Lab Manual

Mobile Application Development Using Android Studio

By:

Wajeeha Azmat

Lecturer

Department of Computer Science

University of Agriculture Faisalabad

How to install Android Studio:

Android Studio is the official IDE (Integrated Development Environment) for Android app development and it is based on JetBrains' IntelliJ IDEA software. Android Studio provides many excellent features that enhance productivity when building Android apps, such as:

- A blended environment where one can develop for all Android devices
- Apply Changes to push code and resource changes to the running app without restarting the app
- A flexible Gradle-based build system
- A fast and feature-rich emulator
- GitHub and Code template integration to assist you in developing common app features and importing sample code
- Extensive testing tools and frameworks
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine, and many more.
- Provides GUI tools that simplify the less interesting parts of app development.
- Easy integration with real-time database ' firebase '.

Android Studio System Requirements for Windows

- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- 4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
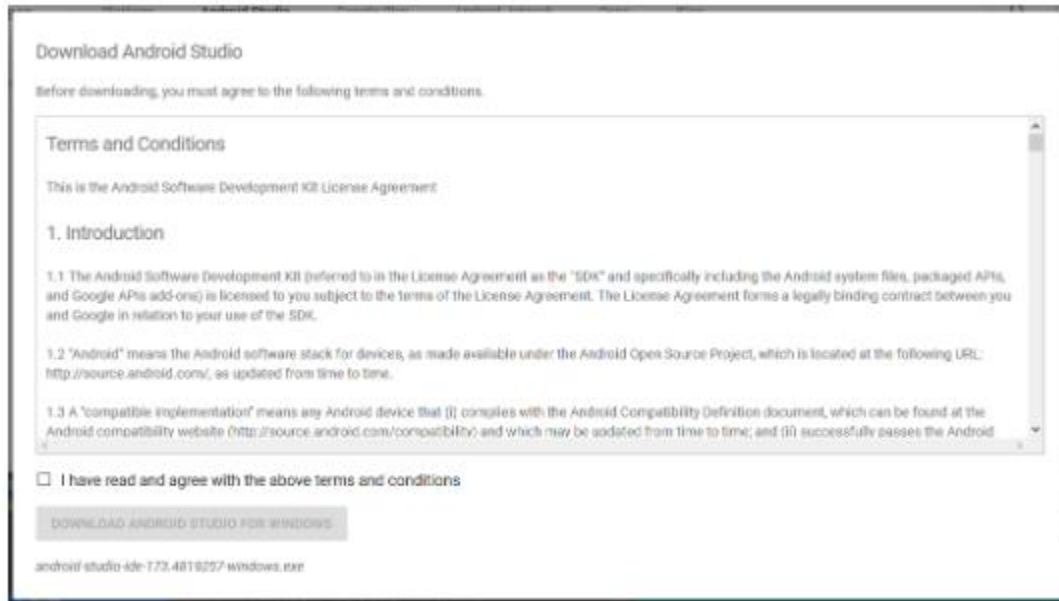- 1280 x 800 minimum screen resolution

Steps to Install Android Studio on Windows

Step 1: Head over to this link to get the Android Studio executable or zip file.

Step 2: Click on the Download Android Studio Button.

Click on the "I have read and agree with the above terms and conditions" checkbox followed by the download button.


Click on the Save file button in the appeared prompt box and the file will start downloading.

Step 3: After the downloading has finished, open the file from downloads and run it. It will prompt the following dialog box.

Click on next. In the next prompt, it'll ask for a path for installation. Choose a path and hit next.

Step 4: It will start the installation, and once it is completed, it will be like the image shown below.
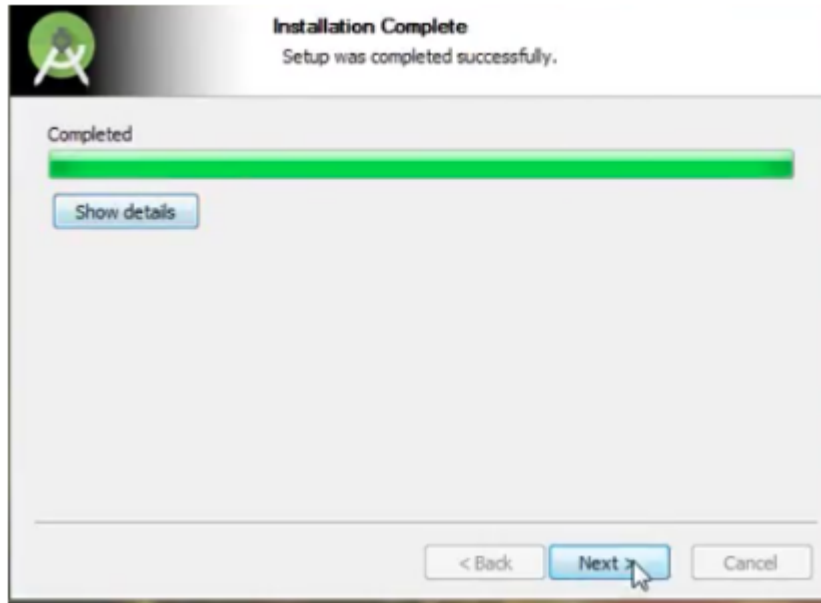
Click on next.

Step 5: Once " Finish " is clicked, it will ask whether the previous settings need to be imported [if the android studio had been installed earlier], or not. It is better to choose the 'Don't import Settings option'.
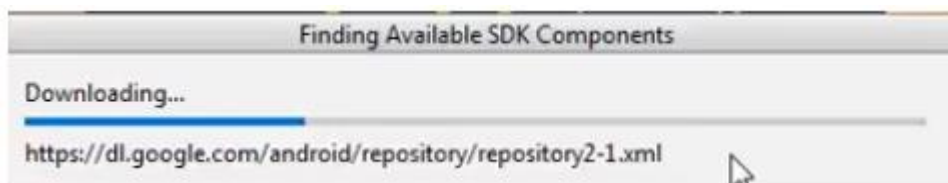
Click the **OK** button.

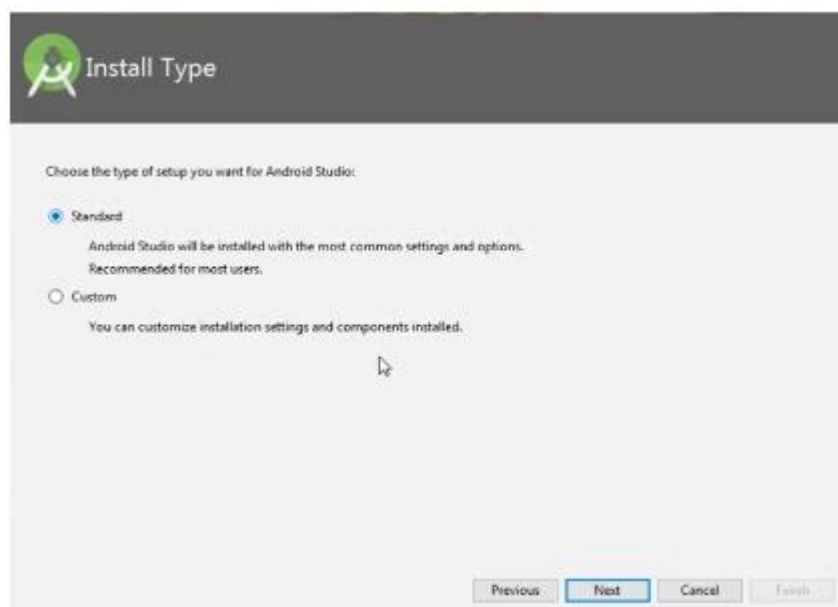**Step 6:** This will start the Android Studio.

Meanwhile, it will be finding the available SDK components.

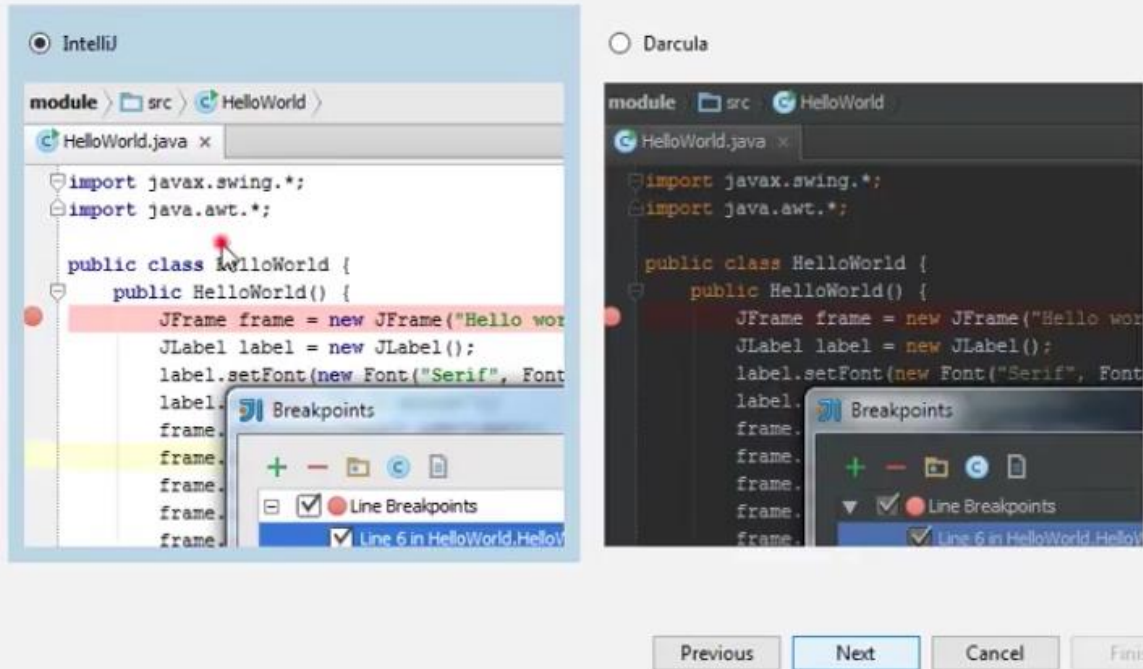Step 7: After it has found the SDK components, it will redirect to the Welcome dialog box.

Click on **Next** .

Choose Standard and click on Next. Now choose the theme, whether the Light theme or the Dark one. The light one is called the IntelliJ theme whereas the dark theme is called Dracula . Choose as required.
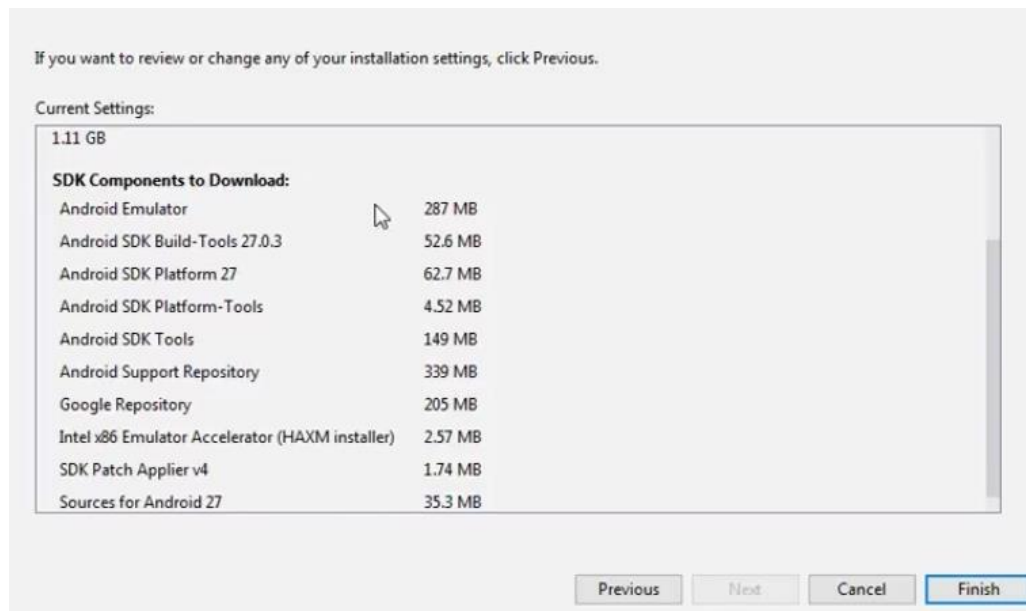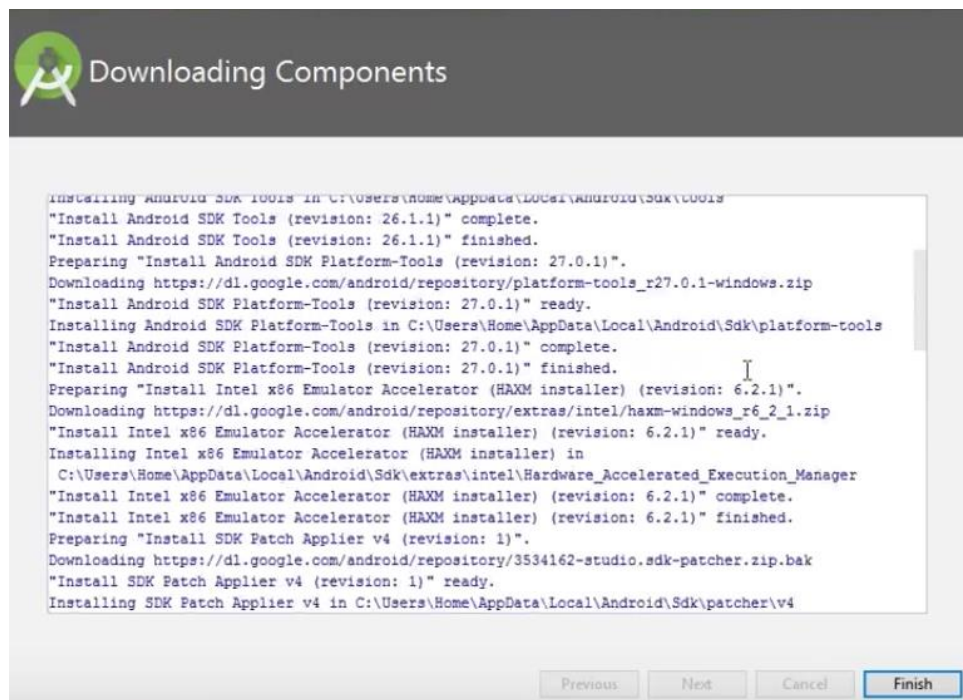
Click on the **Next** button.

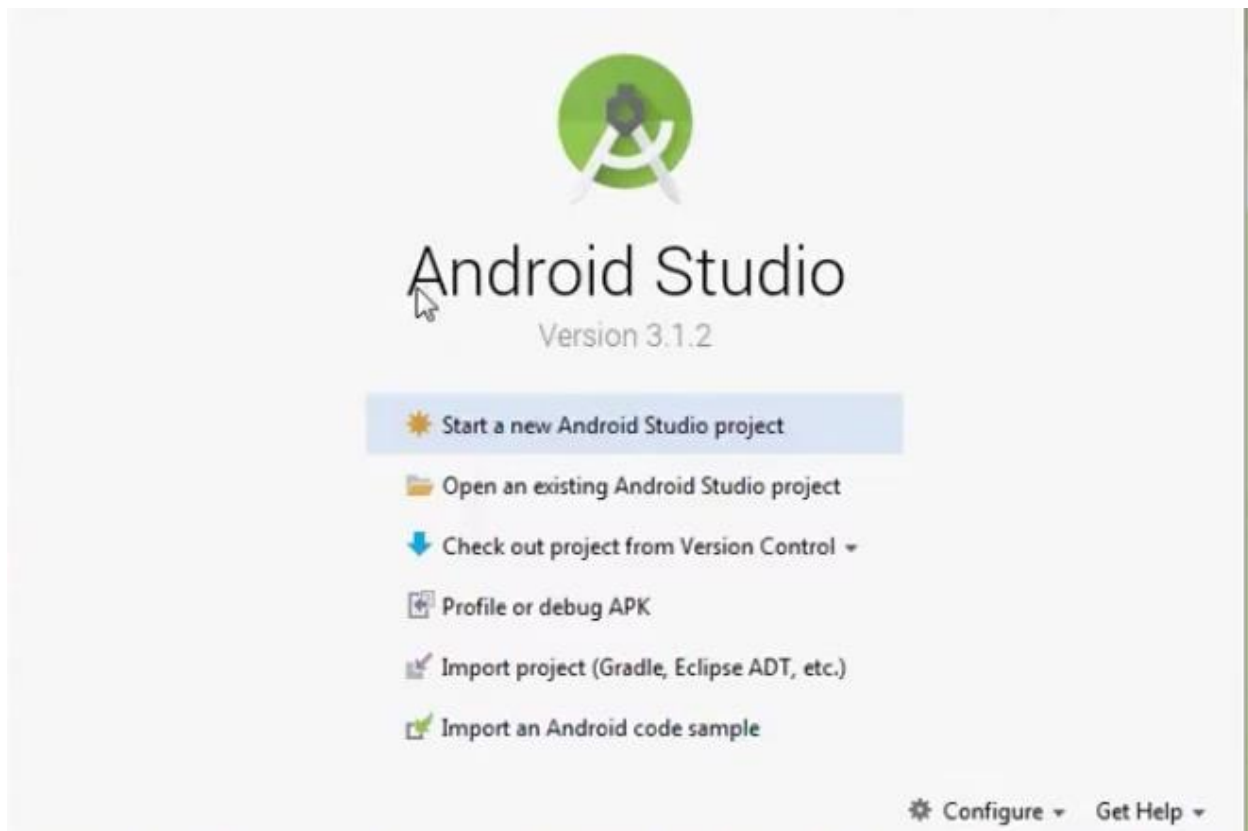**Step 8:** Now it is time to download the SDK components.

