**Section 9: Unit Converter App - KG to Pounds**

**Key Concepts Taught**

1. **Basic UI Components**: EditText, Button, TextView
2. **Data Conversion**: String to double parsing
3. **Mathematical Operations**: Unit conversion formula
4. **Event Handling**: Button click listeners
5. **UI Styling**: Background images, text colors

**Implementation Steps with Detailed Comments**

**1. XML Layout (**activity\_main.xml**)**

xml

<androidx.constraintlayout.widget.ConstraintLayout

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

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@drawable/background\_image"> *<!-- Custom background -->*

*<!-- Input field for kilograms -->*

<EditText

android:id="@+id/editText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter weight in kilos"

android:textColor="@android:color/white" <!-- White text for visibility -->

android:textColorHint="@android:color/white" *<!-- White hint text -->*

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"/>

*<!-- Conversion trigger -->*

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Convert kilos to pounds"

app:layout\_constraintTop\_toBottomOf="@id/editText"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"/>

*<!-- Result display -->*

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="0"

android:textSize="70sp"

android:textColor="@android:color/white" <!-- White text for contrast -->

app:layout\_constraintTop\_toBottomOf="@id/button"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>

**Comments**:

* Background image improves visual appeal
* White text ensures visibility against dark background
* ConstraintLayout positions elements vertically
* Large 70sp text size for clear result display

**2. MainActivity Logic (**MainActivity.java**)**

java

public class MainActivity extends AppCompatActivity {

private EditText editText;

private Button button;

private TextView textView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

*// 1. Initialize UI components*

editText = findViewById(R.id.editText);

button = findViewById(R.id.button);

textView = findViewById(R.id.textView);

*// 2. Set button click listener*

button.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

*// 3. Get user input*

String inputText = editText.getText().toString();

*// 4. Validate and convert input*

if (!inputText.isEmpty()) {

try {

double kilos = Double.parseDouble(inputText);

double pounds = convertKilosToPounds(kilos);

textView.setText(String.valueOf(pounds));

} catch (NumberFormatException e) {

*// Handle invalid input*

Toast.makeText(MainActivity.this,

"Enter valid number!", Toast.LENGTH\_SHORT).show();

}

}

}

});

}

*// 5. Conversion method (1 kg = 2.20462 lbs)*

private double convertKilosToPounds(double kilos) {

return kilos \* 2.20462;

}

}

**Step-by-Step Explanation**:

1. **Component Initialization**: Connect XML elements to Java objects
2. **Click Listener**: Triggers conversion when button is pressed
3. **Input Handling**: Retrieve text from EditText
4. **Error Handling**:
   * Check for empty input
   * Catch NumberFormatException for non-numeric input
5. **Conversion**: Separate method for clean unit conversion logic
6. **Result Display**: Update TextView with formatted result

**Tools & APIs Used**

* **Android APIs**: EditText, Button, TextView, Toast
* **Key Classes**: NumberFormatException, Double
* **Design Elements**: Background images, text color customization

**Best Practices & Modern Approaches**

1. **Input Validation**:

java

*// Use InputFilter to restrict to numbers only*

editText.setInputType(InputType.TYPE\_CLASS\_NUMBER | InputType.TYPE\_NUMBER\_FLAG\_DECIMAL);

1. **Decimal Formatting**:

java

*// Format to 2 decimal places*

DecimalFormat df = new DecimalFormat("#.##");

textView.setText(df.format(pounds));

1. **Resource Management**:
   * Use string resources for UI text (strings.xml)
   * Extract conversion constant to resources:

xml

<dimen name="kg\_to\_lb">2.20462</dimen>

1. **Modern Alternatives**:
   * **Data Binding**: Reduce boilerplate code
   * **ViewModel**: Preserve data during configuration changes
   * **LiveData**: Auto-update UI on data changes
2. **Accessibility**:

xml

android:contentDescription="Result in pounds"

android:importantForAccessibility="yes"

**Part B: Important Topics Not Covered**

1. **Multi-Unit Conversion**:
   * Implement spinner for unit selection
   * Create conversion matrix:

java

switch(selectedUnit) {

case "Pounds": return kg \* 2.20462;

case "Ounces": return kg \* 35.274;

*// ...*

}

1. **Currency Conversion Patterns**:
   * Use NumberFormat for locale-specific formatting:

java

NumberFormat.getInstance(Locale.US).format(pounds)

1. **Reverse Conversion**:
   * Add toggle button for kg→lb and lb→kg modes
   * Update formula dynamically
2. **History Feature**:
   * Save conversions to SQLite/Realm
   * Display in RecyclerView
3. **Architecture Components**:

java

*// MVVM implementation*

public class ConverterViewModel extends ViewModel {

MutableLiveData<Double> result = new MutableLiveData<>();

public void convert(double kg) {

result.setValue(kg \* 2.20462);

}

}

1. **Testing**:
   * Unit tests for conversion logic:

java

@Test

public void testConversion() {

assertEquals(220.462, converter.convertKilosToPounds(100), 0.001);

}

* + Espresso UI tests for input validation

1. **Error Handling**:
   * Show error icons for invalid input
   * Use TextInputLayout for advanced validation:

xml

<com.google.android.material.textfield.TextInputLayout

app:errorEnabled="true">

**Summary**

This section covered creating a kilogram-to-pound converter app. Key takeaways:

1. **Core Components**: EditText (input), Button (action), TextView (output)
2. **Data Handling**: String parsing, mathematical conversions
3. **Error Prevention**: Try-catch blocks for number formatting
4. **UI/UX**: Background images, text styling for readability

For production apps:

* Implement **input validation** and **error handling**
* Add **multi-unit support** using spinners
* Use **locale-aware formatting** for numbers
* Adopt **MVVM architecture** for maintainability
* Include **unit tests** for conversion logic
* Add **accessibility features** for inclusive design