SecondActivity to retrieve the value.
- **2. Second Argument (firstName): The Value**
  - **Purpose**: The second argument is the **value** you are passing, which in this case is the actual data from the EditText that you want to transfer.
  - This value can be a string, integer, boolean, or other data type. In your example, firstName is the string value the user has entered in the EditText.

MainActivity.java

```java
btn1 = findViewById(R.id.btn1);

btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Create an Intent to navigate to SecondActivity
        Intent intent = new Intent( packageContext: MainActivity.this, MainActivity2.class);

        edtxtMain = findViewById(R.id.edtxtMain);

        // Get the text from EditText
        String firstName = edtxtMain.getText().toString();

        // Put the data (firstName) into the intent
        intent.putExtra( name: "firstName", firstName);

        // Start SecondActivity
        startActivity(intent);

    }
});
```

SecondActivity.java

```java
public class MainActivity2 extends AppCompatActivity {
2 usages
public EditText edtxtSecond;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main2);

        edtxtSecond = findViewById(R.id.edtxtSecond);

        // Get the intent that started this activity
        Intent intent = getIntent();

    // Retrieve the string data sent from MainActivity
        String firstName = intent.getStringExtra( name: "firstName");

    // Set the retrieved string to the EditText
        edtxtSecond.setText(firstName);

    }
}
```

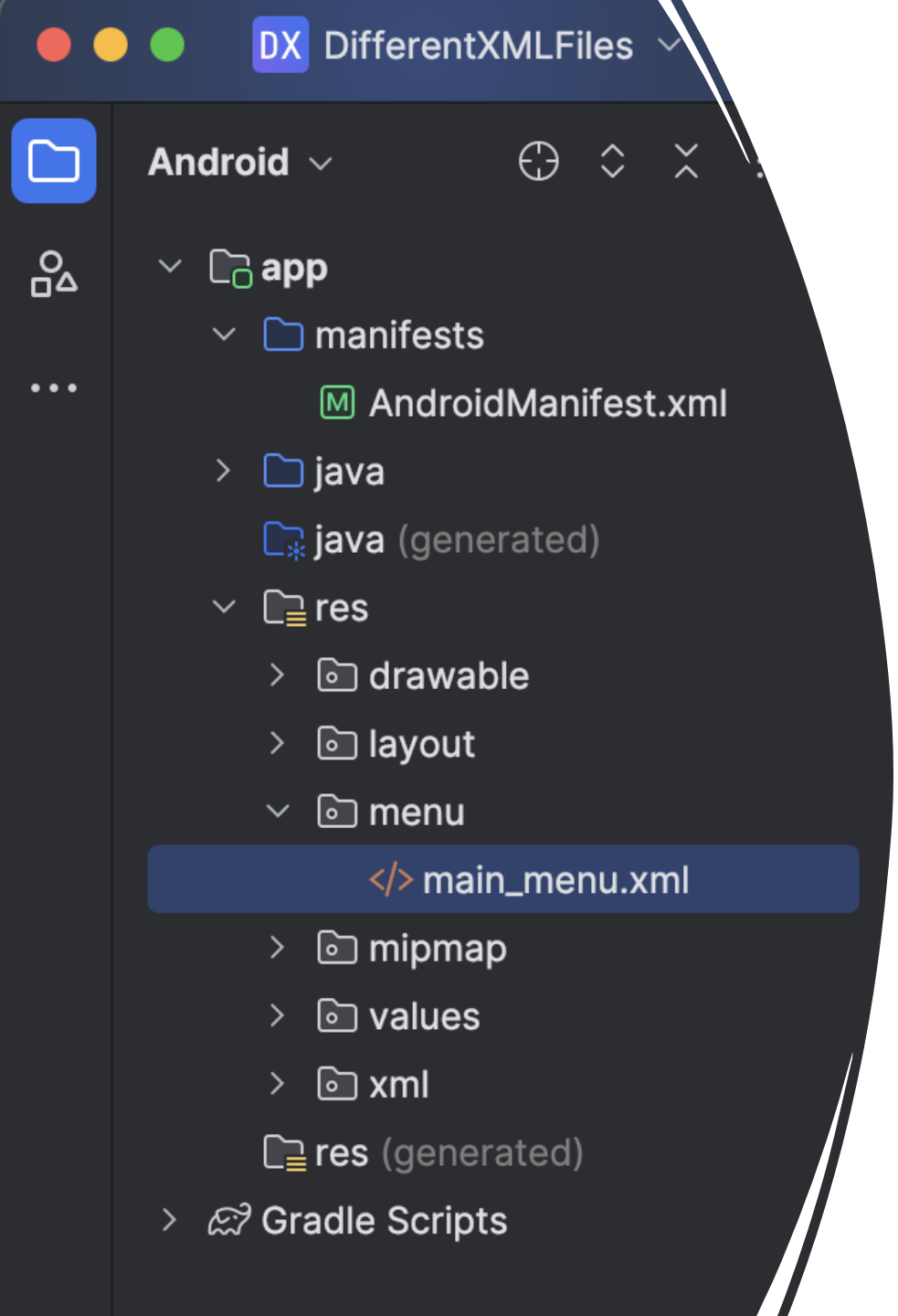
Themes

```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.DifferentXMLFiles" parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your dark theme here. -->
        <!-- <item name="colorPrimary">@color/my_dark_primary</item> -->
    </style>

    <style name="newAppTheme" parent="Theme.MaterialComponents.Light.DarkActionBar.Bridge">
        <!-- Customize your dark theme here. -->
        <item name="colorPrimaryDark">@color/green</item>
        <item name="colorPrimaryFixed">@color/material_dynamic_tertiary20</item>
        <item name="colorPrimaryFixedDim">@color/design_default_color_error</item>
    </style>
</resources>
```

Manifest File

- Is the place we define the general features and attributes of our application

```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">

  <application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.DifferentXMLFiles"
    tools:targetApi="31">
    <activity
      android:name=".MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```




```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">
<item
  android:title="Register"
  android:id="@+id/registerMenu"
  android:icon="@drawable/ic_register"
  app:showAsAction="always"/>

<item
  android:id="@+id/examMenu"
  android:icon="@drawable/ic_exam"
  android:title="Exam"
  app:showAsAction="always" />