Click on Finish. Components begin to download let it complete.

The Android Studio has been successfully configured. Now it's time to launch and build apps. Click on the Finish button to launch it.

Step 9: Click on Start a new Android Studio project to build a new app.

Introduction to Android Studio Environment:

Android Application Package (APK):

An APK is an acronym for "Android Application Package." It is a file format used by the Android operating system to distribute and install applications. An APK file contains all the necessary files and code required to run a single Android application on an Android device.

APK files are typically downloaded and installed from Google Play Store or other third-party app stores, but they can also be downloaded directly from websites or shared through other means such as email or file sharing services.

When an APK file is installed on an Android device, it is first checked for malware and compatibility with the device's hardware and software. If the APK file is deemed safe and compatible, the Android system extracts the files and installs the application on the device.

APK files can be useful for developers who want to distribute beta versions of their apps to a select group of users, or for users who want to install apps from third-party sources that are not available on the Google Play Store. However, it is important to note that installing apps from outside the Google Play Store can increase the risk of malware or other security issues.

Android Virtual Device (AVD):

An Android Virtual Device (AVD) is a software emulation of an Android device that runs on a computer. It is used by developers to test and debug Android apps without the need for a physical Android device. An AVD allows developers to simulate different device configurations and test their apps in various environments.

To create an AVD, developers use the Android Virtual Device Manager, which is part of the Android SDK. The manager allows developers to specify the device type, screen resolution, operating system version, and other hardware and software characteristics.

Once an AVD is created, developers can launch it using an emulator provided by the Android SDK. The emulator simulates the device hardware and software environment and allows developers to test their apps as if they were running on an actual device.

AVDs can be useful for a variety of purposes, such as testing app compatibility with different versions of Android, verifying that apps behave correctly in various device configurations, and debugging issues that only occur on specific devices.

One of the advantages of using an AVD for app development is that it allows developers to test their apps on a wide range of devices without the need to purchase each device. This can save time and money, especially for small developers or startups.

Overall, an Android Virtual Device is an essential tool for developers looking to build high-quality, compatible, and robust Android apps. It enables them to test their apps in various environments and ensure that they run correctly on a wide range of devices.

Android User Interface (UI) layout:

Android user interface (UI) layout refers to the visual arrangement of the user interface components of an Android app, such as buttons, text fields, and images. It determines how these components are positioned and how they interact with each other to provide a coherent and user-friendly interface.

The Android UI layout is designed using XML files, which describe the UI components and their relationships to each other. The XML files are stored in the app's resources and can be modified using a layout editor or a text editor.

The Android UI layout consists of several key elements, including views, layouts, and widgets. Views are the basic building blocks of the UI, such as text views, image views, and buttons. Layouts are containers that hold views, such as linear layout, relative layout, and constraint layout. Widgets are interactive UI elements, such as checkboxes, radio buttons, and spinners.

Android provides a range of UI layout options that developers can choose from depending on the specific needs of their app. For example, a linear layout arranges views in a single column or row, while a grid layout arranges views in a grid-like pattern. A relative layout positions views relative to each other, while a constraint layout enables developers to create complex and flexible UI designs.

One of the key advantages of the Android UI layout is its flexibility and customization options. Developers can modify the layout's appearance, behavior, and interaction to meet the specific needs of their app and target audience. They can also use third-party libraries and tools to enhance the UI layout and improve the overall user experience.

In summary, the Android UI layout is a crucial aspect of app development that determines how users interact with an app's components. It enables developers to create visually appealing, user-friendly, and responsive interfaces that enhance the app's usability and functionality.

Extensible Markup Language (XML):

XML (Extensible Markup Language) is a markup language that is widely used in Android app development. It is a versatile format that is used to define the structure and content of user interface layouts, resources, and data storage in Android apps. Here are some of the main ways XML is used in Android:

User Interface Layouts:

In Android, user interface layouts are defined using XML files. The XML layout files describe the structure and properties of UI elements such as text views, buttons, images, and input fields. Developers can specify the size, position, alignment, and other properties of these elements using XML attributes.

Resources:

In Android, resources such as images, strings, colors, and dimensions are defined in XML files. These files can be stored in separate directories for different languages or screen sizes. The resources defined in XML files can be accessed programmatically using Java code.

Data Storage:

XML is often used for data storage in Android apps. Developers can use XML files to store app data such as preferences, settings, and other user-specific information. XML files can also be used to define data structures such as lists and arrays.

Manifest:

The Android Manifest is an XML file that contains important information about an Android app. It specifies the app's package name, version, permissions, activities, services, and other metadata.

Overall, XML is a fundamental part of Android app development. It is a flexible and powerful tool that enables developers to define app resources, user interfaces, and data storage structures

in a structured and organized way. Understanding XML is essential for building high-quality, user-friendly, and efficient Android apps.

Gradle Build System:

Gradle is a build automation tool known for its flexibility to build software. A build automation tool is used to automate the creation of applications. The building process includes compiling, linking, and packaging the code. The process becomes more consistent with the help of build automation tools.

Comparison and limitations of popular Cross platform development tools

There are several popular cross-platform development tools available for building mobile applications, including:

React Native:

React Native is an open-source framework for building mobile applications using JavaScript and the React library. It allows developers to write code once and run it on both iOS and Android platforms. React Native has a large developer community and a rich set of third-party libraries and plugins, making it a popular choice for many developers.

Limitations:

React Native still has some limitations with regards to its performance compared to native apps, especially for more demanding applications.

Access to native APIs can be more limited and may require additional development effort.

Xamarin:

Xamarin is a cross-platform development tool that uses C# and the .NET framework to build native mobile applications for iOS, Android, and Windows. It allows developers to share code across platforms, reducing the amount of time and effort required to develop and maintain multiple versions of the same application.

Limitations:

Xamarin has a steeper learning curve compared to some of the other cross-platform tools, especially for developers who are not familiar with C# and the .NET framework.

The size of the Xamarin app can be larger compared to native apps due to the inclusion of the .NET runtime.

Flutter:

Flutter is a relatively new cross-platform development tool that uses the Dart programming language to build high-performance, visually appealing mobile applications for iOS and Android. It has a fast development cycle and a rich set of built-in widgets and tools, making it a popular choice for developers who want to create attractive user interfaces.

Limitations:

Flutter is still a relatively new technology and has a smaller developer community compared to more established cross-platform tools.

Access to platform-specific features and APIs can be more limited compared to native development.

Ionic: Ionic is a popular open-source framework for building hybrid mobile applications using HTML, CSS, and JavaScript. It provides a set of UI components and tools for building attractive and functional applications that run on both iOS and Android.

Limitations:

Ionic applications can have slower performance compared to native apps, especially for more demanding applications.

The user interface may not be as seamless and native-like compared to other cross-platform tools.

Each cross-platform development tool has its own strengths and limitations, and the best choice for a particular project will depend on the specific requirements and goals of the development team. It's important to carefully consider the trade-offs and limitations of each tool before making a decision on which one to use.
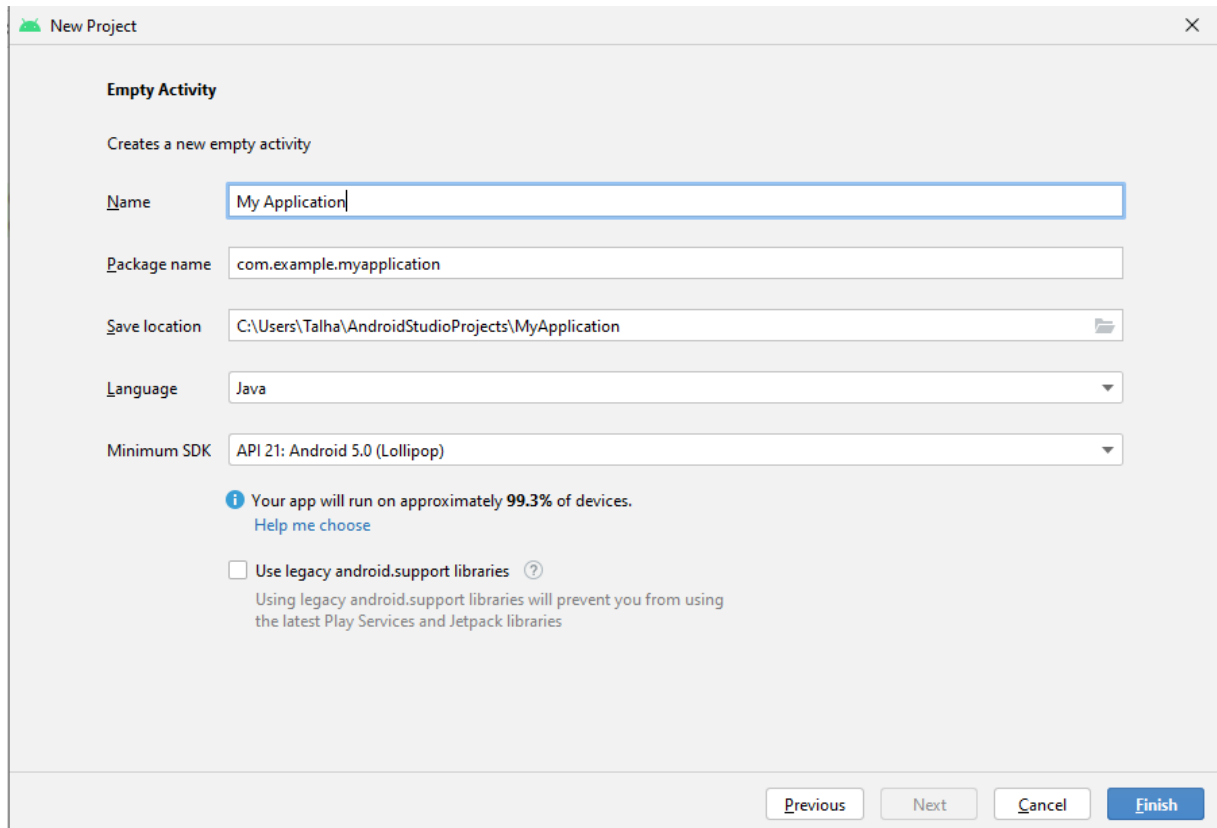
First Application:

How to create an App in android studio.

Case Study: KG to Pounds Converter App

Make an android application that can be used as converter app. The app will take input from user in Kilograms and convert that input into pounds.

Open Android Studio. Create a new Android Studio project with an empty activity.

Then, in the design view add two UI elements: an **EditText** for the user to enter a value in kilograms, and a **Button** to initiate the conversion. Finally, add a **TextView** to display the converted value in pounds. An XML file containing tags for all of the above will be created. you can adjust setting using XML code or the design view.

public class MainActivity extends AppCompatActivity {

Now code here.

the usual way to write code is data_type VariableName, variableName2;

    EditText inputKg;   //data_type var_name;

    Button convertButton;   //data_type variable_name;

TextView resultText;

```
setContentView(R.layout.activity_main);
```

Now for each variable you created above we need to set the id of each UI element to linkup the above variables with their respective UI element.

inputKg = findViewById(R.id.input_kg);

convertButton = findViewById(R.id.convert_button);

resultText = findViewById(R.id.result_text);

// variable_name = findViewById(R.id.id_of_UI_element)

//implementation of button Convert

convertButton.setOnClickListener(new View.OnClickListener() {

float kg = Float.parseFloat(inputKg.getText().toString());

float pounds = kg * 2.20462;

resultText.setText(String.format("%.2f pounds", pounds));

In this code, we first declare three variables to represent the UI elements: an EditText for input, a Button for conversion, and a TextView for displaying the result.

Then, in the onCreate() method, we initialize these variables by finding them in the activity's layout using their IDs. We also set an OnClickListener on the convertButton to trigger the conversion when the user taps the button.

Inside the OnClickListener, we first retrieve the value entered by the user in the inputKg EditText field and convert it to a double. We then perform the conversion to pounds by multiplying the kilograms value by the conversion factor (2.20462). Finally, we format the result as a string with two decimal places and set it as the text of the resultText TextView.

Understanding MainActivity.java

AppCompatActivity is a class provided by the Android Support Library (now AndroidX) that serves as a base class for activities that use the ActionBar feature. It extends the FragmentActivity class and adds support for the ActionBar, which is a bar that appears at the top of the activity window and provides navigation and other options.

AppCompatActivity provides various methods to work with the ActionBar, such as getSupportActionBar() to retrieve the ActionBar instance and setSupportActionBar() to set the ActionBar for the activity. It also provides support for other features such as displaying fragments, handling configuration changes, and managing the activity lifecycle. AppCompatActivity is a useful class for developers who want to create activities with ActionBar support and take advantage of the many other features provided by the Android Support Library.

setOnClickListener() is a method in Android programming that is used to add a listener to a button or other interactive view element. When the view is clicked, the listener's onClick() method is executed, which contains the code to be executed when the button is clicked. setOnClickListener() is a very useful method in Android programming as it allows developers to add interactivity to their apps by responding to user input.

Calculator App:

Step 1: Open the MainActivity.java file located in the java/com.example.yourappname directory.

Step 2: Define the functionality of the calculator in Java. We'll handle button clicks for numbers, operators, equals, and clear operations.

Here's the complete Java code for MainActivity.java:

```java
// MainActivity.java

package com.example.yourappname;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    // Define UI elements
    EditText resultEditText;
    private double operand1 = 0;
    private String operator = "";

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

```
    // Initialize UI elements
    resultEditText = findViewById(R.id.resultEditText);
}


// Method to handle number button clicks
public void onNumberClick(View view) {
    Button button = (Button) view;
    String buttonText = button.getText().toString();
    String currentText = resultEditText.getText().toString();


    // Append the clicked number to the EditText
    if (currentText.equals("0")) {
        resultEditText.setText(buttonText);
    } else {
        resultEditText.setText(currentText + buttonText);
    }
}


// Method to handle operator button clicks
public void onOperatorClick(View view) {
    Button button = (Button) view;
    operator = button.getText().toString();
    operand1 = Double.parseDouble(resultEditText.getText().toString());
    resultEditText.setText("");
}


// Method to handle equals button click
```

```java
public void onEqualClick(View view) {
    double operand2 = Double.parseDouble(resultEditText.getText().toString());
    double result = 0;

    // Perform the operation based on the operator
    switch (operator) {
        case "+":
            result = operand1 + operand2;
            break;
        case "-":
            result = operand1 - operand2;
            break;
        case "x":
            result = operand1 * operand2;
            break;
        case "/":
            if (operand2 != 0) {
                result = operand1 / operand2;
            } else {
                resultEditText.setText("Error");
                return;
            }
            break;
    }

    // Display the result
    resultEditText.setText(String.valueOf(result));
}
```

```java
    // Method to handle clear button click

    public void onClearClick(View view) {

        resultEditText.setText("0");

        operand1 = 0;

        operator = "";

    }

}
```

Explanation:

We import necessary classes including AppCompatActivity, Bundle, View, Button, and EditText.

We define a class MainActivity which extends AppCompatActivity.

We declare private variables resultEditText to reference the EditText where the input and result are displayed, operand1 to store the first operand, and operator to store the operator.

In onCreate() method, we initialize the UI elements by finding them using their respective IDs.

We define four methods onNumberClick(), onOperatorClick(), onEqualClick(), and onClearClick() to handle button clicks for numbers, operators, equals, and clear operations respectively.

In onNumberClick() method, we append the clicked number to the EditText.

In onOperatorClick() method, we store the operator clicked and the current value in the EditText as the first operand.

In onEqualClick() method, we perform the operation based on the stored operator and display the result in the EditText.

In onClearClick() method, we reset the EditText and clear the stored operands and operator.

onCreate(Bundle savedInstanceState):

This method belongs to the AppCompatActivity class.

It is called when the activity is first created.

In this method, we initialize the activity, set the content view using setContentView(R.layout.activity_main), and initialize the resultEditText variable.

onNumberClick(View view):

This method is called when a number button (0-9) is clicked.

It retrieves the text from the clicked button and updates the text displayed in the resultEditText.

onOperatorClick(View view):

This method is called when an operator button (+, -, *, /) is clicked.

It retrieves the text from the clicked button, stores it as the current operator, and parses the current value displayed in the resultEditText as the first operand.

onEqualClick(View view):

This method is called when the equal button (=) is clicked.

It parses the current value displayed in the resultEditText as the second operand.

Based on the current operator, it performs the corresponding arithmetic operation (addition, subtraction, multiplication, or division) using a switch-case statement.

The result is displayed in the resultEditText.

onClearClick(View view):

This method is called when the clear button (C) is clicked.

It resets the resultEditText to display "0" and resets the operand1 and operator variables.

Tip Calculator App:

Develop an app that calculates the tip amount based on the bill total and percentage of tip selected by the user.

