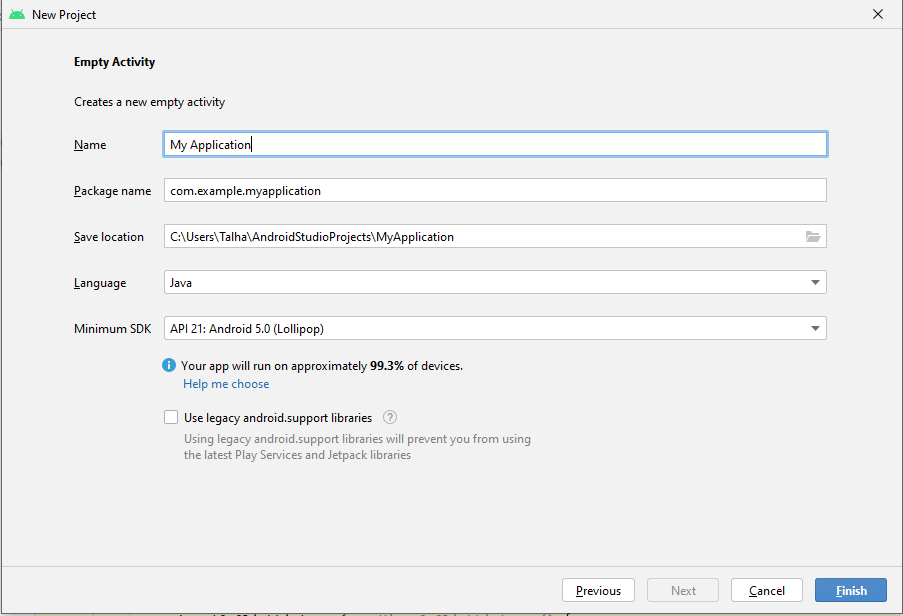
Open Android Studio. Create a new Android Studio project with an empty activity.

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() {

double kg = Double.parseDouble(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.

**setOnClickListener in Android Studio:**

In Android, the OnClickListener() interface has an onClick(View v) method that is called when the view (component) is clicked. The code for a component's functionality is written inside this method, and the listener is set using the setOnClickListener() method.

**AppCompatActivity** is a base class that we can extend for using newer platform features on older Android devices. Some of these backported features include the usage of the action bar, including action items, etc. It can switch between light and dark themes by using default androidx. appcompat themes.

**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.