</menu>
```

Create Menu file

When you want to create menu items for your app.

Make sure you define your titles name to the string file and icons into the drawable file

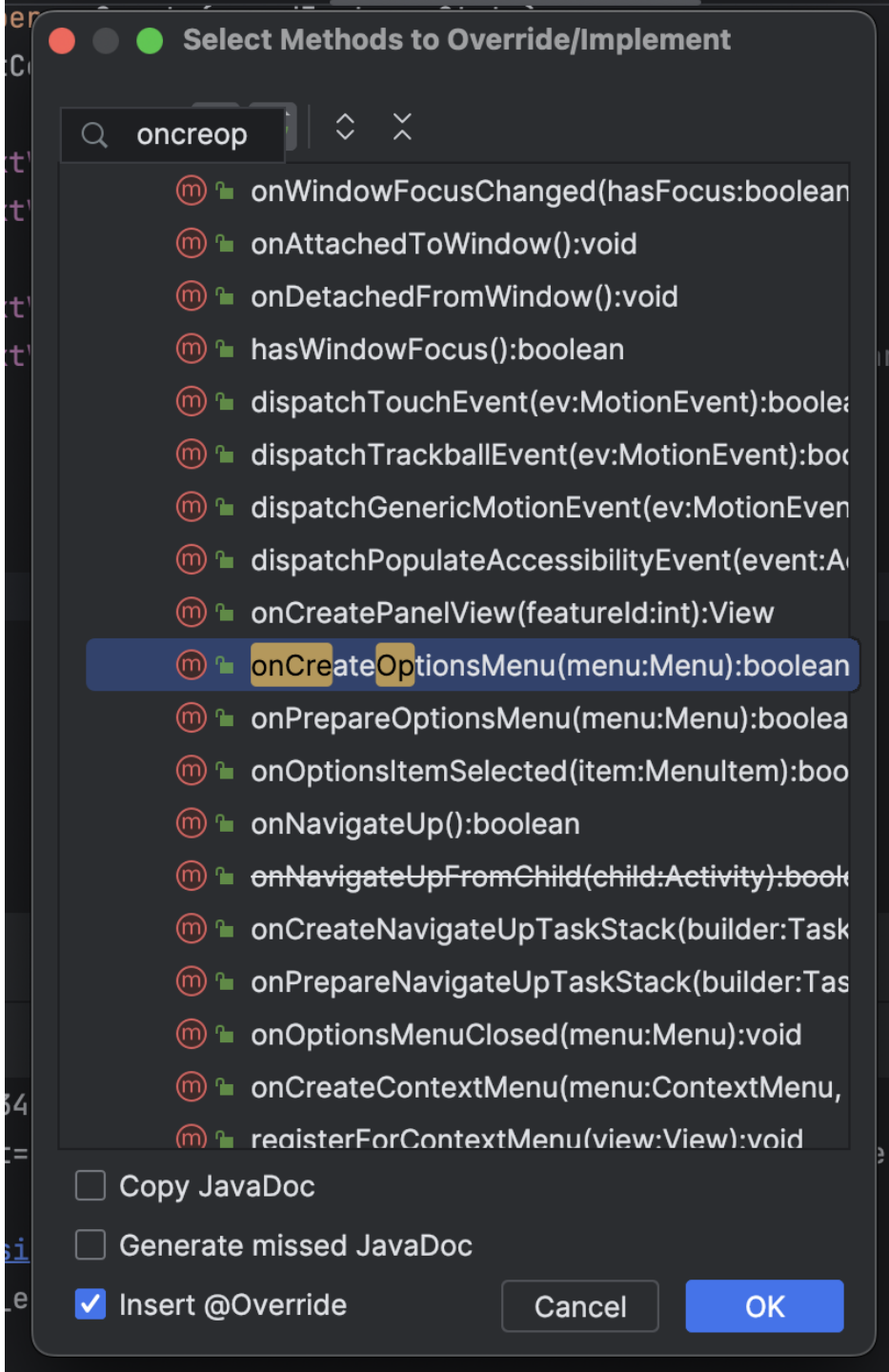
Create Icon assets

- Go to drawable right click, and create new image asset.
- Set the properties and finish.

Passing your menus to the main activity

- Go to the java file and overwrite it.
- Press ctr + o, select onCreateOptionMenu method
- Then Write this code.
- If you cannot see the menu try to change your app theme `android:theme="@style/Theme.AppCompat.Light"`

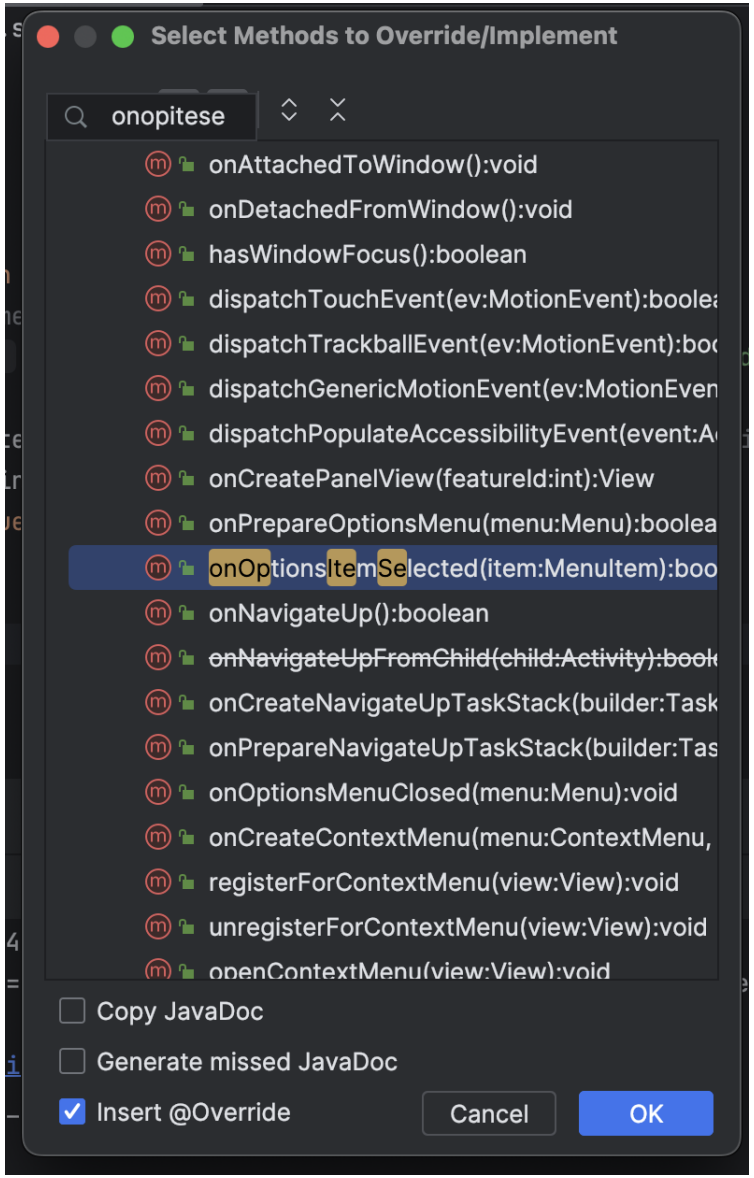
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater(); // this is like findViewById

    inflater.inflate(R.menu.main_menu, menu);
    return true;
}
```