```
<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"
    android:padding="16dp"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextBillAmount"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Bill Amount"
        android:inputType="numberDecimal" />

    <EditText
        android:id="@+id/editTextTipPercentage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/editTextBillAmount"
        android:layout_marginTop="16dp"
        android:hint="Enter Tip Percentage"
        android:inputType="number" />

    <Button
        android:id="@+id/buttonCalculate"
```

```xml
        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:layout_below="@id/editTextTipPercentage"

        android:layout_marginTop="16dp"

        android:text="Calculate Tip" />


    <TextView

        android:id="@+id/textViewTipAmount"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_below="@id/buttonCalculate"

        android:layout_marginTop="16dp"

        android:text="Tip Amount: $0.00"

        android:textSize="20sp" />


</RelativeLayout>
```

JAVA CODE:

```java
import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;


import androidx.appcompat.app.AppCompatActivity;


import java.text.DecimalFormat;
```

```java
public class MainActivity extends AppCompatActivity {

    private EditText editTextBillAmount, editTextTipPercentage;
    private Button buttonCalculate;
    private TextView textViewTipAmount;

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

        editTextBillAmount = findViewById(R.id.editTextBillAmount);
        editTextTipPercentage = findViewById(R.id.editTextTipPercentage);
        buttonCalculate = findViewById(R.id.buttonCalculate);
        textViewTipAmount = findViewById(R.id.textViewTipAmount);

        buttonCalculate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                calculateTip();
            }
        });
    }

    private void calculateTip() {
        // Get bill amount from EditText and convert it to a double
        double billAmount = Double.parseDouble(editTextBillAmount.getText().toString());
```

```
        // Get tip percentage from EditText and convert it to a double

        double tipPercentage = Double.parseDouble(editTextTipPercentage.getText().toString());


        // Calculate tip amount

        double tipAmount = (billAmount * tipPercentage) / 100;


        // Format the tip amount to display with 2 decimal places

        DecimalFormat decimalFormat = new DecimalFormat("#.00");

        String formattedTipAmount = decimalFormat.format(tipAmount);


        // Display the tip amount in the TextView

        textViewTipAmount.setText("Tip Amount: $" + formattedTipAmount);

    }

}
```

parseDouble(String): Converts a string to a double
DecimalFormat.format(double):Formats the number to two decimal places
setOnClickListener(...): Triggers action when button is clicked

CODE EXPLANATION:

parseDouble(String s):

- This method is part of the Double class.
- It converts the string representation of a floating-point number into a double.
- In this code, it's used to convert the user input from the EditText fields (bill amount and tip percentage) into double values for calculation.

format(String pattern):

- This method is part of the **DecimalFormat** class.
- This class allows you to control the display of leading and trailing zeros, prefixes and suffixes, grouping (thousand) separators, and the decimal separators.
- It formats a given number according to the specified pattern.
- In this code, it's used to format the calculated tip amount with two decimal places before displaying it in the TextView.

setOnClickListener(View.OnClickListener):

- This method is called on the Button (buttonCalculate) to set an event listener for handling click events.

- It takes an instance of View.OnClickListener as a parameter, which defines the action to be performed when the button is clicked.
- In this case, it's an anonymous inner class implementing View.OnClickListener, where the onClick(View v) method is overridden to call the calculateTip() method.

Splash Screen:

- Starting in Android 12, the SplashScreen API enables a new app launch animation for all apps when running on a device with Android 12 or higher.
- This includes an into-app motion at launch, a splash screen showing your app icon, and a transition to your app itself.
- A SplashScreen is a Window and therefore occludes an Activity.
- This experience brings standard design elements to every app launch, but it's also customizable so your app can maintain its unique branding.
- To understand why splash screens are essential, you need to understand the various app startup states.

App Startup States

Users launch Apps in many ways; first-time open from a fresh install, quickly switching between Recents or from the main launcher for the first time in days. Your app needs to handle all the above scenarios. Your app will launch from one of the following states:

- Cold start
- Warm start
- Hot start

How the Splash Screen Works:

When a user launches an app while the app's process is not running (a cold start) or the Activity has not been created (a warm start), the following events occur. (The splash screen is never shown during a hot start.) The system shows the splash screen using themes and any animations that you've defined. When the app is ready, the splash screen is dismissed and the app is displayed. The splash screen can be customized, allowing you to supply your own logo animation and branding.

To insert and use ImageView

- Download a background image and calculator png image from google.
- Open the image and copy it using ctrl + c (or right click the opened image and click on copy)
- Paste the image in android studio.
- To paste:
- On the left most pane of your android studio project, select the first tab i.e. Project
- Expand the folder res, then expand the folder Drawable. The folder already contains two image files.
- Paste the copied image here, on pasting a dialog box will appear write the name of the image (must not contain spaces in name, must not contain any capital letter in the name, only SMALL letters)
- To use that image, go to palette, click on Widgets, drop ImageView to the design view.
- A dialog box appear, choose the name of the image you want to choose and press ok.

To add Splash Screen (XML)

- Create a new empty activity. To create:
  - On the left most pane open the tab Project
  - Expand the folder Java
  - Right click on the package name, select New, the select Activity, then select Empty Activity
  - Give the name of the activity e.g. SplashScreen
  - A new java and xml file appears with the given name.
- Set the background of your splash screen
- Drop an ImageView and choose the logo you wish to appear on your splash screen.
- Set the constraints of your image dropped.

Create New Activity in the project
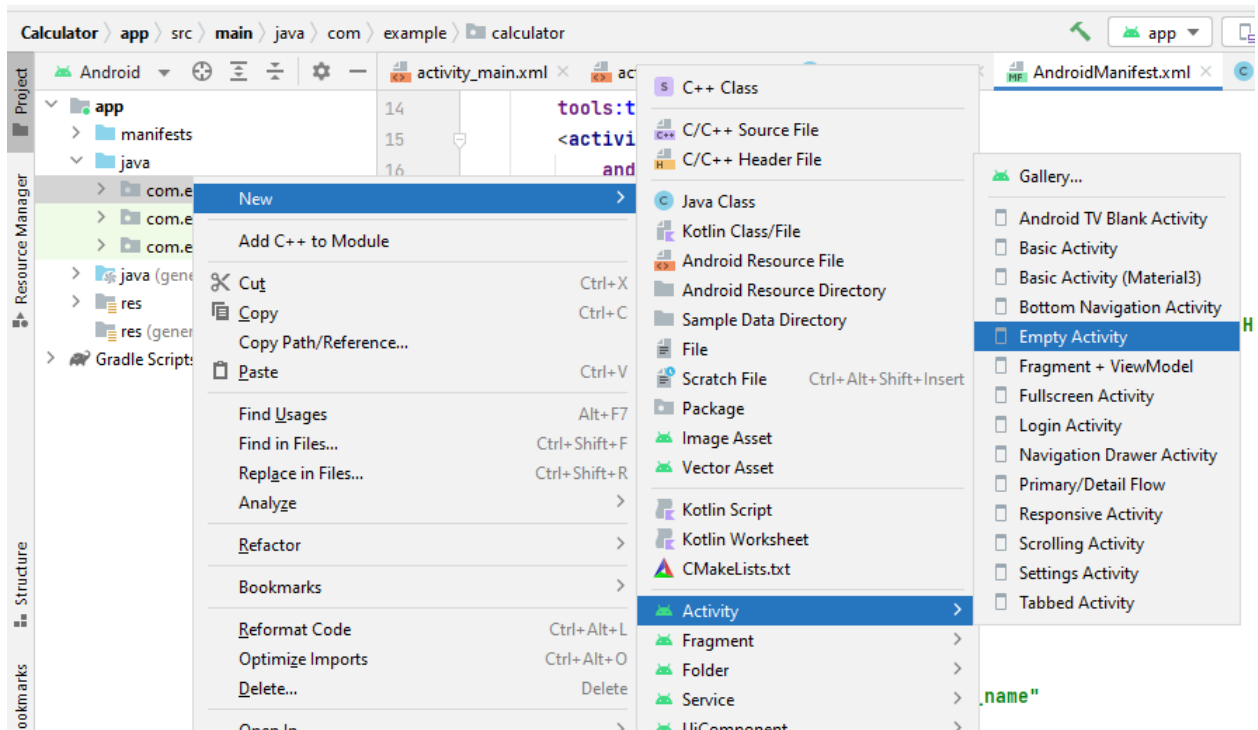
To add Splash Screen (Java)

- Open the Java file of your splash screen activity.
- Under the main class activity class (i.e. "public class MainActivity2 extends AppCompatActivity {") make an object of Hnadler class.
  - Handler h = new Handler(); // here the object name is h and Handler is the class name.
  - Syntax to create object of class in java is as follows:
  - ClassName objectName = new ClassName();
- Now go to onCreate Method, under the R.layout method
  - Use postDelayed method, this method is used to delay the execution time of your code. Write new Runnable and hit enter it will do rest of code on its own.
  - Right after where the method runnable ends },5000)
  - It means you want to delay the code for 5 seconds (5000 miliseconds)
  - Now create an object of Intent class under the public void run()
  - Intent I = New Intent(SplashScreen.this , MainActivity.class)
  - Here we pass two parameter one for this current activity and one for the activity where we want to go
  - Now in the next line simply call the startActivity(i) and pass the Intent object.
  - Now go to the next line and just add the finish(); function
  - Here is the code:

```java
public class MainActivity2 extends AppCompatActivity {

    Handler h = new Handler();                          //create object of Handler class

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main2);

        h.postDelayed(new Runnable() {     //use the postDelayed method with Handler class object

            @Override

            public void run() {

                Intent i = new Intent(MainActivity2.this, MainActivity.class); //MainActivity2 is splash screen activity and
                                        //MainActivity is the calculator screen

                startActivity(i);               //startActivity function is called and intent class object is passed

                finish();               //finish function is used

            }

        }, 5000); //the delay time is given in miliseconds

    }
}
```

```
    activity_main.xml ×    activity_main2.xml ×    © MainActivity2.java ×    AndroidManifest.xml ×    © MainActivity.java ×

1       package com.example.calculator;
2
3       import ...
8
9       public class MainActivity2 extends AppCompatActivity {
10          Handler h = new Handler();
11
12          @Override
13          protected void onCreate(Bundle savedInstanceState) {
14              super.onCreate(savedInstanceState);
15              setContentView(R.layout.activity_main2);
16              h.postDelayed(new Runnable() {
17                  @Override
18                  public void run() {
19                      Intent i = new Intent( packageContext: MainActivity2.this, MainActivity.class);
20                      startActivity(i);
21                      finish();
22                  }
23              }, delayMillis: 5000);
24          }
25      }
```

Android Manifest XML file

As we created the calculator on MainActivity class and splash screen later on. We need to tell the android studio which file needs to be run first. To do so:

- Go to Project, expand the folder app, expand the folder manifests.
- Open AndroidManifest.xml file.
- Cut the intent filter tag(the complete tag from start till end) from MainActivity tag and paste the complete tag in SplashScreen Acitivity tag.
- Set the android:exported value as "true"
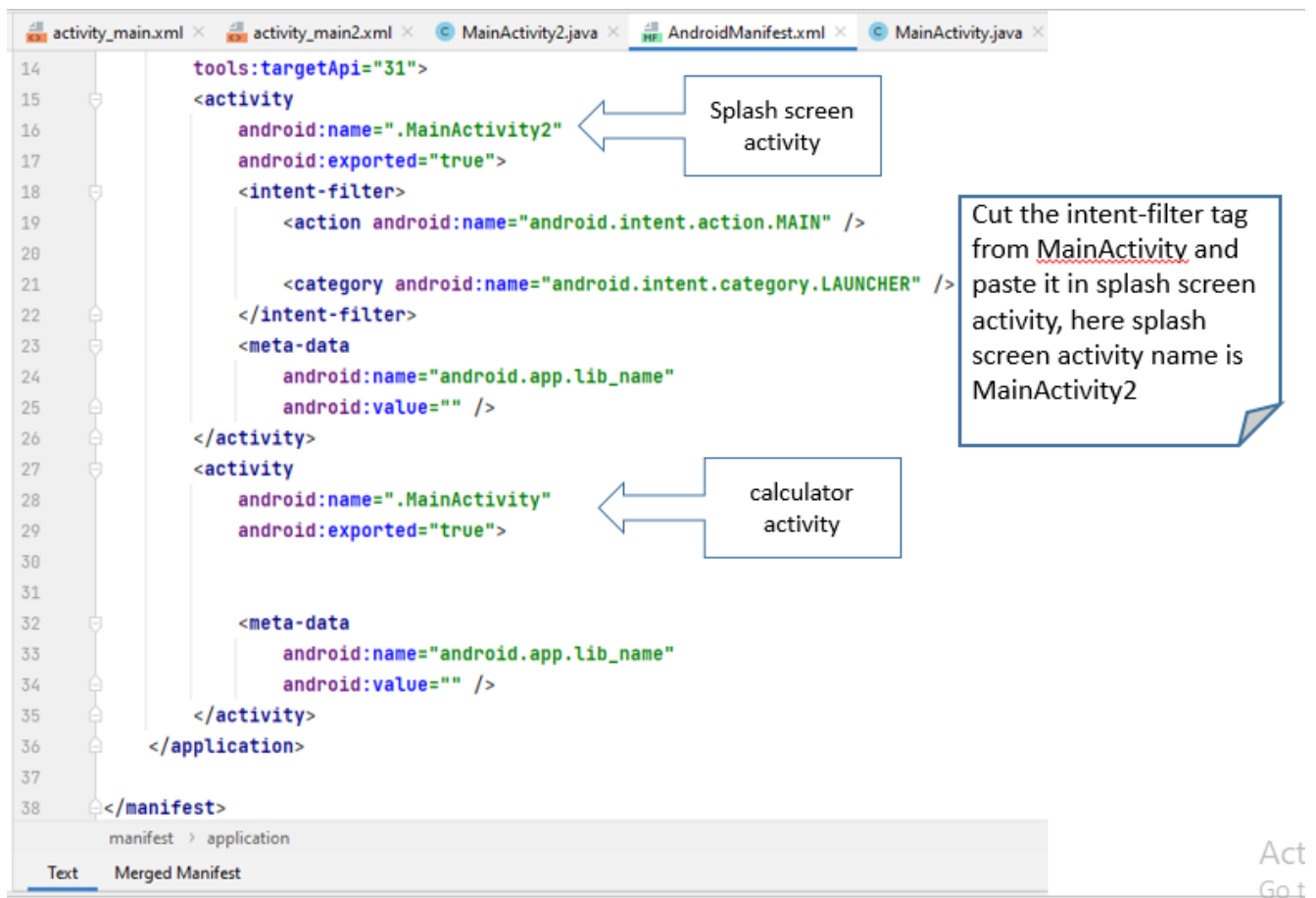
Explanation

Handler Class:

- In android Handler is mainly used to update the main thread from background thread or other than main thread.
- A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue.
- Each Handler instance is associated with a single thread and that thread's message queue. When you create a new Handler it is bound to a Looper.
- It will deliver messages and runnables to that Looper's message queue and execute them on that Looper's thread.
- Scheduling message is accomplished with post(Runnable)

Runnable:

- The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.
- The class must define a method of no arguments called run.

Looper:

- Class used to run a message loop for a thread.

Thread:

- A thread is a lightweight sub-process, it going to do background operations without interrupt to UI.
- When the user launches your app, Android creates a new Linux process along with an execution thread.
- This main thread, also known as the UI thread, is responsible for everything that happens onscreen. Understanding how it works can help you design your app to use the main thread for the best possible performance.
- To keep your application responsive, it is essential to avoid using the main thread to perform any operation that may end up keeping it blocked.

postDelayed:

- postDelayed is a method that causes the Runnable r to be added to the message queue, to be run after the specified amount of time elapses.

Intent:

Intent is to perform an action. It is mostly used to start activity, send broadcast receiver, start services and send message between two activities. There are two intents available in android as Implicit Intents and Explicit Intents.

Explicit Intent

- It connects the internal world of an application such as start activity or send data between two activities. To start new activity we have to create Intent object and pass source activity and destination activity.

Intent i = new Intent(MainActivity.this, SecondActivity.class);

startActivity(i);

Implicit Intents

- It connects our app with any other application on our device such as call, mail, phone, visit any website ..etc. In implicit intent we have to pass an action using setAction() as shown below example.

    Intent i = new Intent();

    i.setAction(Intent.ACTION_VIEW);

    i.setData(Uri.parse("www.uaf.edu.pk"));

    //uri is uniform resource identifier