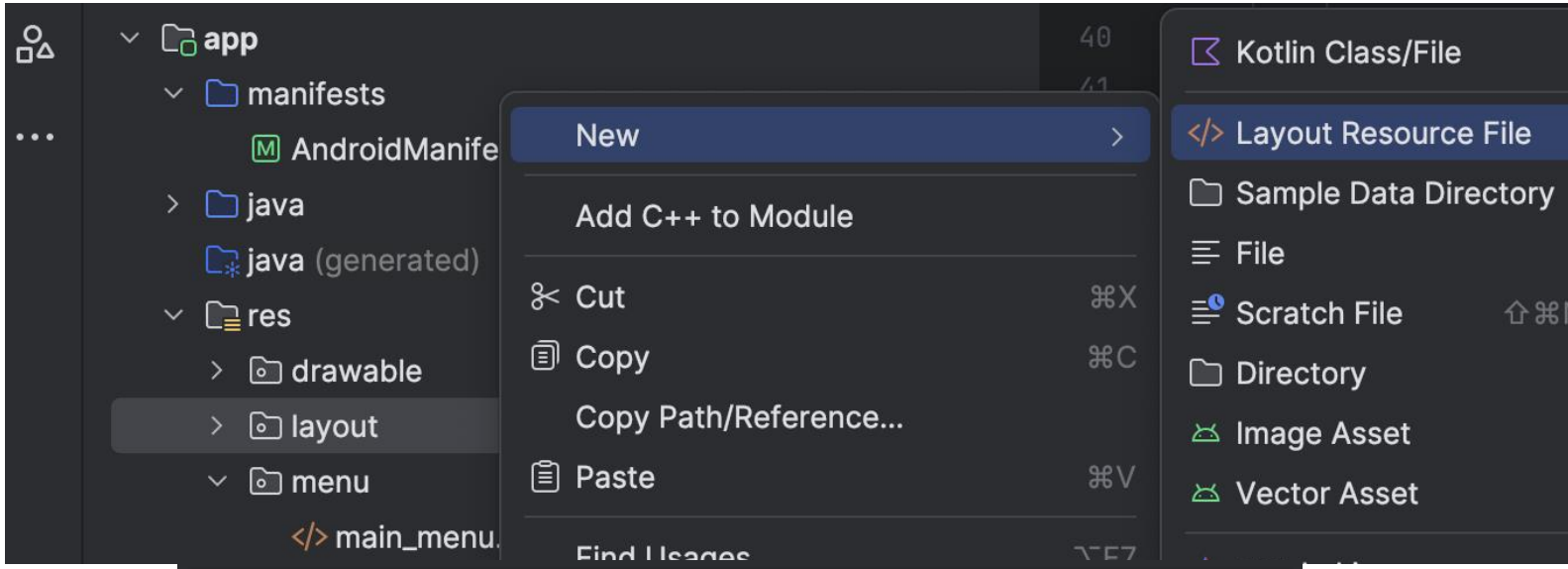
Set Click Listener to the menu items

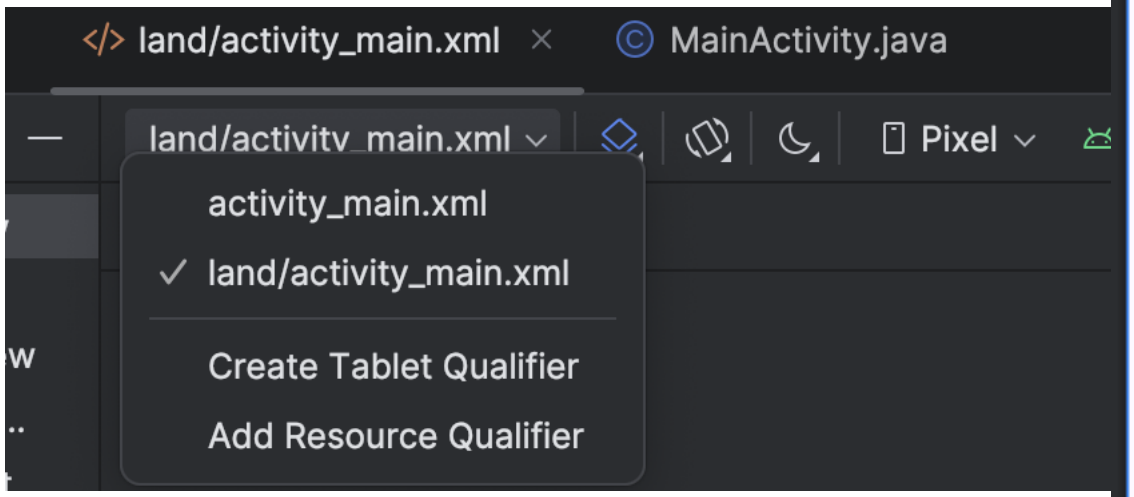
- Ctrl + o, select onOptionsItemSelected method

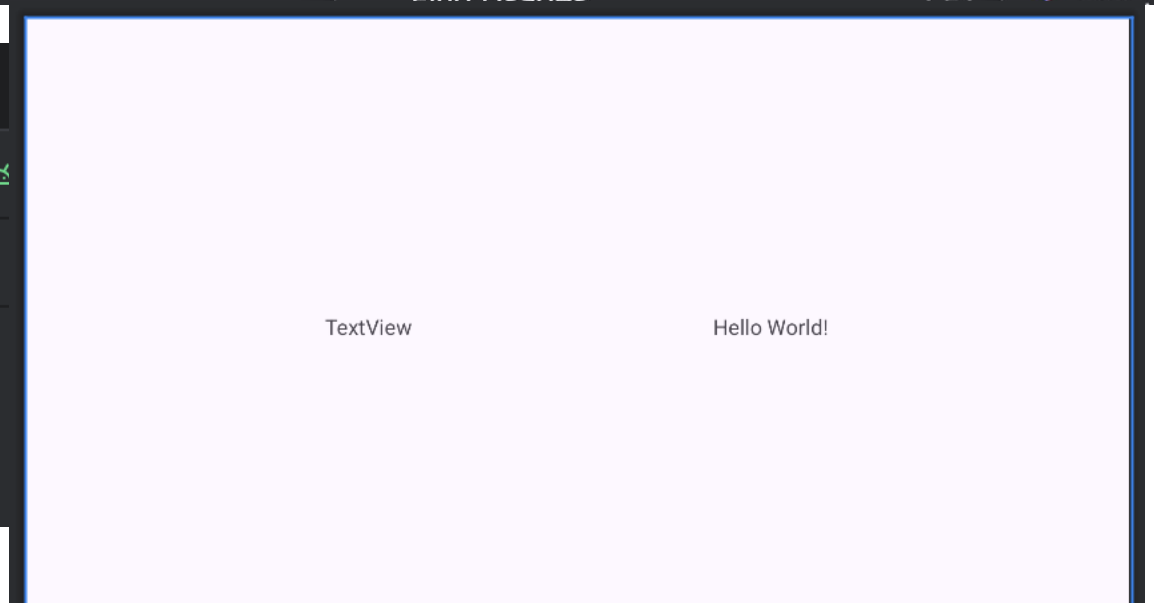
```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if (item.getItemId() == R.id.registerMenu) {
        Toast.makeText(this, "Register is selected",
Toast.LENGTH_SHORT).show();
        return true;
    } else if (item.getItemId() == R.id.examMenu) {
        Toast.makeText(this, "Exam is selected",
Toast.LENGTH_SHORT).show();
        return true;
    } else {
        return super.onOptionsItemSelected(item);
    }
}
```



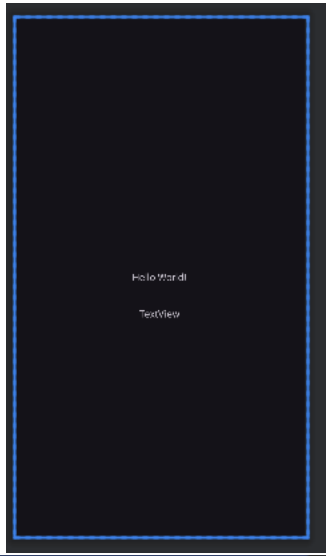
Create new layout

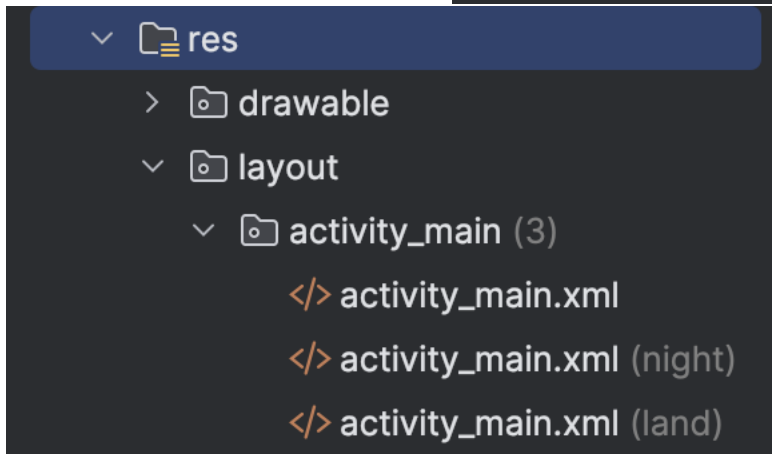


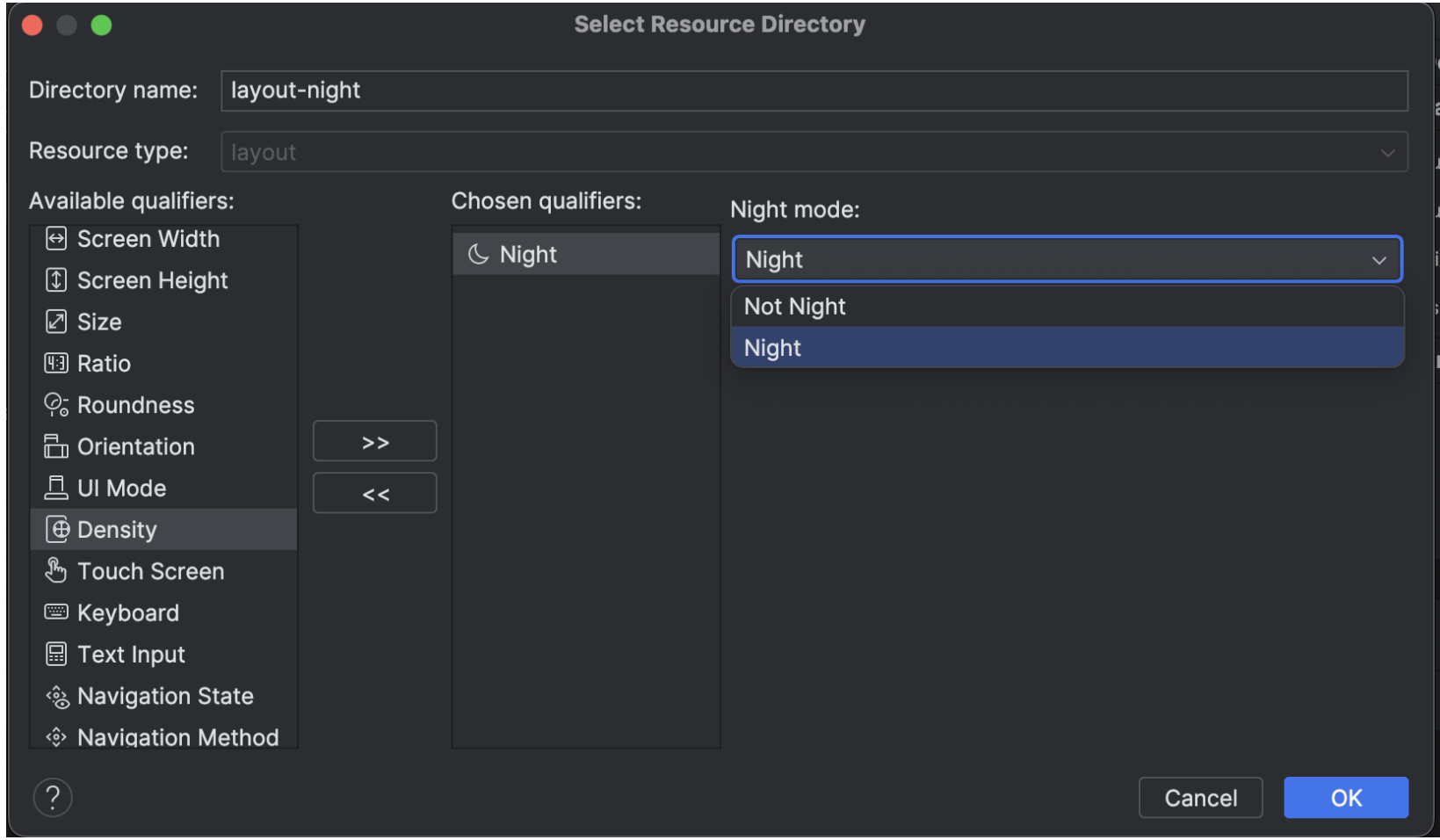




Create night mode layout



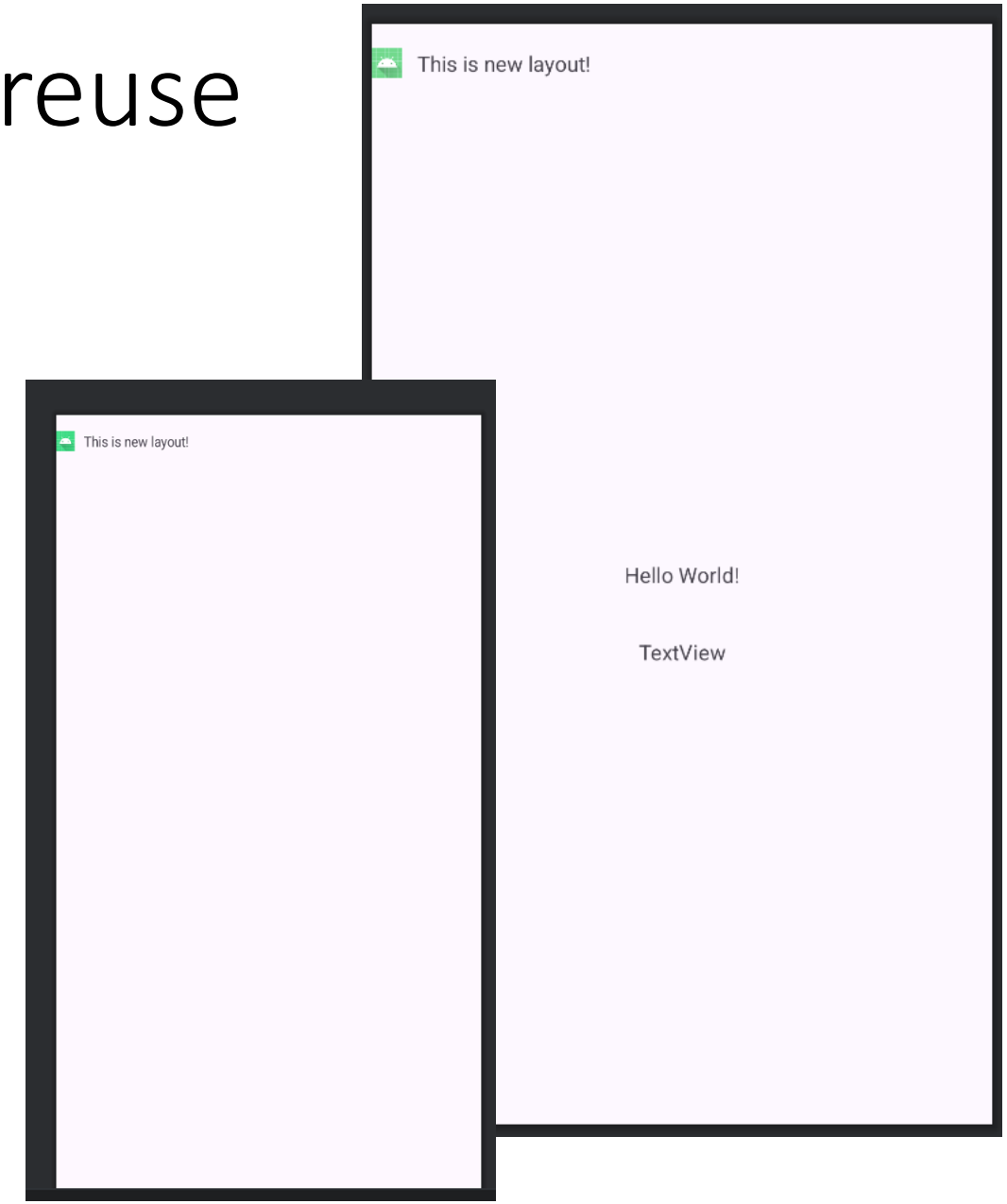




Create a new layout for reuse

- Put elements in the new layout and reuse the layout for other purpose.
- Use the include tag

```
<include layout="@layout/trademark"/>
```



You can overwite the style of the new layout in the main layout

```
<include layout="@layout/trademark"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"/>
```

Material Design Library

- Is a guideline for designing your application
- Components and themes you can use in your application
- To use and not to use examples
- Styling of different components
- To use the material design you must add their dependency

app
Gradle Scripts
build.gradle.kts (Project: MaterialDesignLibrary)

```
dependencies {
  classpath("com.android.tools.build:gradle:7.0.4") // Make sure to use the latest version of the Android Gradle Plugin
}
```