```
startActivity(i);
```

Lottie Animation in Android Apps

- A Lottie is a JSON-based animation file format that allows you to ship animations on any platform as easily as shipping static assets.
- They are small files that work on any device and can scale up or down without pixelation.
- LottieFiles lets you create, edit, test, collaborate on and ship a Lottie in the easiest way possible.
- Lottie animations are much smaller while retaining the same quality compared to other formats like GIF or MP4.
- Lottie animations are based on vectors, which means you can scale them up and down without worrying about resolution.
- Multi-platform support and libraries

What is Library

- It is structurally the same as an Android app module.
- It can include everything needed to build an App.
- Instead of compiling into an APK, it complies into an Android Archive (AAR) file that is used as a dependency for an Android app module.
- Unlike JAR files, AAR files offer the following functionality for Android apps:
- AAR files can contain Android resources and a manifest file, which lets you bundle in shared resources like layouts and drawables in addition to Kotlin or Java classes and methods.
- AAR files can contain C/C++ libraries for use by the app module's C/C++ code.

Why Library

- A library module is useful in the following situations:
- When building multiple apps that use some of the same components, such as activities, services, or UI layouts
- When building an app that exists in multiple APK variations, such as a free and paid version, that share core components
- In either case, move the files you want to reuse into a library module and then add the library as a dependency for each app module.

How to get Lottie Files

https://lottiefiles.com/ use this official website

- You can create your account there
- Then sign in
- Simply search the animation from the above link
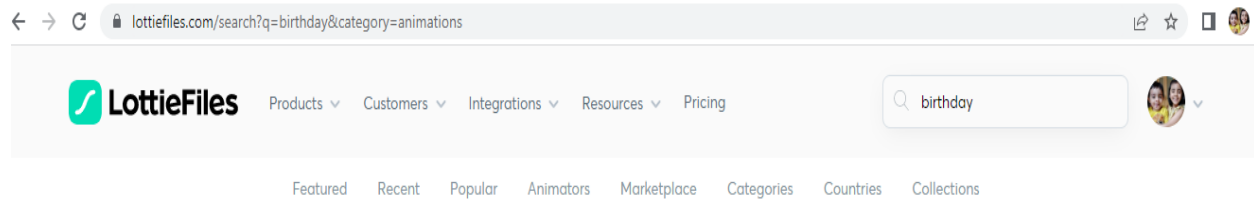
Download Lottie Animation

- Click on the animation you want to download, a new detailed window appears.
- You can adjust play speed and color combo of animation from this palette.
- To download click on download and download JSON file.
- You can change the name of lottie animation JSON file, the name should not start with numbers.
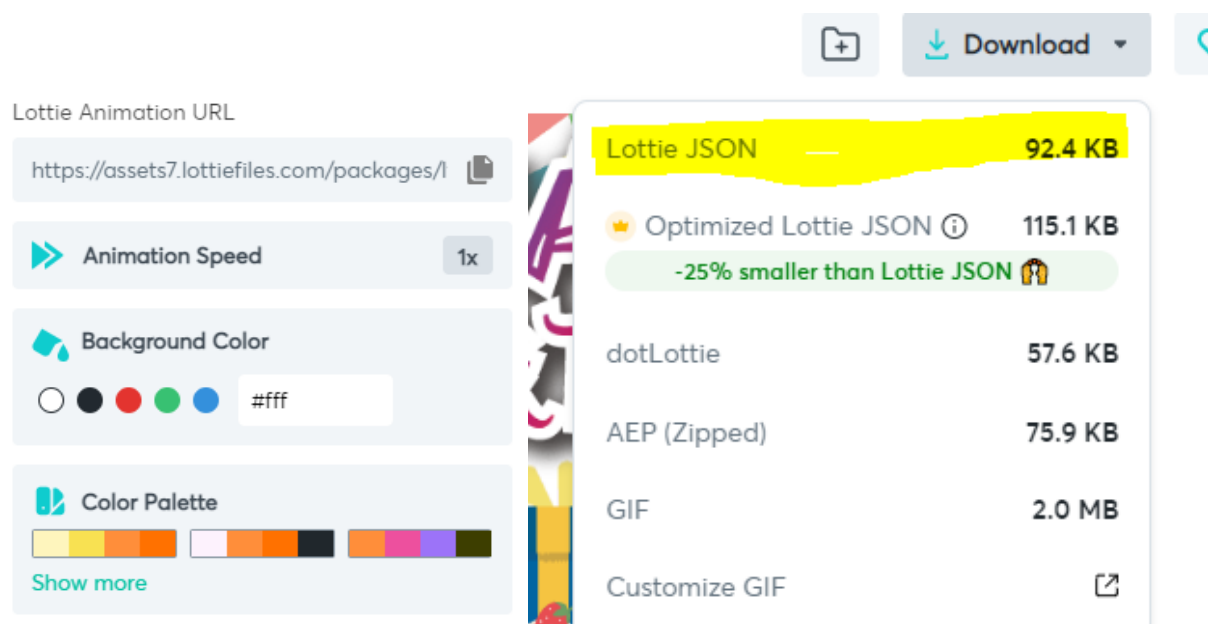
Create New Project to use lottie animation

- Create a new project
- Select Empty activity and press Finish.
- Now we have to paste that JSON file in our project. All the media will be pasted in "res" directory
- Now right click on the "res" directory, then click "New", then click on "Android Resource Directory"

- A new dialog box appears, give the directory name as "raw" (all small letters) and click "OK".
- A new folder named "raw" is created in "res" directory.
- Now paste the copied JSON file in raw folder and press ok. You can rename the file here as well.
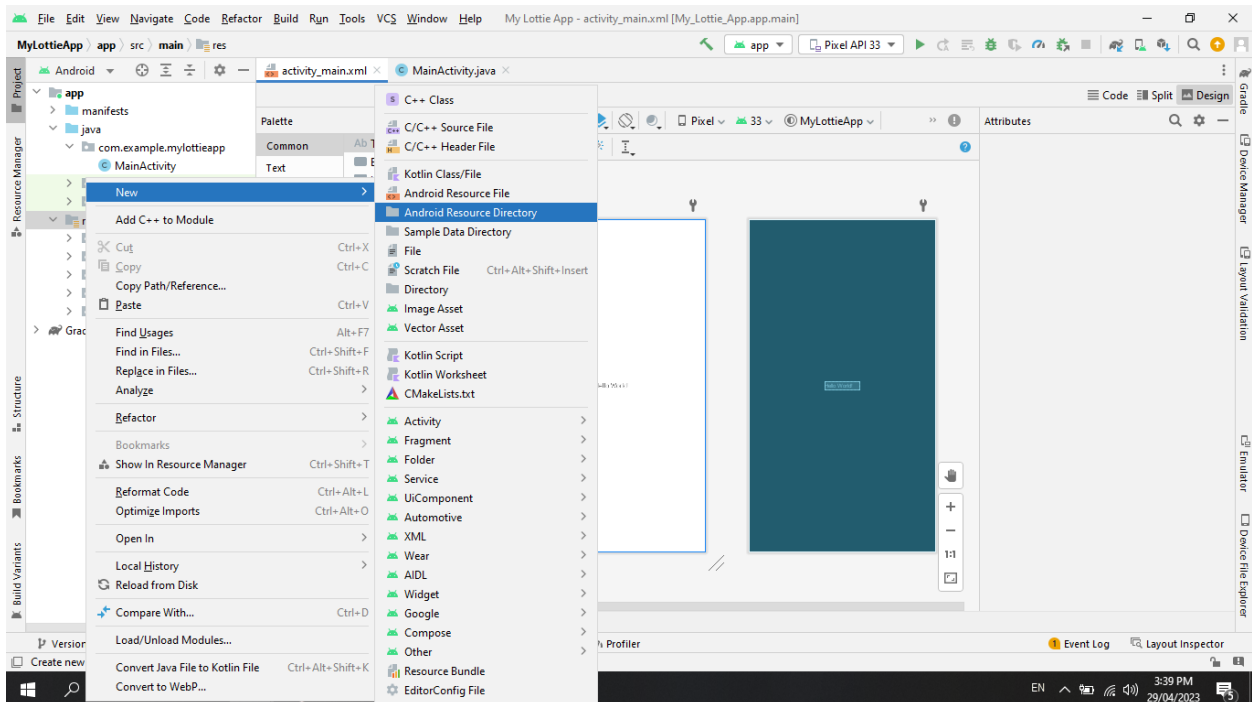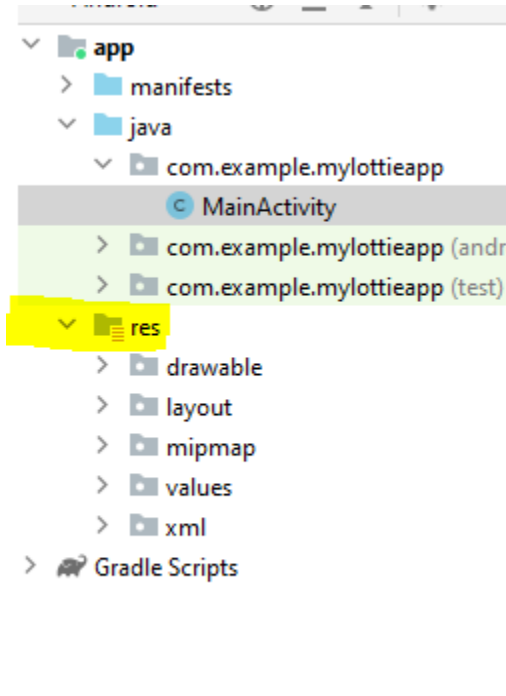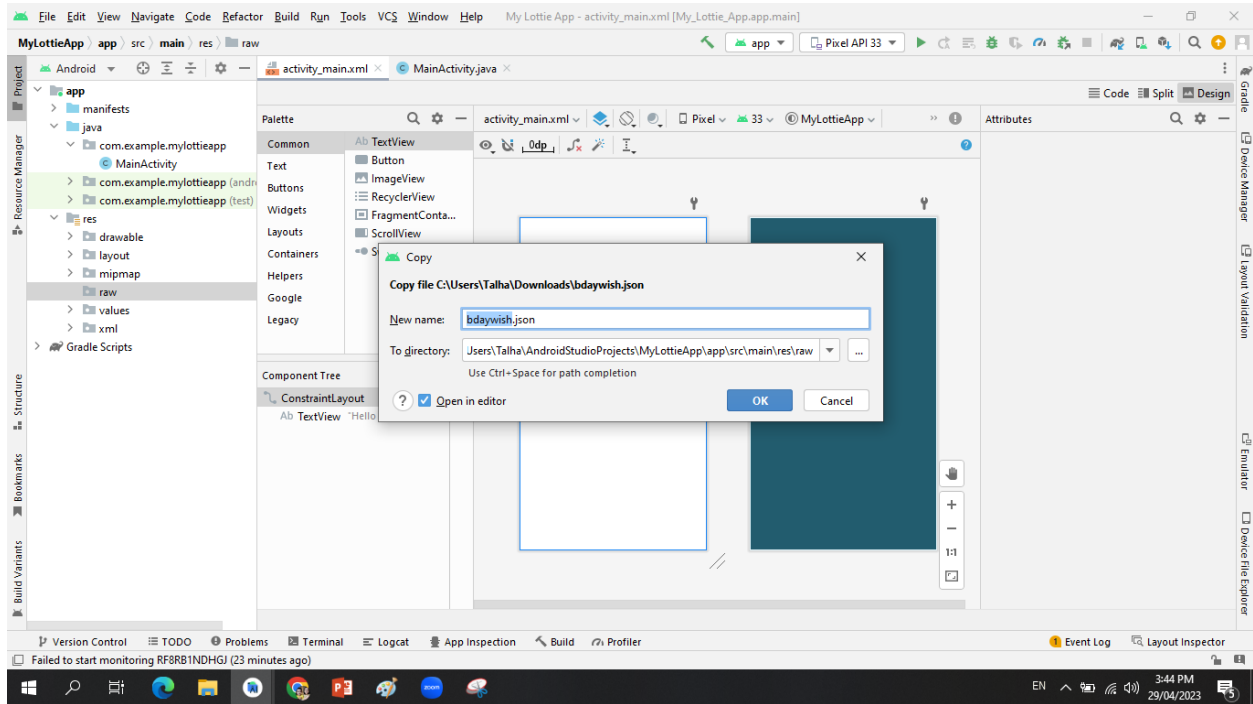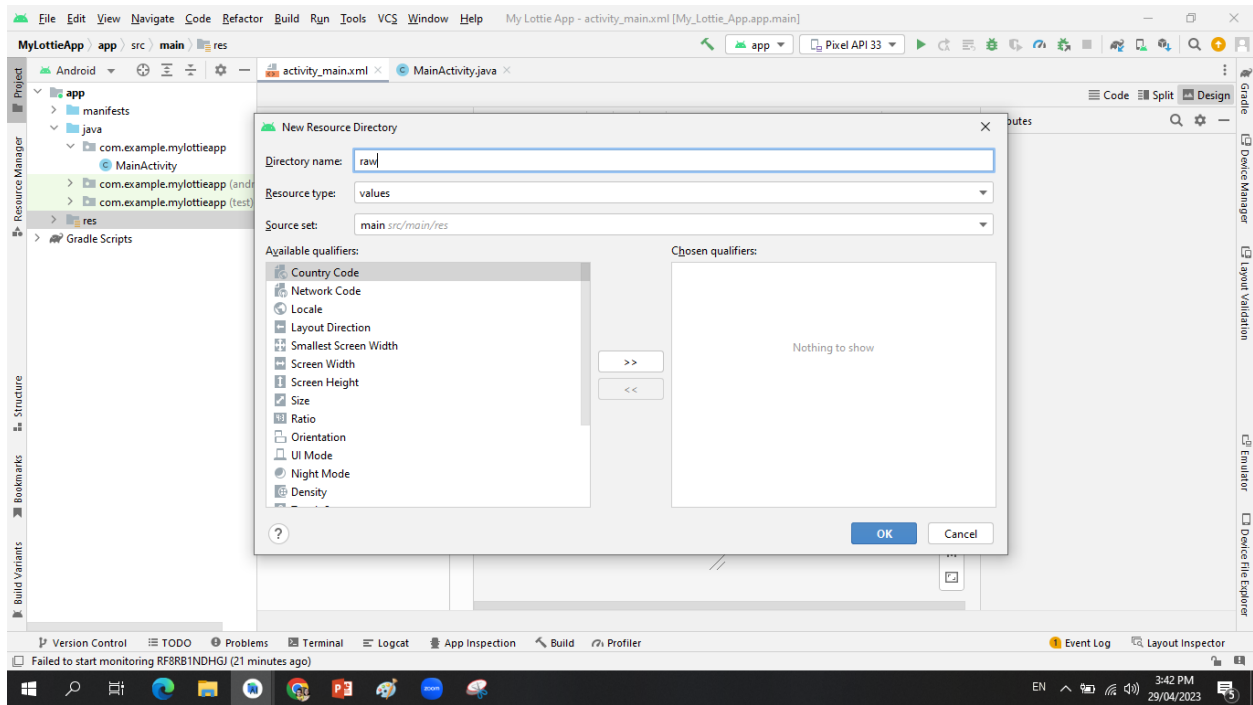
Library integration

- Now to use lottie animation in my android project. We have to integrate lottie animation project in our android project.

- This will give us lottie animation view which will convert downloaded JSON file as animation file to be used by our android project.
- Now open your browser and search "lottie animation github"
- Open the first link of githubt
- Scroll down to the following part: https://github.com/airbnb/lottie-android

Download

Gradle is the only supported build configuration, so just add the dependency to your project `build.gradle` file:

```
dependencies {
    implementation 'com.airbnb.android:lottie:$lottieVersion'
}
```

The latest Lottie version is: `maven central` `6.0.0`

The latest stable Lottie-Compose version is: `maven central` `6.0.0` Click here for more information on Lottie-Compose.

Lottie 2.8.0 and above only supports projects that have been migrated to androidx. For more information, read Google's migration guide.

- Now copy the dependencies code as it is:
- implementation 'com.airbnb.android:lottie:$lottieVersion'
- Now here we have to replace $lottieVersion with lottie version available on the site, currently 6.0.0 version is available.
- Now go to your project
- Expand "gradle" of left most panel of your project.
- Click on the build.gradle of your app (the first build.gradle file belongs to project you have to click on the second file that is for app)
- Now paste the above copied code in the dependencies. Replace the $lottieVersion with the version available on the site.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.airbnb.android:lottie:6.0.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```
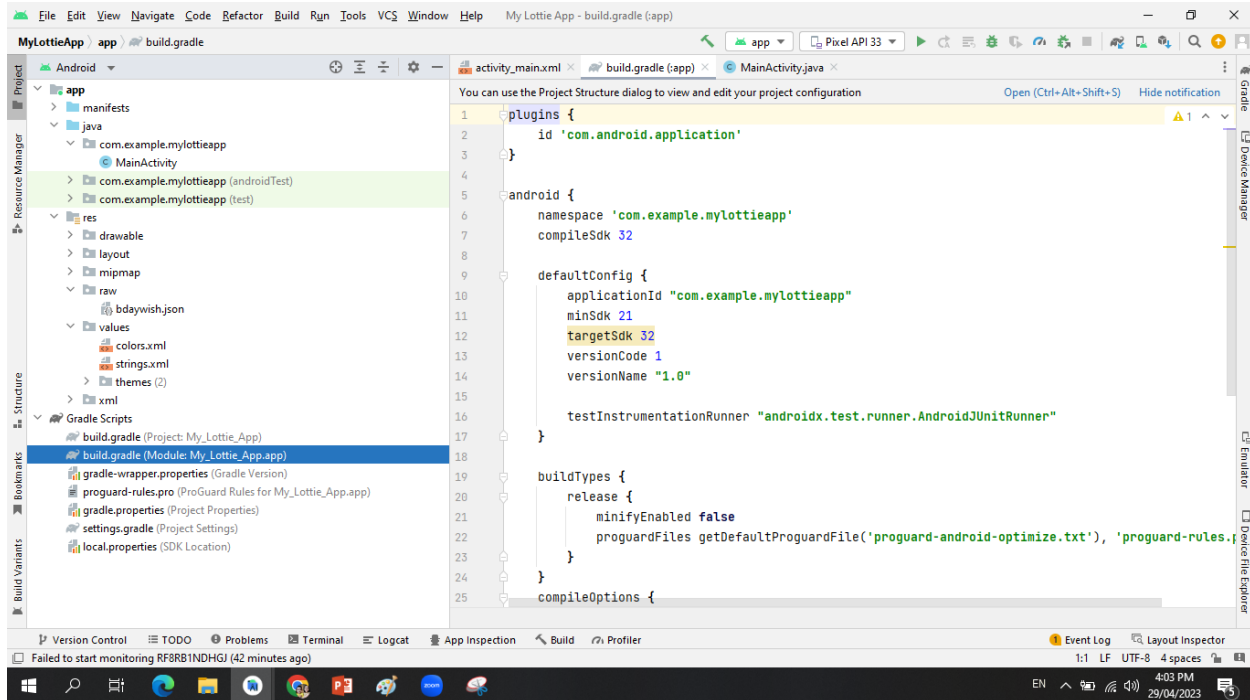
- Whenever we integrate the library in our project we have to sync it. As dependency is URL, so to sync it there will be "sync now" option available at top, click on that. You should have an active internet connection. It will download the animation project and integrate it with our android project.
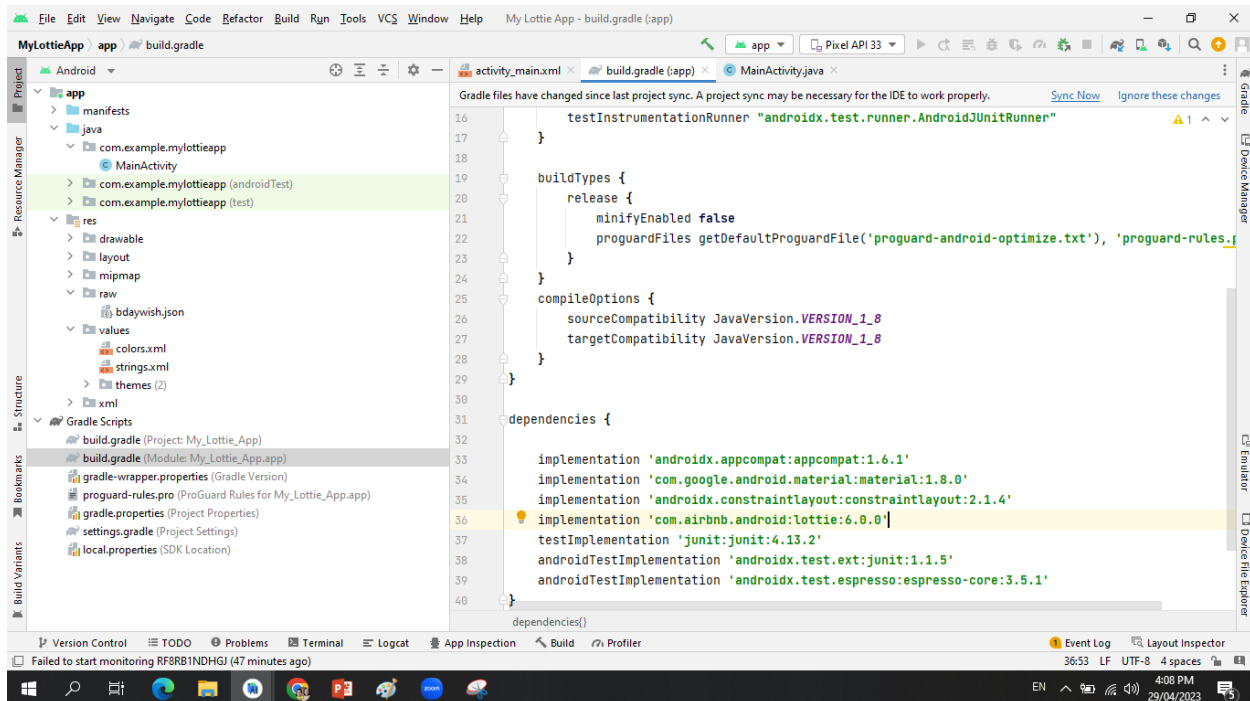
- After Syncing your project, once your Build is successful now you can proceed further.
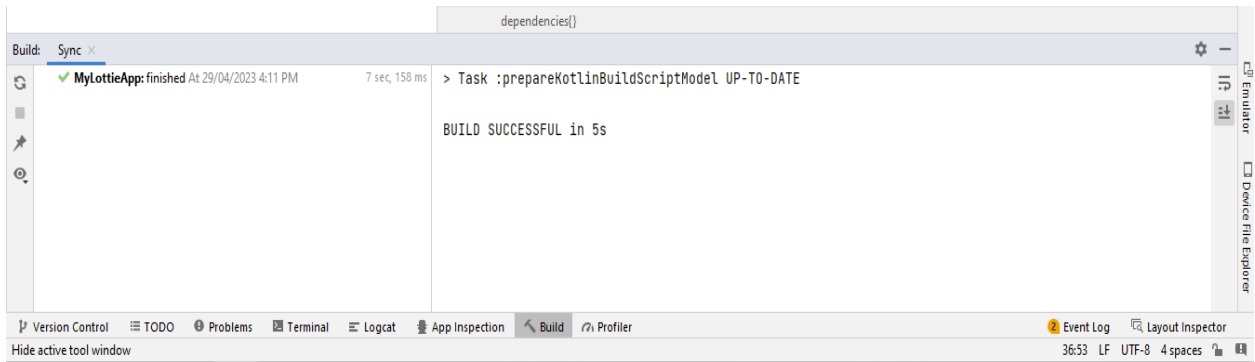
Open XML file of MainActivity.

- Start a tag "<" and just right lottie and take lottie animation view.
- The tag is completed when you take lottie animation view, you can give height and width here.
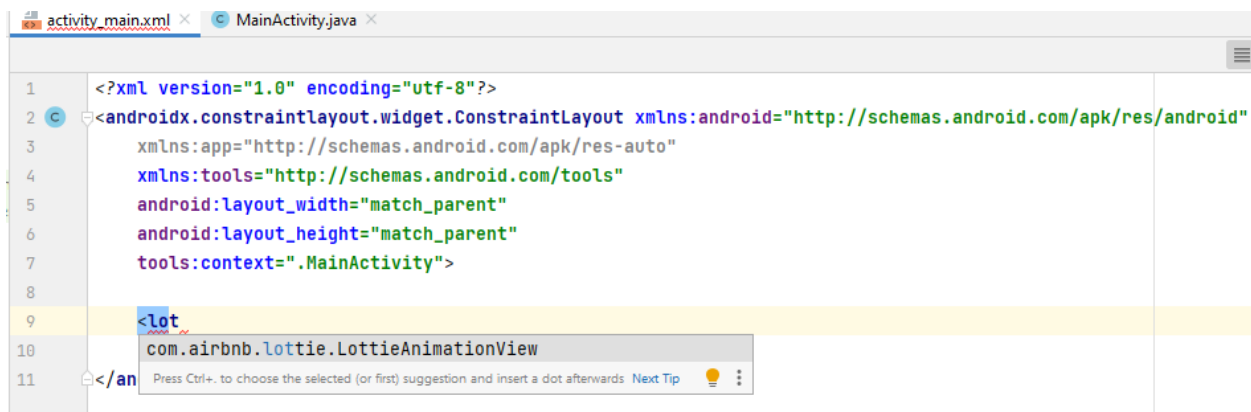
- To set the lottie animation file in XML, we take rawres and then give the name of our lottie file by telling the folder name that is @raw. To give the name write @raw/filename.
  - app:lottie_rawRes="@raw/bdaywish"
- Now take the autoplay and set is true, it will play it automatically whenever we open the file, we set it true when we want to play the animation directly from XML, no code required to be done in JAVA file.
  - app:lottie_autoPlay="true"
- If we want to play the animation in loop the we will take loop and set it as true.
  - app:lottie_loop="true"

- It will directly run the animation without any code in JAVA, if we want to run our animation by clicking some button etc then we have to do some code in java.
- Minimum compilesdk required is 33 and minsdk required is 16, so if the app is not successfully running then open gradle.build for your app and change the sdk accordingly

to run it successfully. Then click on sync now, whenever you make a change in gradle.build always sync the file.

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
1    plugins {
2        id 'com.android.application'
3    }
4
5    android {
6        namespace 'com.example.birthdaywish'
7    💡  compileSdk 33
8
9        defaultConfig {
10           applicationId "com.example.birthdaywish"
11           minSdk 21
12           targetSdk 32
13           versionCode 1
14           versionName "1.0"
15
16           testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
17       }
18
19       buildTypes {
20           release {
21               minifyEnabled false
22               proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard
23           }
24       }
```

Lottie animation play using JAVA file

- Go to the java file, take LottieAnimationView as first line under MainActivity class. Make a variable for LottieAnimationView.
  - public class MainActivity extends AppCompatActivity {
  - LottieAnimationView myView;
- Now you can set any view for your variable under the following code.
  - setContentView(R.layout.activity_main);

Run your Lottie Animation by clicking on the button (Java code)

- Here we declared a variable for LottieAnimationView and a Button.
- Connect the above declared variable with their design views using findViewById method.
- The findViewById() method is a method of Android's View and Activity classes.
- The method is used to find an existing view in your XML layout by its android:id attribute.

```
public class MainActivity extends AppCompatActivity {
    LottieAnimationView animationView;
    Button btn;

    setContentView(R.layout.activity_main);
    animationView = findViewById(R.id.main);
    btn = findViewById(R.id.button);
```

The code behind button

animationView.setVisibility(View.VISIBLE);

// this line is used to set visibility of lottie animation once the button is clicked.

btn.setVisibility(View.INVISIBLE);

//we hide button visibility when clicked to play animation.

Thread class is used to add delay for the next activity intent. It will help to sleep the next intent for 5 secs usinh try and catch and finally method of Thread class

```java
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        animationView.setVisibility(View.VISIBLE);
        btn.setVisibility(View.INVISIBLE);
        animationView.playAnimation();

        Thread thread = new Thread(){
            public void run(){
                try {
                    sleep( millis: 5000);
                }catch (Exception e){
                    e.printStackTrace();
                }finally{
                    Intent intent = new Intent( packageContext: MainActivity.this, MainActivity2.class);
                    startActivity(intent);

                }
            }

        };thread.start();
    }
});
```

ListView

- You can drop ListView from legacy tab of your palette.
- Set the height as wrap_content to expand the list as long as you have item's list.
- Set the width of your list as match_parent to make the list as wide as your layout is.
- You can adjust the height and width of your own choice by giving the values in dp.
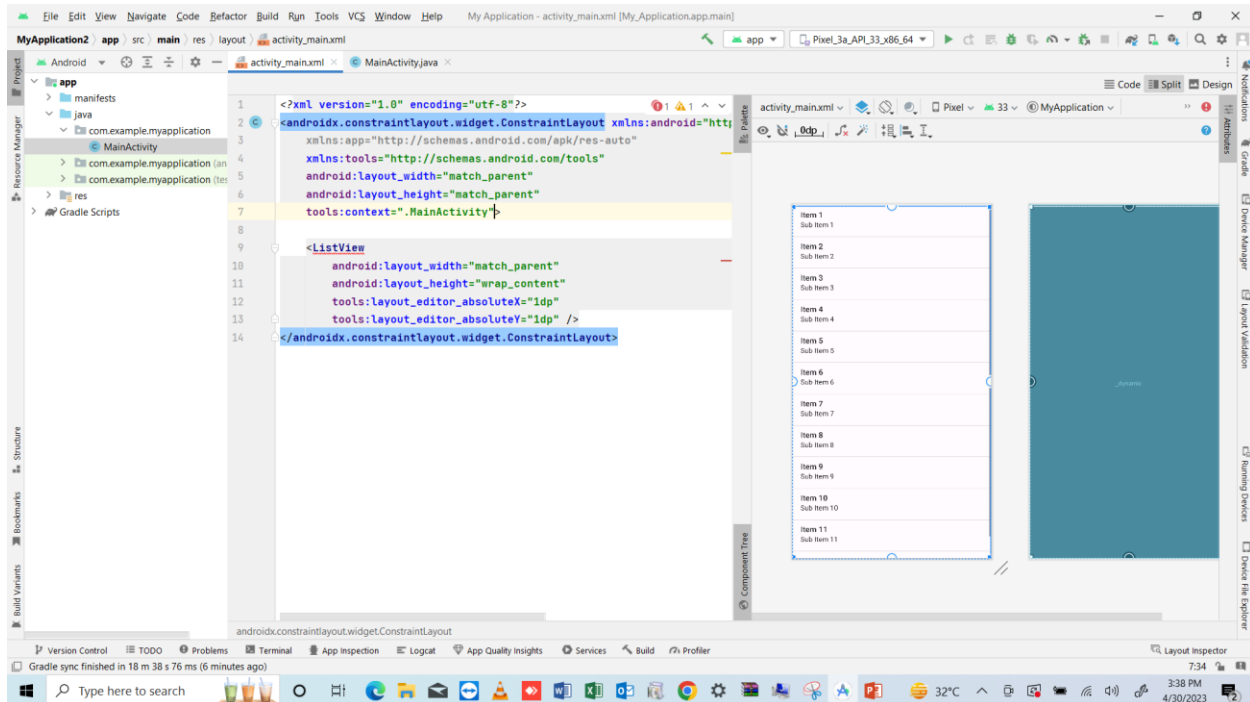
Java file

- A list has been added, now open the java file to code.
- Creat a variable of ListView type and pass the id of your list using findViewById method.
- We can add list item using array or arrayList. Array is not dynamic means it has fix length. Array list can be expanded at run time when you exactly don't know total data items to be stored.
- So we will use ArrayList. So we create an ArrayList in java, we have to give the data type of ArrayList as well in <>angle brackets.
- Syntax: ArrayList<data_type> array_name = new ArrayList<>;
- This dynamic list will allow to add as many data items in it as you wish.