Gradle Scripts
build.gradle.kts (Project: MaterialDesignLibrary)
build.gradle.kts (Module :app)

```
implementation("com.google.android.material:material:1.11.0")
```

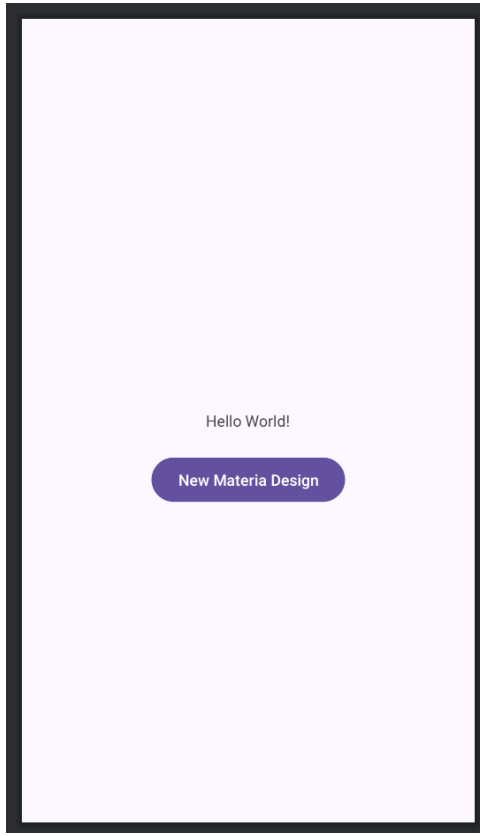
settings.gradle.kts (Project Settings)

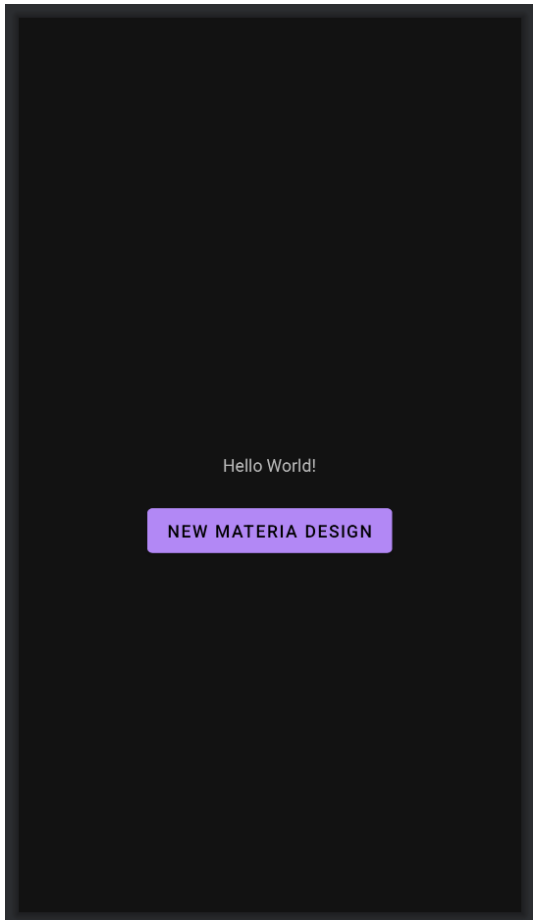
```
repositories {
  google()
  mavenCentral()
}
```

Apply the new theme

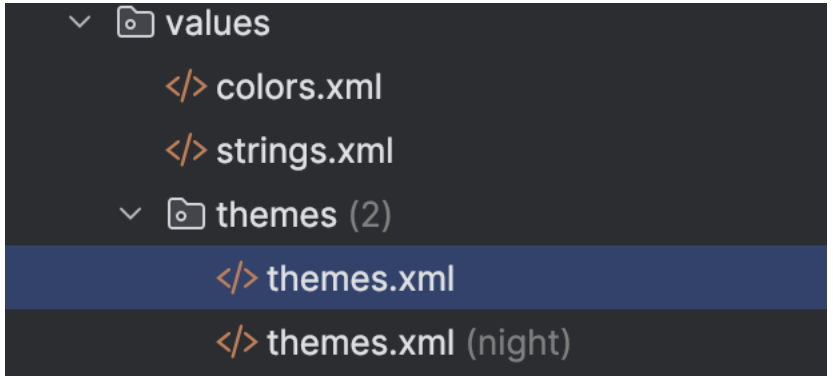
- After the material design dependency setup
- Adopt the new material design
- Go the themes in the values folder

```
<resources xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <style name="Base.Theme.MaterialDesignLibrary" parent="Theme.MaterialComponents">
    <!-- Customize your light theme here. -->
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
  </style>

  <style name="Theme.MaterialDesignLibrary" parent="Base.Theme.MaterialDesignLibrary" />
</resources>
```





After change



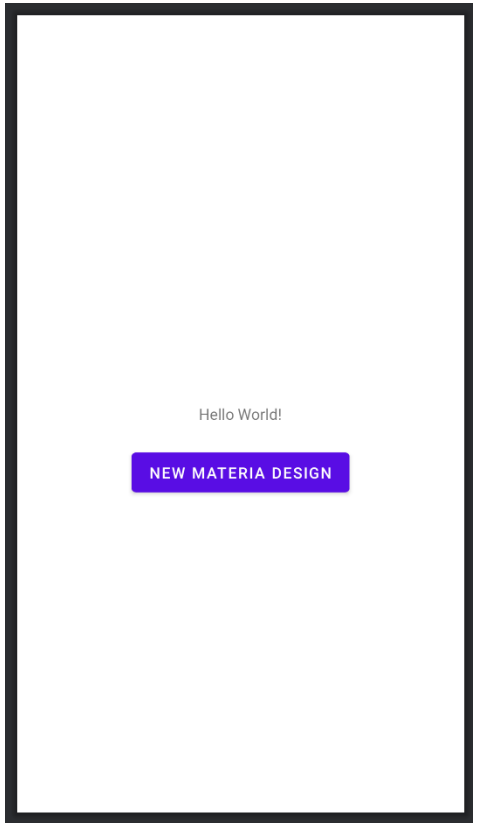
Before change

```
<resources xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <style name="Base.Theme.MaterialDesignLibrary" parent="Theme.Material3.DayNight.NoActionBar">
    <!-- Customize your light theme here. -->
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
  </style>

  <style name="Theme.MaterialDesignLibrary" parent="Base.Theme.MaterialDesignLibrary" />
</resources>
```

Play with changes

Try to change the style of the button etc.
Note some themes resist to set a background of the button, try to get clean theme.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Materia Design"
    android:layout_centerInParent="true"
    android:layout_below="@+id/txtHelloWorld"
    android:layout_marginTop="20dp"
    android:background="@color/green"
    style="@style/TextAppearance.AppCompat.Headline"/>
```




```
<style name="Base.Theme.MaterialDesignLibrary" parent="Theme.MaterialComponents.Light">
```

Extra components that the new theme support

- Add **FloatingActionButton**

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentEnd="true"
    android:layout_marginBottom="20dp"
    android:layout_marginRight="20dp"/>
```

Use layout marginEnd instead of marginRight which is given warning.

Add Animation to the **FloatingActionButton**

- Define the bounce animation in the **res/animator** directory. Create a new XML file named **bounce.xml**:

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:propertyName="translationY"
        android:valueFrom="0"
        android:valueTo="-50"
        android:duration="100"
        android:interpolator="@android:interpolator/accelerate_decelerate"
        android:startOffset="0"/>
    <objectAnimator
        android:propertyName="translationY"
        android:valueFrom="-50"
        android:valueTo="0"
        android:duration="200"
        android:interpolator="@android:interpolator/bounce"
        android:startOffset="100"/>
</set>
```

Apply the animation to the FloatingActionButton:

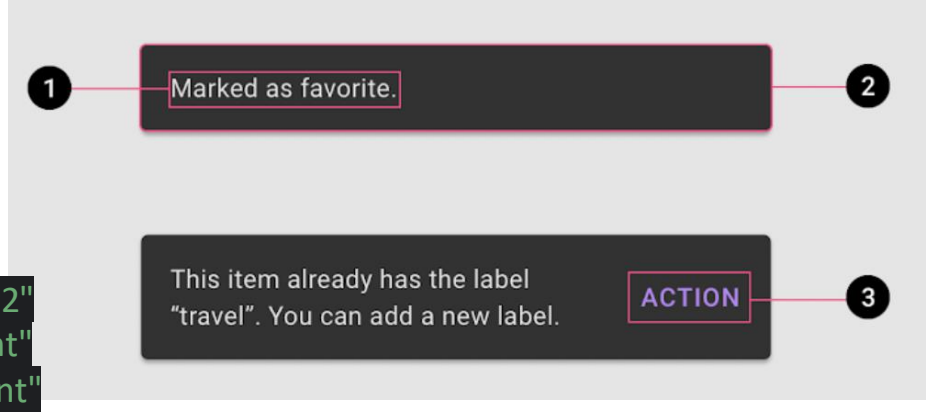
```xml
<com.google.android.material.floatingactionbutton.FloatingActionButton
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentBottom="true"
  android:layout_alignParentEnd="true"
  android:layout_marginBottom="20dp"
  android:layout_marginEnd="20dp"
  android:src="@drawable/ic_plus"
  android:backgroundTint="@color/maroon"
  app:backgroundTint="#FFF"
  app:rippleColor="@color/green"
  
app:layout_behavior="com.google.android.material.behavior.HideBottomViewOnScrollBehavio
r"
  android:onClick="onFabClick"
  android:id="@+id/fABtn"/>
```

Add an **onClick** attribute to the FloatingActionButton

```java
public void onFabClick(View view) {
    AnimatorSet animatorSet = (AnimatorSet)
AnimatorInflater.loadAnimator(
            this,
            R.animator.bounce);
    animatorSet.setTarget(view);
    animatorSet.start();
}
```

```java
import android.animation.AnimatorSet;
import android.os.Bundle;
import android.view.View;
import android.animation.AnimatorInflater;
```

SnackBar




```
private View parent;
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:text="Show SnackBar "
    android:id="@+id/btnShowSnackBar"/>
```

```
<Button
    android:id="@+id/btnShowSnackBar2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Show SnackBar 2"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="@+id/btnShowSnackBar"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="@+id/btnShowSnackBar"
    app:layout_constraintTop_toBottomOf="@+id/btnShowSnackBar"
    app:layout_constraintVertical_bias="0.163" />
```

```
// intialize the component
parent = findViewById(R.id.parent);
btnShowSnackBar = findViewById(R.id.btnShowSnackBar);
btnShowSnackBar2 = findViewById(R.id.btnShowSnackBar2);

btnShowSnackBar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showSnackBar();
    }
});
// declare a seprate method and call insdie the btn click event
private void showSnackBar(){
    Snackbar.make(parent,"This is a SnackBar",Snackbar.LENGTH_SHORT).show();
}
```

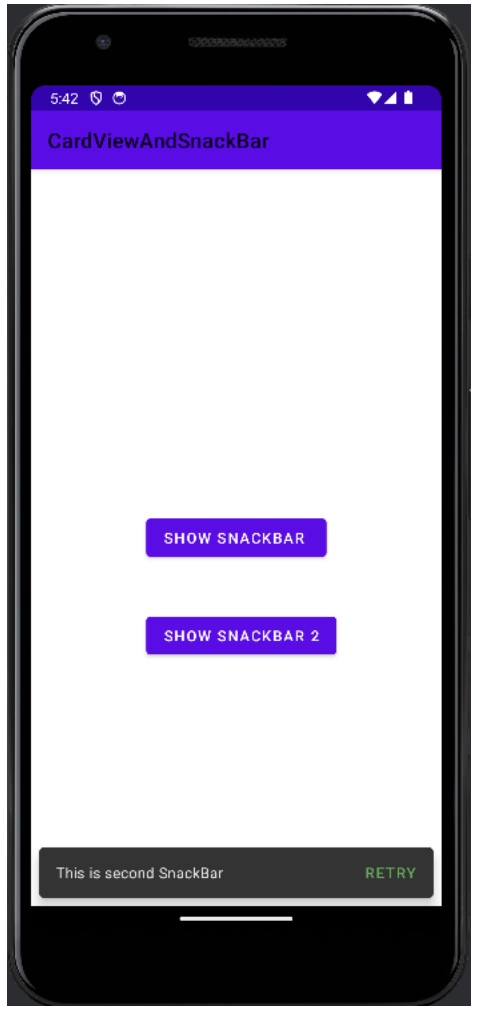
```
btnShowSnackBar2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Snackbar.make(parent,"This is second SnackBar",Snackbar.LENGTH_INDEFINITE)
            .setAction("Retry", new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Toast.makeText(MainActivity.this, "Thanks", Toast.LENGTH_SHORT).show();
                    //you can also show another snackbar if you want
                    showSnackBar();
                }
            })
            .show();
    }
});
```

Change the color of the SnackBar