To add data in ArrayList

- We use add() method to add data in the ArrayList.
- Syntax: array_name.add("data");
- Note: We add data in ArrayList under setContentView.
- Now add as many items as you want.

- Now we have to use this ArrayList. There ae two ways to do so, one is add every element manually or use android array adapter to add all the items.
- Adapter is used to set the ArrayList in the andoird ListView.

ArrayAdapter

- Note: we will do the following code under setContentView
- To create the ArrayAdapter use the following java syntax.
- Syntax: ArrayAdapter<data_type> name_of_adapter = new ArrayAdapter (class_name, TextView_layout, array_name);
- Code: ArrayAdapter<String> adapter = new ArrayAdapter<>(getApplicationContext(), android.R.layout.simple_list_item_1, arrFruits);
- Note: the data type you choose for ArrayAdapter should be same as the data_type of ArrayList. We have created String type ArrayList, so the ArrayAdapter must be of String type.
- Note: class_name means the class where you want to call it, we have three ways to class the class:
  - MainActivity.this
  - this
  - getApplicationContext()
- In this example I am calling getApplicationContext, because we have already tried the above two ways to call the same class.
- To set the text in the list, ArrayAdapter needs a layout that can be used to set the text, in android we use TextView to set the text. we do not need to create TextView layout, we just need to call that already created TextView Layout here.
- To call the already creadted layouts we use "android.R.layout.ListView" //we will use first list available in pop-up.

Setting up the ArrayAdapter on ListView

- Now after creating ArrayAdapter we have to set that to ListView so that we can use it to add data items in ListView.
- Synatx: listView_variable_name.setAdapter(adapter_name);
- Code: myListView.setAdapter(adapter);

```
2 usages
public class MainActivity extends AppCompatActivity {
    2 usages
    ListView myListView;
    9 usages
    ArrayList<String> arrFruits = new ArrayList<>();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myListView = findViewById(R.id.myList1);
        arrFruits.add("Mango");
        arrFruits.add("Banana");
        arrFruits.add("Orange");
        arrFruits.add("Apple");
        arrFruits.add("Strawberry");
        arrFruits.add("Pine Apples");
        arrFruits.add("Peach");
        arrFruits.add("Watermelon");


        //creating arrayadapter
        ArrayAdapter<String> adapter = new ArrayAdapter<>(getApplicationContext(), android.R.layout.simple_list_item_1, arrFruits);
        myListView.setAdapter(adapter);

    }
}
```

To make the items of list as clickable

```
//creating arrayadapter
ArrayAdapter<String> adapter = new ArrayAdapter<>(getApplicationContext(), android.R.layout.simple_list_item_1, arrFruits);
myListView.setAdapter(adapter);
//setting onclickListener on list's items
myListView.setOnClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
```

- We will use setOnClickListener for items to do so.
- Syntax: list_variable_name.setOnClickListener(new AdapterView.OnItemClickListener() //just take the first option and class will created automatically.
- Code: myListView.setOnClickListener(new AdapterView.OnItemClickListener() {
- In the above code, position means the position of each text in the list, as we will need to set different things against click on each item.
- We can use position of every item using if(position==0) and so on.

How to set position

- You can set position of every item in the list one by one.
- The first item's index is 0, the second item's index is 1, the third item's index is 2 and so on.
- You can do whatever you want to do on clicking the item name by using if, else-if structure in java.

- Here in example I am using Toast, on the click of every item in list.
- I use Toast to show a message, take the Toast from popup, the whole syntax will appear automatically.
- You can use intent to move to another activity where you can explain each item, or show the image, or show details etc.
- Toast is explained in upcoming slides.

```java
//creating arrayadapter
ArrayAdapter<String> adapter = new ArrayAdapter<>(getApplicationContext(), android.R.layout.simple_list_item_1, arrFruits);
myListView.setAdapter(adapter);
//setting onclickListener on list's items
myListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if(position==0){
            Toast.makeText( context: MainActivity.this,  text: "You clicked on the first item", Toast.LENGTH_SHORT).show();
        } else if (position==1) {
            Toast.makeText( context: MainActivity.this,  text: "You clicked on the second item", Toast.LENGTH_SHORT).show();
        } else if (position==2) {
            Toast.makeText( context: MainActivity.this,  text: "You clicked on the third item", Toast.LENGTH_SHORT).show();
        } else if (position==15) {
            Intent intent = new Intent( packageContext: MainActivity.this, MainActivity2.class);
            startActivity(intent);
        }

    }
});
```

What is a Toast in android

- A toast is a short, informational message that an app displays briefly near the bottom of the screen.
- Only one toast can be displayed at a time.
- The toast tells a user about an action the app has taken or will take.
- It does not require any user action or response.
- After 8 seconds, the toast disappears automatically.

Spinner in Android

- Spinners provide a quick way to select one value from a set.
- In the default state, a spinner shows its currently selected value.
- Touching the spinner displays a dropdown menu with all other available values, from which the user can select a new one.
- You can add a spinner to your layout with the Spinner object.

How to add Spinner in Project

- I added a linear layout first.
- To add the spinner, first of all we need a heading that will help the user what he list is about.
- So add a TextView. Give a text, you can adjust text size, style, color etc.
- Then add a Spinner from containers tab in the palette.
- Set the id for spinner, coz we will be using this id in our java file.
- That's all for layout, now we will code in java file.
- XML file has been pasted in the next slide for the reference.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity2">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Select the programming language you want to learn"
            android:textSize="25sp"
            android:textStyle="bold"/>

        <Spinner
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/spinner"/>

    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Let's code in java file

- Create a variable for Spinner.

- Syntax: Spinner var_name;
- Code: Spinner spinner;
- Now give the id of the spinner under setContentView portion by using findViewById method. We pass the argument using R.id.its-id

```java
public class MainActivity2 extends AppCompatActivity {
    1 usage
    Spinner spinner;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        spinner = findViewById(R.id.spinner);
    }
}
```

Create ArrayList for spinner

- Create an ArrayList for dropdown of spinner.
- Note: use the same procedure explained earlier in the slides.
- Add items in the ArrayList using add method.
- Create ArrayAdapter.
- Set ArrayAdapter to the spinner.
- Note while creating the ArrayAdapter, when choosing the android layout choose "simple_spinner_dropdown_item" from the popup.

```
 simple_dropdown_item_1line  ( = 17367050)      int
 simple_expandable_list_item_2 ( = 17367047)    int
 simple_gallery_item ( = 17367051)              int
 simple_list_item_2 ( = 17367044)               int
 simple_list_item_activated_2 ( = 17367063)     int
 simple_list_item_checked ( = 17367045)         int
 simple_list_item_multiple_choice ( = 17367056) int
 simple_list_item_single_choice ( = 17367055)   int
 simple_selectable_list_item ( = 17367061)      int
 simple_spinner_dropdown_item ( = 17367049)     int
 simple_spinner_item ( = 17367048)              int
 test_list_item ( = 17367052)                   int
 Ctrl+Down and Ctrl+Up will move caret down and up in the editor  Next Tip

    //to set-up the list we have to craete ArrayAdapter
    ArrayAdapter<String> spinnerAdapter = new ArrayAdapter<>( context: MainActivity2.this, android.R.layout.
}
```

AutoCompleteTextView in Android

- AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.
- The list of suggestions is displayed in drop down menu.
- The user can choose an item from there to replace the content of edit box with.

How to add AutoCompleteTextView

- You can choose AutoCompleteTextView from the Text tab of your palette.
- You can choose width as "match_parent" and height as "wrap_content".
- Give an id, which will be used later on in the java file to code.

```
</LinearLayout>

<AutoCompleteTextView
    android:id="@+id/actxtView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="160dp"
    android:hint="Search here..."
    app:layout_constraintTop_toBottomOf="@+id/linearLayout"
    tools:layout_editor_absoluteX="16dp" />
```

Code in java file

- Create a variable for AutoCompleteTextView.
- Add it's id with it using the method findViewByID.
- Create an ArrayList for dropdown of AutoCompleteTextView.
- Note: use the same procedure explained earlier in the slides.
- Add items in the ArrayList using add method.
- Create ArrayAdapter.
- Note while creating the ArrayAdapter, when choosing the android layout choose "simple_list_item_1" from the popup.
- Set ArrayAdapter to the AutoCompleteTextView.
- Set threshold using AutoCompleteTextView_variable.setThreshold();
- We can set threshold for the list, means after how many letter typed by the user, the items from the list should be shown to auto complete the list.

- I am giving threshold 1 means, the data items will appear after the user type one letter.
- The code is pasted in next slide.

```java
public class MainActivity2 extends AppCompatActivity {

    2 usages
    Spinner spinner;

    3 usages
    AutoCompleteTextView actxtView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        spinner = findViewById(R.id.spinner);
        actxtView = findViewById(R.id.actxtView);
        ArrayList<String> arrLanguages = new ArrayList<>(); //i used arr with name just to remind me that i used array.
        ArrayList<String> arrSearch = new ArrayList<>(); //ArrayList for AutoCompleteTextView
```

- Note:
- The above code contains Spinner and AutoCompleteTextView java file code.

```java
//to add data in my autocompletetextview list
arrSearch.add("C");
arrSearch.add("C++");
arrSearch.add("C sharp");
arrSearch.add("Perl");
arrSearch.add("Php");
arrSearch.add("Objective C");

//array adapter for autocomplete text view
ArrayAdapter<String> actvAdapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, arrSearch);

//set adapter on autoCompleteTextView
actxtView.setAdapter(actvAdapter);

//giving the threshold
actxtView.setThreshold(1);
```

RecyclerView in Android

- RecyclerView is the ViewGroup that contains the views corresponding to your data.
- It's a view itself, so you add RecyclerView to your layout the way you would add any other UI element.
- Each individual element in the list is defined by a view holder object.
- RecyclerView makes it easy to efficiently display large sets of data.
- You supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they're needed.
- As the name implies, RecyclerView recycles those individual elements.

How to add RecyclerView

- You can add RecyclerView from Container Tab of your design palette.
- Create a variable for RecyclerView in java.
- Assign the id to this variable using findViewById method.

Creating Layout Resource File

- To use the RecyclerView we have to create the new layout.
- On the right most pane of your project, go to res directory, then expand layout folder, there is only one layout available currently.
- right click on layout, then click on New, then select Layout resource file. A new dialog box appears.
- Give the file name e.g sample_recyclerview and press OK.

res
dr
lay
mi
va
xm
res (g
Gradle Sc
build.
build.
progu
gradl

un: app

04/
Ins
$ a
Ope
Con

Version Contr
reate a new Lay

New	>
Add C++ to Module	
Cut	Ctrl+X
Copy	Ctrl+C
Copy Path/Reference...	
Paste	Ctrl+V
Find Usages	Alt+F7
Find in Files...	Ctrl+Shift+F
Replace in Files...	Ctrl+Shift+R
Analyze	>
Refactor	>
Bookmarks	>
Show In Resource Manager	Ctrl+Shift+T
Reformat Code	Ctrl+Alt+L
Optimize Imports	Ctrl+Alt+O
Delete...	Delete
Open In	>
Local History	>
Repair IDE on File	
Reload from Disk	
Compare With...	Ctrl+D
Mark Directory as	>
Convert Java File to Kotlin File	Ctrl+Alt+Shift+K

Kotlin Class/File
Layout Resource File
Sample Data Directory
File
Scratch File   Ctrl+Alt+Shift+Insert
Directory
Image Asset
Vector Asset
Kotlin Script (Beta)
Kotlin Worksheet (Beta)
CMakeLists.txt
Activity   >
Fragment   >
Folder   >
Service   >
UiComponent   >
Automotive   >
XML   >
Wear   >
AIDL   >
Widget   >
Google   >
Compose   >
Other   >
Resource Bundle
EditorConfig File

New Resource File   ✕

File name: sample_recyclerview

Root element: androidx.constraintlayout.widget.ConstraintLayout

Source set: main *src/main/res*   ▼

Directory name: layout

Available qualifiers: Chosen qualifiers:

Country Code
Network Code
Locale
Layout Direction
Smallest Screen Width
Screen Width   >>   Nothing to show
Screen Height
Size   <<
Ratio
Orientation
UI Mode
Night Mode

?   OK   Cancel

CardView in Android

- The system provides the CardView API as an easy way for you to show information inside cards that have a consistent look across the platform.
- These cards have a default elevation above their containing view group, so the system draws shadows below them.
- Cards provide an easy way to contain a group of views while providing a consistent style for **the container.**

CardView in palette

- CardView is (or maybe) available in the containers tab of palette.
- If it is not available then you can download the library by clicking on the downloading arrow.
- Then you can use it.
- To add some images in card view, we have to download some png images. Copy all the downloaded images and paste in drawables folder of res directory.
- Add linear layout on your CardView.
- Then add ImageView on it, and add your image from the drawbales. If the image is not visible properly then you can scale it, search scaleType in attributes and adjust accordingly.