```
setActionTextColor(getColor(R.color.green))
```

Fulll code in the main java file

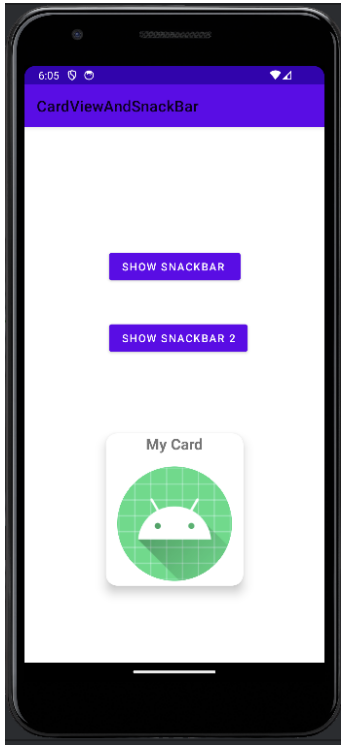
```
btnShowSnackBar2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Snackbar.make(parent,"This is second SnackBar",Snackbar.LENGTH_INDEFINITE)
                .setAction("Retry", new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Toast.makeText(MainActivity.this, "Thanks", Toast.LENGTH_SHORT).show();
                        //you can also show another snackbar if you want
                        showSnackBar();
                    }
                })
                .setActionTextColor(getColor(R.color.green))
                .show();
    }
});
```

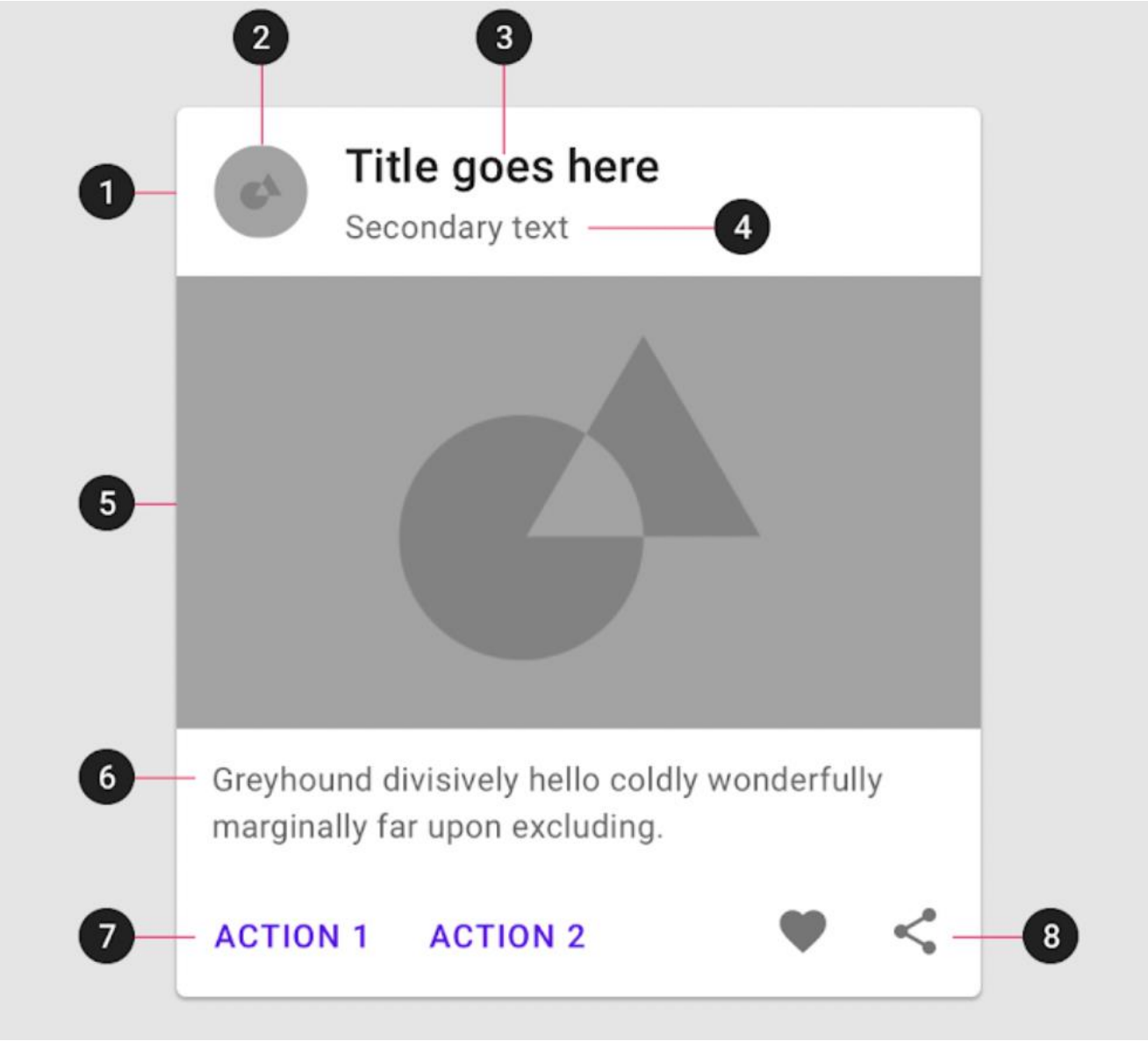


CradView

- In a CardView you can add multiple view elements

```
<com.google.android.material.card.MaterialCardView
  android:id="@+id/cardView1"
  android:layout_width="180dp"
  android:layout_height="200dp"
  app:layout_constraintBottom_toBottomOf="parent"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintStart_toStartOf="parent"
  app:layout_constraintTop_toBottomOf="@+id/btnShowSnackBar2"
  app:cardCornerRadius="15dp"
  app:cardElevation="8dp">

  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="My Card"
      android:layout_centerHorizontal="true"
      android:id="@+id/txtMyCard"
      android:textStyle="bold"
      android:textSize="20dp"/>
    <ImageView
      android:layout_width="150dp"
      android:layout_height="match_parent"
      android:layout_below="@+id/txtMyCard"
      android:src="@mipmap/ic_launcher"
      android:layout_marginTop="10dp"
      android:layout_centerHorizontal="true"/>
  </RelativeLayout>

</com.google.android.material.card.MaterialCardView>
```



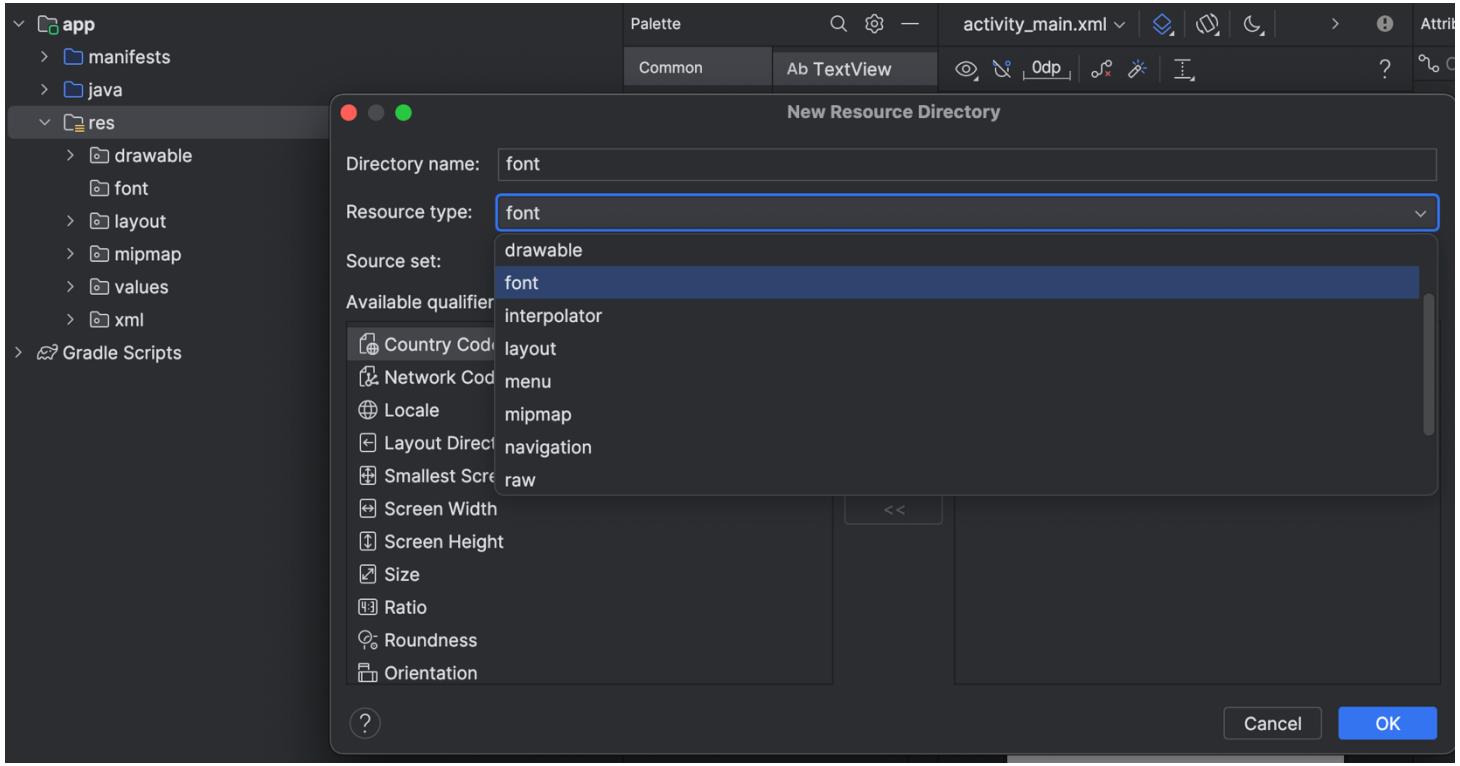


CardView Click listner

```
private MaterialCardView cardView1;