Model Class in RecyclerView

- Your model would usually be a set of classes that hold your data and business logic.
- To create model class in your android project.
- On the right most pane of your project, expand Java, then right click on the package name, then click on the New then select Package.
- A new dialog box appear where you can name your package. For example Models.
- Now we have to create new java class for the above created mode.
- Go to the model name you created, right click on it, click New, then click Java Class.
- A new dialog box appears, give the name to your class.
- I named it as FoodModel.
- We hae already created first card view, we create the model to replicate the first card view using get and set methods.
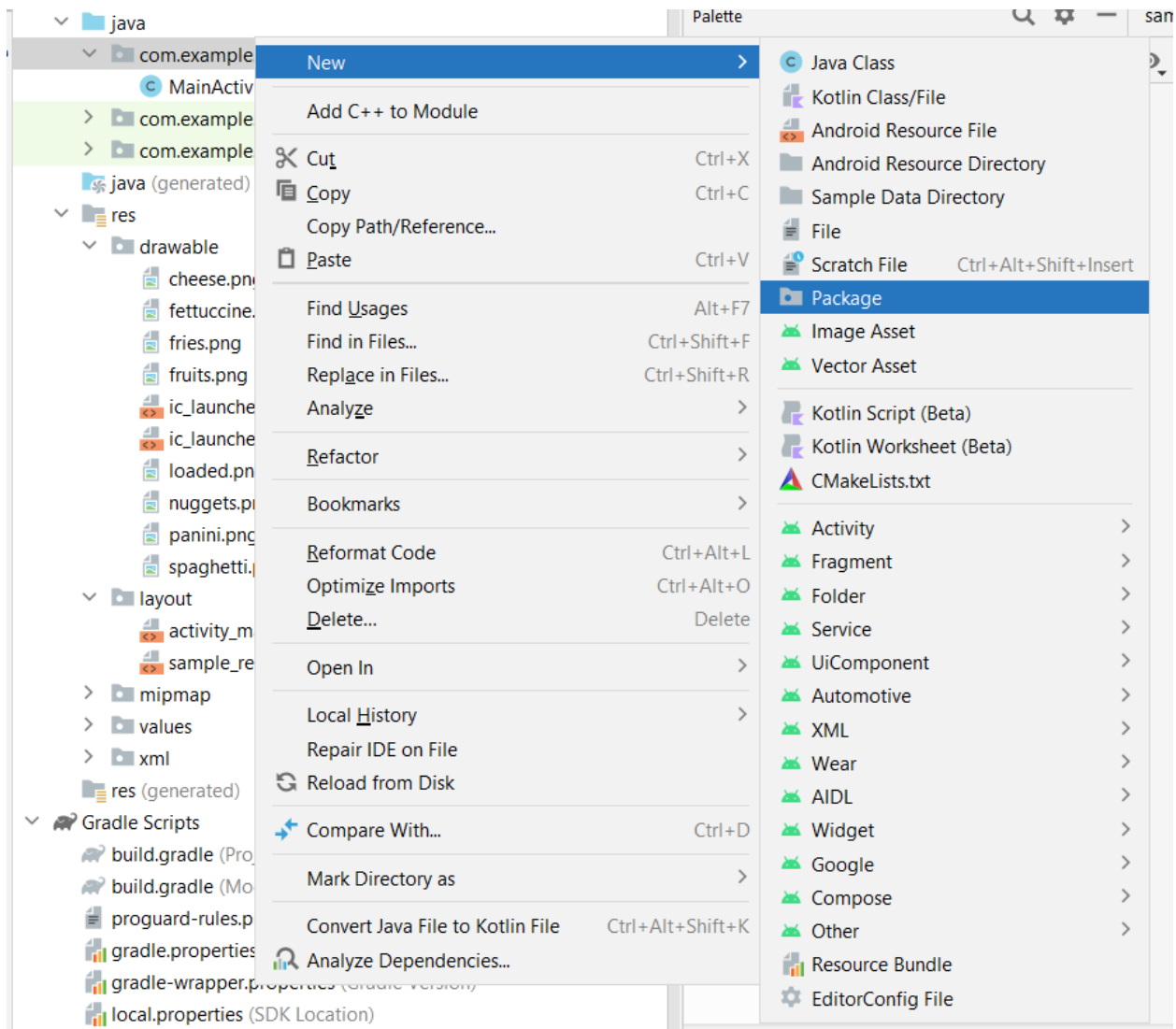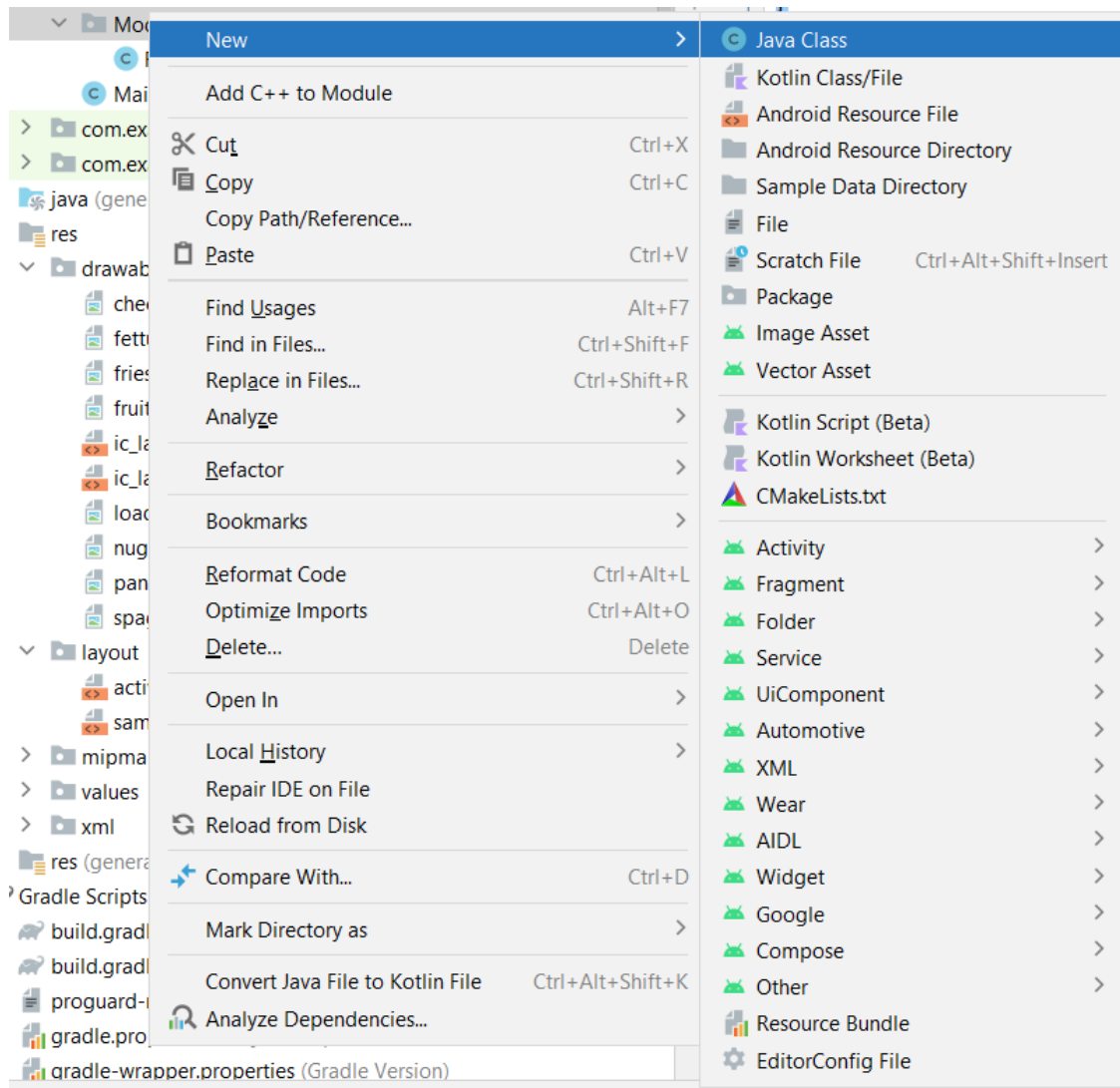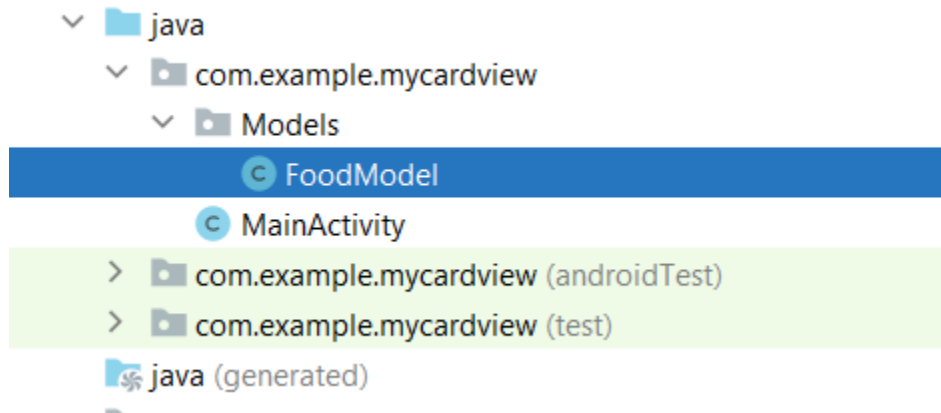
Java code for the new java class created in the model package.

- We have two things in our CardView, first is image and second is TextView.
- By default the picture are saved in binary numbers so the picture data type will be int.
- Data type for TextView will be String.


```
package com.example.mycardview.Models;

public class FoodModel {
    int images1;
    String text;
```

- Now we have to create constructor for our class. Right click then choose generate, then choose the first option Constructor.
- A new dialog box appears, select both the data items available and press OK. (as shown on next slide)

💡 Show Context Actions		Alt+Enter
📋 Paste		Ctrl+V
Copy / Paste Special		>
Column Selection Mode		Alt+Shift+Insert
Find Usages		Alt+F7
Find Sample Code		Alt+F8
Refactor		>
Folding		>
Analyze		>
Go To		>
Generate...		Alt+Insert
Open In		>
Local History		>
Compare with Clipboard		
Create Gist...		

Generate

- Constructor
- Getter
- Setter
- Getter and Setter
- equals() and hashCode()
- toString()
- Override Methods...   Ctrl+O
- Delegate Methods...
- Test...
- Copyright

Code for constructor class

```java
public FoodModel(int images1, String text) {
    this.images1 = images1;
    this.text = text;
}
```

Generate geter and seter

▶ Right click then click on generate, select getter and setter.

▶ A new window appears, select both data items and press Ok.

▶ The code will appear in the class.

▶ Getters and setters are used to protect your data, particularly when creating classes.

▶ For each instance variable, a getter method returns its value while a setter method sets or updates its value.

Generate

Constructor
Getter
Setter
Getter and Setter
equals() and hashCode()
toString()
Override Methods...   Ctrl+O
Delegate Methods...
Test...
Copyright

Select Fields to Generate Getters and Setters   ✕

Getter template: IntelliJ Default

Setter template: IntelliJ Default

com.example.mycardview.Models.FoodModel
   f ○   images1:int
   f ○   text:String
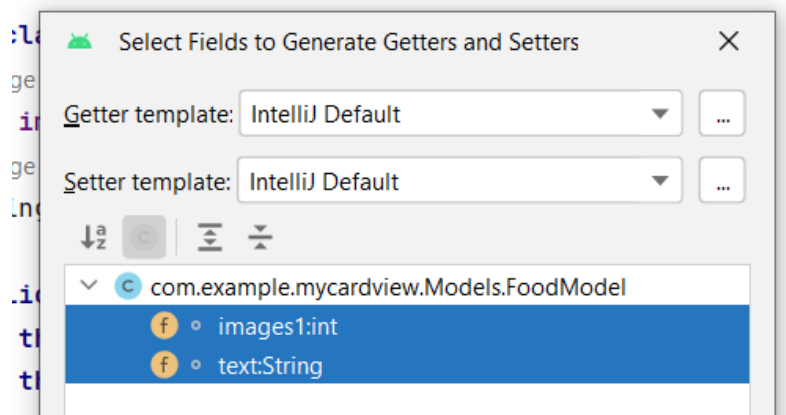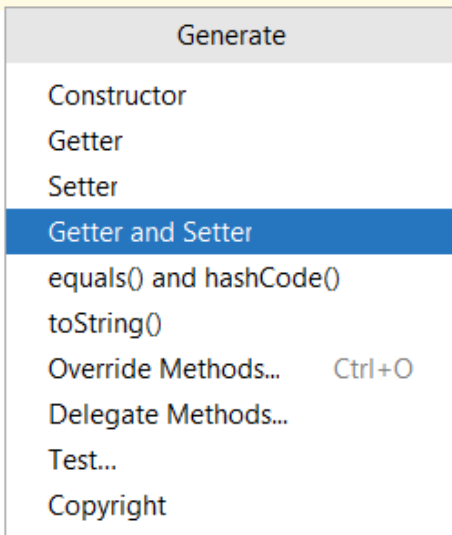
```java
public int getImages1() {
    return images1;
}

public void setImages1(int images1) {
    this.images1 = images1;
}

public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}
```

Creating Adapter

- Adapter is used when we have to get data from some data source.
  - For example earlier we used to get data from ArrayList,we created the ArrayAdapter.
- Now we will create a complete new Adapter class.
- To do so, On the right most pane of your project, expand Java, then right click on the package name, then click on the New then select Package.
- A new dialog box appear where you can name your package. For example Adapters.
- Now right click on Adapter and click New, then click Java Class.
- A new dialog box appears, give the name to your class. I name is as "FoodAdapter"
- we always name Java class with first capital list.
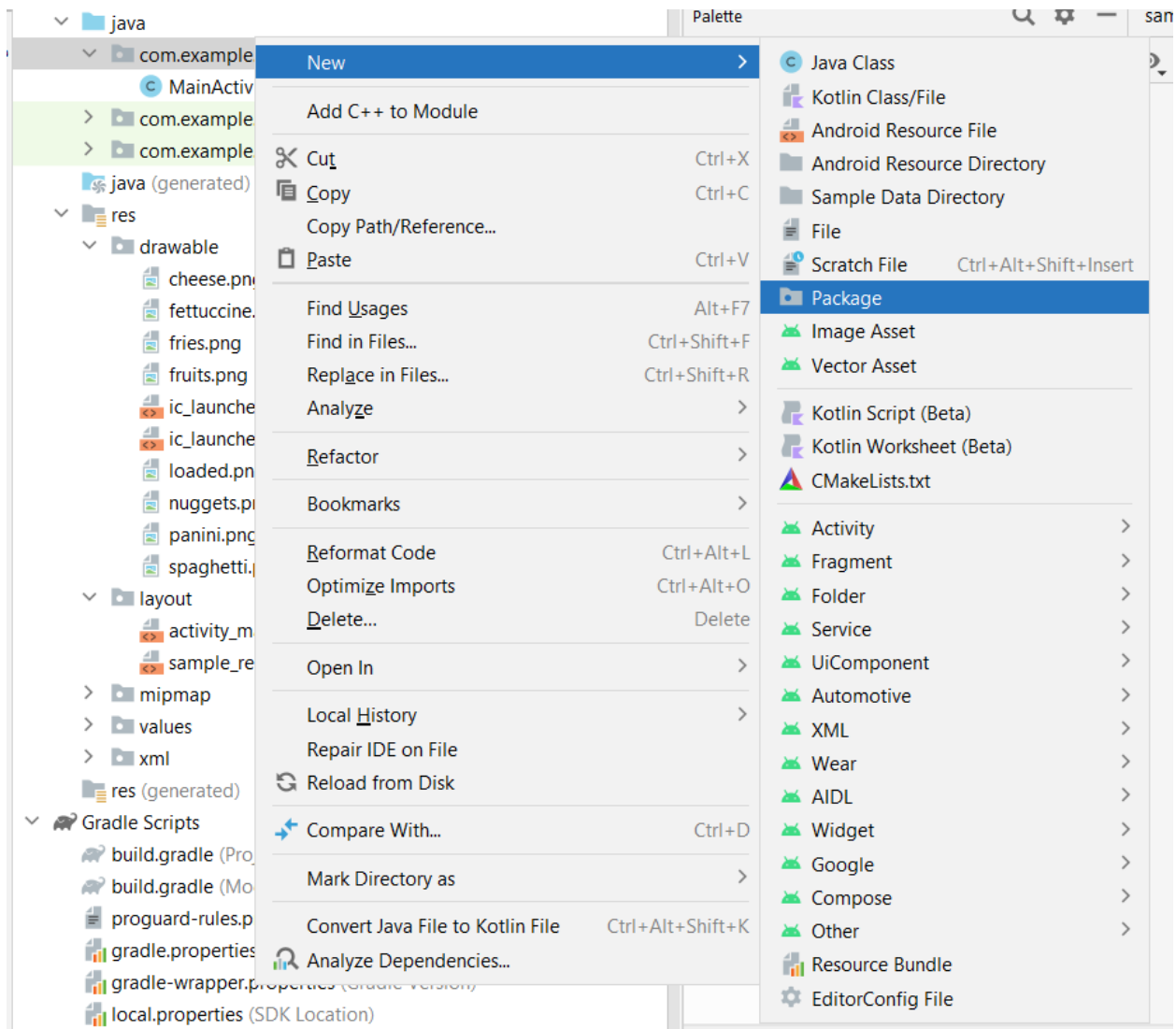
Code in Adapter Java File

► Create public class with name viewHolder and then extend it with RecyclerVew.ViewHolder. A constructor will be created.

► Code: public class viewHolder extends RecyclerView.ViewHolder

► viewHolder class will be used to find and attach the id of UI view with its declared variable, by the following way.

► Now use create ImageView and TextView variables.

► Use itemview.findViewById method to give respective ids to both.

► Now extends the FoodAdapter class with RecyclerAdapter and then use viewHolder class.