// the cardView1
cardView1 = findViewById(R.id.cardView1);
cardView1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, "Card View Element",
Toast.LENGTH_SHORT).show();
    }
});
```

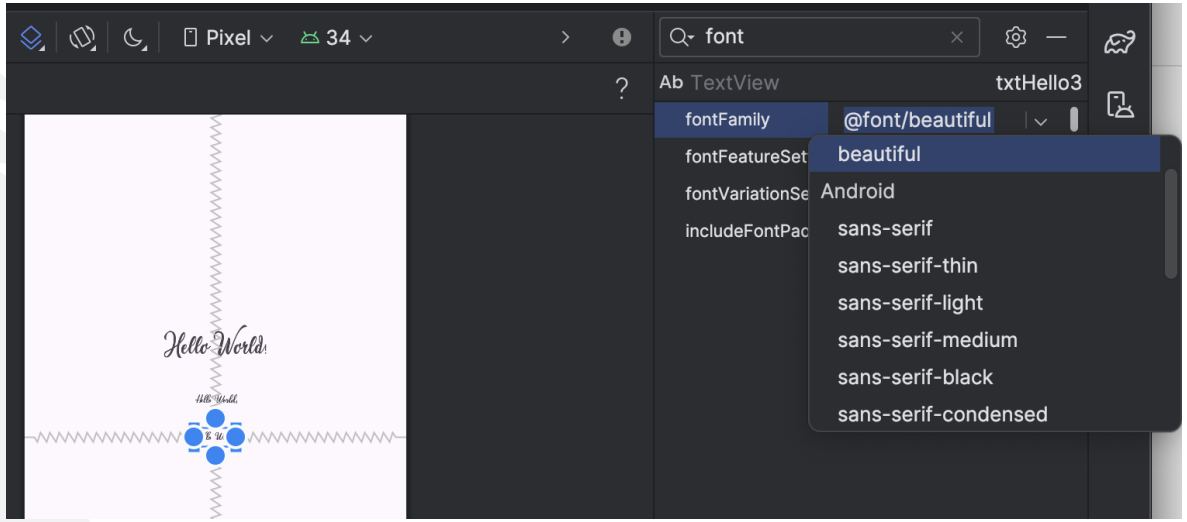
Add External Fonts

- Go to the res folder and create new resource folder, select the font and finish.
- Download and rename the font downloaded from internet.
- Then add in the font folder.

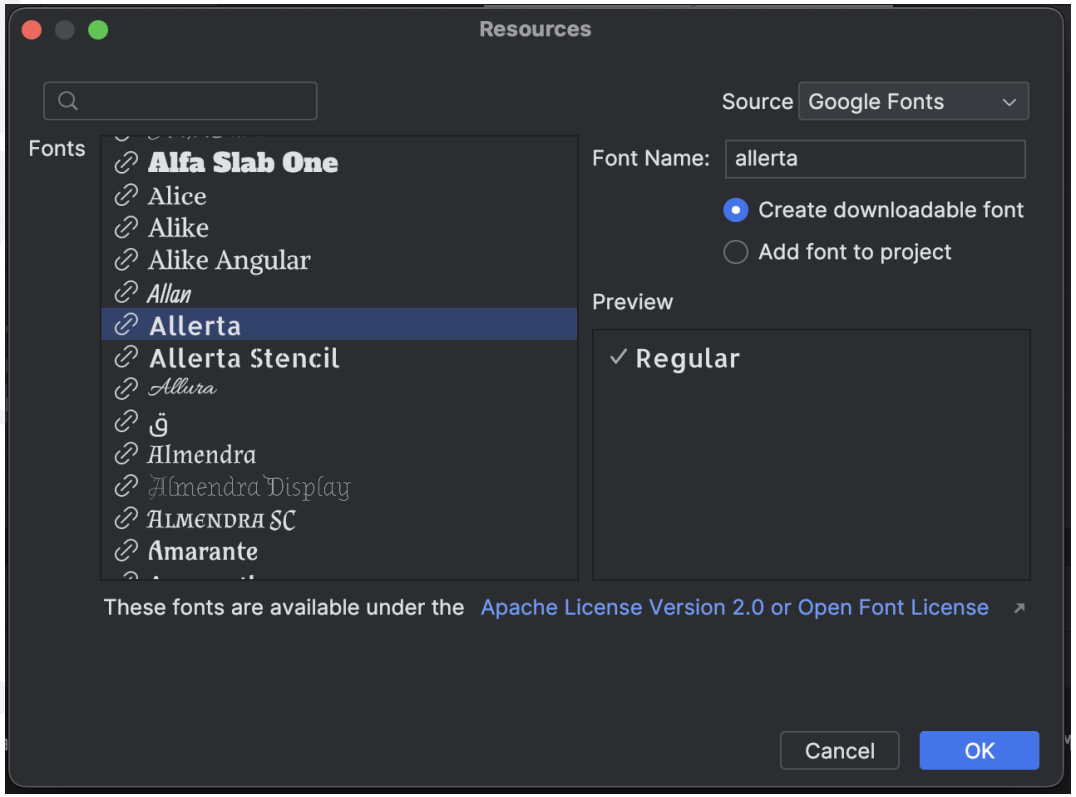


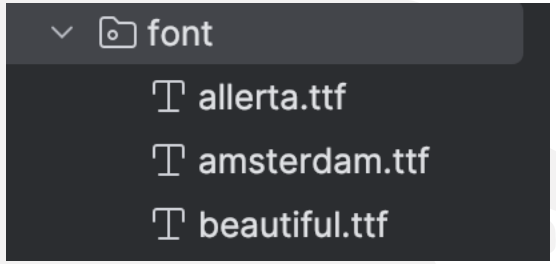

```
<TextView
    android:id="@+id/txtHello"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:layout_centerInParent="true"
    android:fontFamily="@font/amsterdam"/>
```

```
<TextView
    android:id="@+id/txtHello2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:layout_centerInParent="true"
    android:layout_below="@+id/txtHello"
    android:layout_marginTop="20dp"
    android:fontFamily="@font/beautiful"/>
```



- Add google font
- Go to the search in the attribute form,
- Click the font dropdown menu,
- Click more fonts





Change font in eventClick

```
private TextView txtHello5;
private Button btnChangeFont;


txtHello5 = findViewById(R.id.txtHello5);
btnChangeFont = findViewById(R.id.btnChangeFont);

Typeface typeface = ResourcesCompat.getFont(this,R.font.beautiful);

btnChangeFont.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        txtHello5.setTypeface(typeface);
        Toast.makeText(MainActivity.this, "Font changed", Toast.LENGTH_SHORT).show();
    }
});
```

```
<TextView
    android:id="@+id/txtHello5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/txtHello4"
    android:layout_centerInParent="true"
    android:layout_marginTop="20dp"
    android:fontFamily="@font/font_family"
    android:textStyle="italic"
    android:text="Hello World!" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Change Font"
    android:id="@+id/btnChangeFont"
    android:layout_below="@+id/txtHello5"
    android:layout_centerInParent="true"
    android:layout_marginTop="20dp"/>
```

Some Check Points

- See more the **ChallengeOne2** project

```
// validate if all filled or not
1 usage  new *
private boolean validate(){
    if (edtFullName.getText().toString().equals("")) {
        txtWarningName.setVisibility(View.VISIBLE);
        return false;
    }

    if (edtExperience.getText().toString().equals("")) {
        txtWarningExperience.setVisibility(View.VISIBLE);
        return false;
    }

    if (edtPreviousRole.getText().toString().equals("")) {
        txtWarningRole.setVisibility(View.VISIBLE);
        return false;
    }
    return true;
}
```

Checking password match

```
if (!edtTxtPassword.getText().toString().equals(edtTxtPassRepeat.getText().toString())) {
    txtWarnPassRepeat.setVisibility(View.VISIBLE);
    txtWarnPassRepeat.setText("Password doesn't match");
    return false;
}
```

Launch new activity

- Go to ChallengeOne project

```
// launch new activity
btnRegister = findViewById(R.id.btnRegister);
new *
btnRegister.setOnClickListener(new View.OnClickListener() {
    new *
    @Override
    public void onClick(View v) {
        Intent i = new Intent( packageContext: MainActivity.this, SecondActivity.class);
        startActivity(i);

    }
});
```

Variables and Arithmetic Operators