► A red line appears under code, click on implement methods, and add all the methods from new window that appeared.

```java
1 usage
public class FoodAdapter extends RecyclerView.Adapter<FoodAdapter.viewHolder> {


        1 usage
        public class viewHolder extends RecyclerView.ViewHolder {

            1 usage
            ImageView imgviw;
            1 usage
            TextView txtview;

            public viewHolder(@NonNull View itemView) {
                super(itemView);
                imgviw = itemView.findViewById(R.id.imageView);
                txtview = itemView.findViewById(R.id.textView);
            }

        }
```

💡 Implement methods	
❗ Make 'FoodAdapter' abstract	>
Press Ctrl+Q to toggle preview	
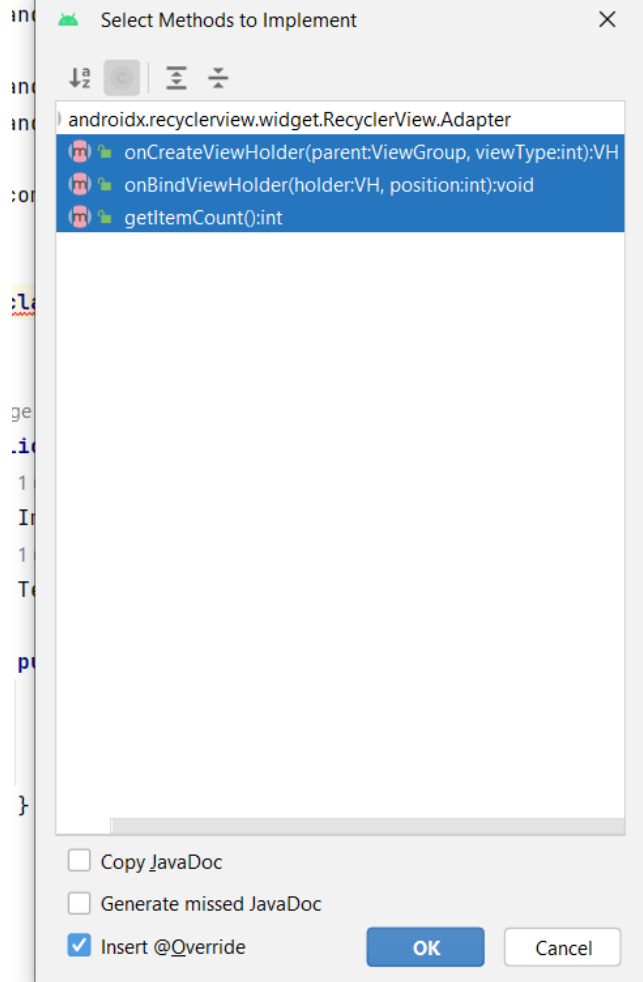
```java
public class FoodAdapter extends RecyclerView.Adapter<FoodAdapter.viewHolder> {


    @NonNull
    @Override
    public viewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }


    @Override
    public void onBindViewHolder(@NonNull viewHolder holder, int position) {


    }


    @Override
    public int getItemCount() {
        return 0;
    }

    3 usages
    public class viewHolder extends RecyclerView.ViewHolder {
        1 usage
        ImageView imgviw;
        1 usage
        TextView txtview;

        public viewHolder(@NonNull View itemView) {
            super(itemView);
            imgviw = itemView.findViewById(R.id.imageView);
            txtview = itemView.findViewById(R.id.textView);
        }
    }
}
```

- Now we will create an ArrayList in this class.
- Syntax: ArrayList<data_type> array_list_name;
- As we have created our own data_type that is model we created, so we will use that model name for our data_type.
- Syntax: ArrayList<model_name> list_name;
- Code: ArrayList<FoodModel> arrlist;
- Then create a variable for Context.
- Right click to create the constructor for above two variables.
- Code

```
1 usage
public class FoodAdapter extends RecyclerView.Adapter<FoodAdapter.viewHolder> {

    1 usage
    ArrayList<FoodModel> arrlist;
    1 usage
    Context context;


    public FoodAdapter(ArrayList<FoodModel> arrlist, Context context) {
        this.arrlist = arrlist;
        this.context = context;
    }

    @NonNull
    @Override
```

- As we have three new methods as shown in the above code in FoodAdapter java class file.
- onCreateViewHolder:
  - It is used to simply inflate the layout with this class.
- onBindViewHolder:
  - It is used to set the values of data items we have taken, here we have taken imageView and TextView.
- getItemCount:
  - It is used to cound the number of items available in list, that is to get the size of list.
  - Just change the return 0 to return list_name.size();

```
@Override
public int getItemCount() {
    return arrlist.size();
}
```

- Now we will inflate the layout in this Adapter class.
- We will use View view = LayoutInflater.from(context).inflate(R.layout.sample_recyclerview, parent, false)
- Here sample_recyclerview is the name of our XML file where we created the CardView.
- Parent, false is used to set the size of each image as same.
- Now change the return null; to return new viewHolder(view);

```
@Override
public viewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(context).inflate(R.layout.sample_recyclerview, parent, attachToRoot: false);
    return new viewHolder(view);
}
```

- Now on the method onBindViewHolder, make the following change.
- Code: FoodModel model = arrlist.get(position);
- Syntax: Model_class_name model_name = list_name.get(position)
- Position means that we will pass the data in list using this postion, position for first image is 0, for next is 1 and so on.

```
@Override
public void onBindViewHolder(@NonNull viewHolder holder, int position) {
    FoodModel model = arrlist.get(position);
    //now i am setting data of images using viewHolder
    holder.imgviw.setImageResource(model.getImages1()); //here this method of getImages1 is taken from FoodModel class
    holder.txtview.setText(model.getText());
```

MainActivity Java File code

- We have already created RecyclerView variable in the MainAcivity java file, and have passed it it's id.
- Now create an ArrayList using the FoodModel we created as data-type.
- Syntax: ArrayList<data_type> list_name = new ArrayList();
- Code: ArrayList<FoodModel> arrlist = new ArrayList<>();
- Now we have to add items in the list, the list items can be added using two arguments according to our model, first argument is image_name and second argument is text.
- Syntax: list_name.add(new ModelName(R.drawable.image_name, "text you want to show")
- Code: arrlist.add(new FoodModel(R.drawable.cheese, "Cheese Burger"));
- Using the adapter in main activity java file:
- Synatx: Adapter_name var_name_for_adapter = new Adapter_name (arguments, this);
- Code: FoodAdapter adapter = new FoodAdapter(arrlist, this);
- Now set the above adapter on recyclerview.
- Synatx: recyclerView_name.setAdapter(var_name_for_adapter);
- Code: recyclerView.setAdapter(adapter);

```java
public class MainActivity extends AppCompatActivity {
    2 usages
    RecyclerView recyclerView;
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.reView);

        ArrayList<FoodModel> arrlist = new ArrayList<>();
        arrlist.add(new FoodModel(R.drawable.cheese,    text: "Cheese Burger"));
        arrlist.add(new FoodModel(R.drawable.fettuccine,    text: "Fettucine Alfredo Pasta"));
        arrlist.add(new FoodModel(R.drawable.fries,    text: "French Fries"));
        arrlist.add(new FoodModel(R.drawable.fruits,    text: "Fruits cocktail"));
        arrlist.add(new FoodModel(R.drawable.loaded,    text: "Loaded Cheese Fries"));
        arrlist.add(new FoodModel(R.drawable.nuggets,    text: "Chicken Nuggets"));
        arrlist.add(new FoodModel(R.drawable.panini,    text: "Panini Grilled Sandwich"));
        arrlist.add(new FoodModel(R.drawable.spaghetti,    text: "Chilli sauce Spaghetti"));

        //now using the Adapter

        FoodAdapter adapter = new FoodAdapter(arrlist,    context: this);
        recyclerView.setAdapter(adapter);
```

Layouts Manager

We have three layouts

- Linear Layout Manager
- Grid Layout Manager
- Staggered Layout Manager

a combiination of two or three things is called staggered layout.

Linear Layout Manager

Horizontal Scroll View

For right side scroll set the value as false, for left side scroll set the value as true.

```java
//horizontal linear layout manager
LinearLayoutManager layoutManager = new LinearLayoutManager( context: this, LinearLayoutManager.HORIZONTAL,    reverseLayout: true);
recyclerView.setLayoutManager(layoutManager);
```

Vertical Scroll View

to manage layout we have a class called linear Layout Manager, then we create the object of it.

```
//vertical linear layout manager
LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
recyclerView.setLayoutManager(layoutManager);
```

Grid Layout Manager

- We can use multiple span to show Grid Layout. For example 2, 3, 4 and so on.
- You can change the spancount according to your own wish.

```
//using Grid Layout manager
GridLayoutManager gridLayoutManager = new GridLayoutManager( context: this,  spanCount: 2);
recyclerView.setLayoutManager(gridLayoutManager);
```

Staggered Layout Manager

```
// using staggered layout manager
StaggeredGridLayoutManager staggered = new StaggeredGridLayoutManager( spanCount: 2, StaggeredGridLayoutManager.HORIZONTAL);
recyclerView.setLayoutManager(staggered);
```
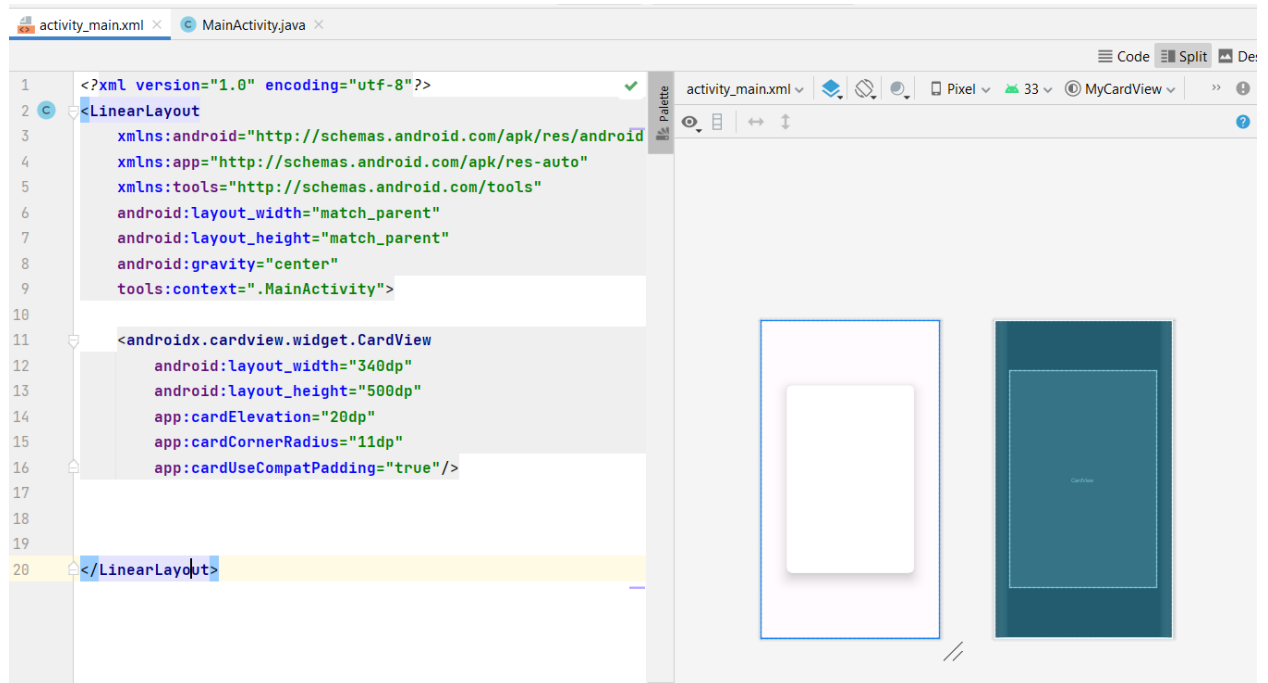
- First argument which we pass in it is spanCount, and then the view that can be horizontal or vertical.
- There re many other ways to use staggeredGridLayoutManager.

```java
3      import ...
17

       2 usages
18     public class MainActivity extends AppCompatActivity {
           3 usages
19         RecyclerView recyclerView;
20         @SuppressLint("MissingInflatedId")
21         @Override
22         protected void onCreate(Bundle savedInstanceState) {
23             super.onCreate(savedInstanceState);
24             setContentView(R.layout.activity_main);
25             recyclerView = findViewById(R.id.reView);
26
27             ArrayList<FoodModel> arrlist = new ArrayList<>();
28             arrlist.add(new FoodModel(R.drawable.cheese,     text: "Cheese Burger"));
29             arrlist.add(new FoodModel(R.drawable.fettuccine,  text: "Fettucine Alfredo Pasta"));
30             arrlist.add(new FoodModel(R.drawable.fries,      text: "French Fries"));
31             arrlist.add(new FoodModel(R.drawable.fruits,     text: "Fruits cocktail"));
32             arrlist.add(new FoodModel(R.drawable.loaded,     text: "Loaded Cheese Fries"));
33             arrlist.add(new FoodModel(R.drawable.nuggets,    text: "Chicken Nuggets"));
34             arrlist.add(new FoodModel(R.drawable.panini,     text: "Panini Grilled Sandwich"));
35             arrlist.add(new FoodModel(R.drawable.spaghetti,  text: "Chilli sauce Spaghetti"));
36
37             //now using the Adapter
38
39             FoodAdapter adapter = new FoodAdapter(arrlist,  context: this);
40             recyclerView.setAdapter(adapter);
41
42             //vertical linear layout manager
43             LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
44             recyclerView.setLayoutManager(layoutManager);
45
```

☰ Code ▐█ Split ▣ Des

```xml
1    <?xml version="1.0" encoding="utf-8"?>
2    <LinearLayout
3        xmlns:android="http://schemas.android.com/apk/res/android
4        xmlns:app="http://schemas.android.com/apk/res-auto"
5        xmlns:tools="http://schemas.android.com/tools"
6        android:layout_width="match_parent"
7        android:layout_height="match_parent"
8        android:gravity="center"
9        tools:context=".MainActivity">
10
11       <androidx.cardview.widget.CardView
12           android:layout_width="340dp"
13           android:layout_height="500dp"
14           app:cardElevation="20dp"
15           app:cardCornerRadius="11dp"
16           app:cardUseCompatPadding="true"/>
17
18
19
20   </LinearLayout>
```

activity_main.xml ⌄   ◆⌄ ◇ ◉⌄   📱 Pixel ⌄   🤖 33 ⌄   © MyCardView ⌄   ›› ❶

👁⌄ ▤   ↔ ↕   ❓

Add Firebase to your Android project

Prerequisites

- Install or update Android Studio to its latest version.
- Make sure that your project meets these requirements (note that some products might have stricter requirements):
  - Targets API level 21 (Lollipop) or higher
  - Uses Android 5.0 or higher
  - Uses Jetpack (AndroidX), which includes meeting these version requirements:

    com.android.tools.build:gradle v7.3.0 or later

    - compileSdkVersion 28 or later
- Set up a physical device or use an emulator to run your app.
- Note that Firebase SDKs with a dependency on Google Play services require the device or emulator to have Google Play services installed.
- Sign into Firebase using your Google account.

You can connect your Android app to Firebase using one of the following options:

Option 1: (recommended) Use the Firebase console setup workflow.

Option 2: Use the Android Studio Firebase Assistant (may require additional configuration).

Step 1: Set up a Firebase project

- Go to Firebase Console.
- Click on Add Project.
- Name your project, then follow the steps to create it.
- Once created, you'll be directed to the project overview page.

Step 2: Register your Android app with Firebase

- In the Firebase console, select your project and click on Add app > Android.
- Enter your Android package name (you can find this in AndroidManifest.xml).
- Optionally, give your app a nickname and enter your SHA-1 certificate (if needed for features like Google Sign-In).
- Click Register app.

Step 3: Download the google-services.json file

- After registering your app, download the google-services.json file.
- Move this file to your Android Studio project's app directory.

Step 4: Add Firebase SDK dependencies

- Open your project-level build.gradle file and ensure you have Google's Maven repository:

```
buildscript {
  dependencies {
    classpath 'com.google.gms:google-services:4.3.10' // use latest version
  }
}
allprojects {
  repositories {
    google()
    mavenCentral()
  }
}
```

- Open your app-level build.gradle file and add the following

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services' // at the end of the file

dependencies {
  implementation platform('com.google.firebase:firebase-bom:32.0.0') // use latest version
  // Add individual Firebase products as needed, for example:
  implementation 'com.google.firebase:firebase-analytics'
  implementation 'com.google.firebase:firebase-auth'
}
```

- Sync your project by clicking Sync Now when prompted.

Step 5: Initialize Firebase in your app

Open your main Application class (or MainActivity) and initialize Firebase:

```
import com.google.firebase.FirebaseApp;
```

```java
public class MyApplication extends Application {

    @Override

    public void onCreate() {

        super.onCreate();

        FirebaseApp.initializeApp(this);

    }

}
```

- Alternatively, in MainActivity:

```java
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.FirebaseApp;


public class MainActivity extends AppCompatActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        FirebaseApp.initializeApp(this);

    }

}
```

Step 6: Add Firebase services (optional)

To add specific Firebase services (like Authentication, Firestore, etc.), add the appropriate dependencies to your build.gradle (app-level) file. Examples include:

- **Firebase Authentication**

implementation 'com.google.firebase:firebase-auth'

- **Firebase Firestore:**

implementation 'com.google.firebase:firebase-firestore'

- **Firebase Realtime Database:**

implementation 'com.google.firebase:firebase-database'

Step 7: Test the Firebase integration

- Go back to the Firebase Console.
- Under Analytics or Authentication, perform an operation (e.g., login, database read/write) in your app to confirm Firebase is working.