

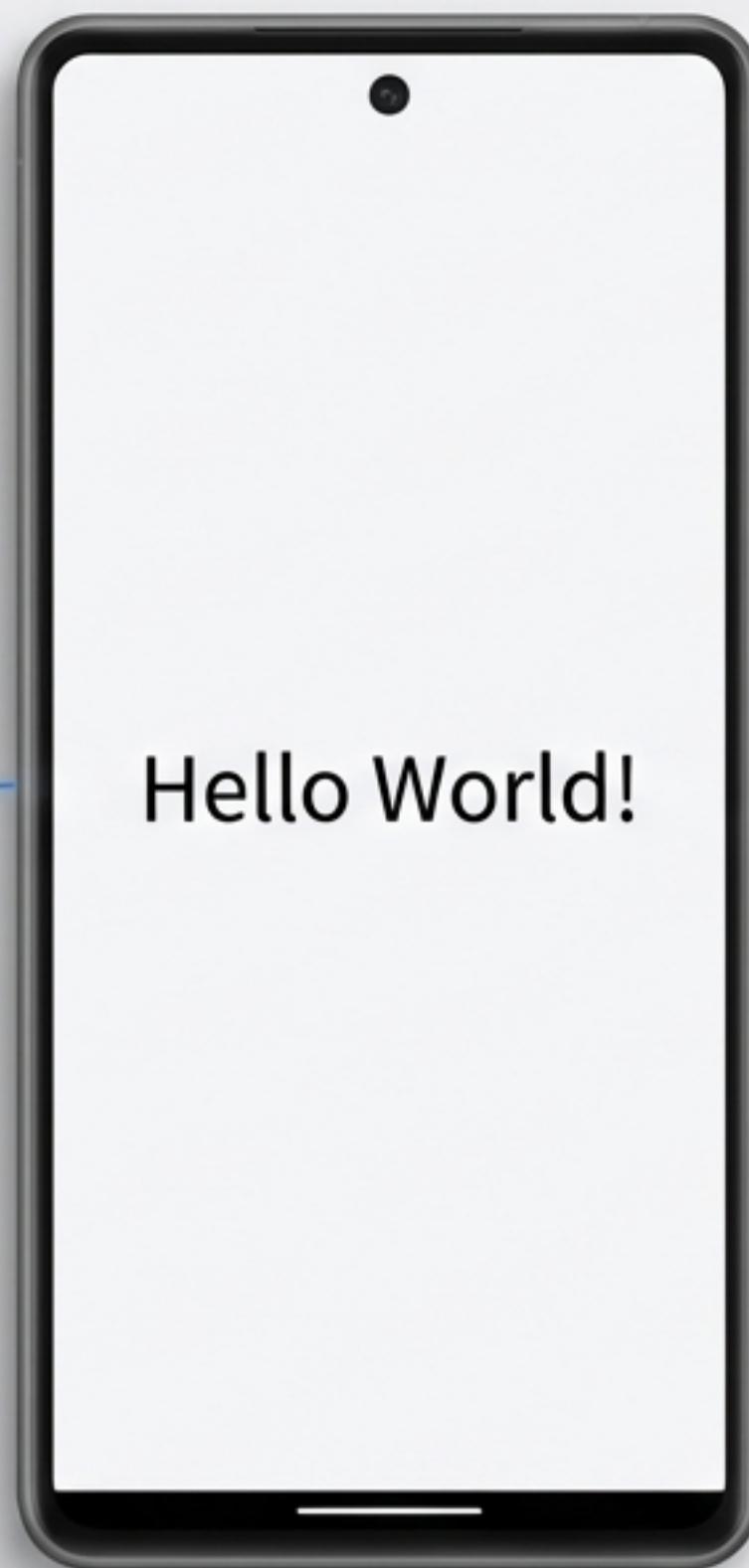
Android アプリ開発入門

Java ベースの基本構成とアーキテクチャ

```
package jp.ac.jec.hello;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



はじめに

Mission Statement

Hello World! を表示するシンプルなアプリを通じて、Android アプリの構成要素を理解します。

学習目標

- ✓ Android アプリのプロジェクト構成
- ✓ Activity の役割とライフサイクル
- ✓ XML レイアウトの基本
- ✓ アプリのビルドと実行

開発環境 (Environment Specs)

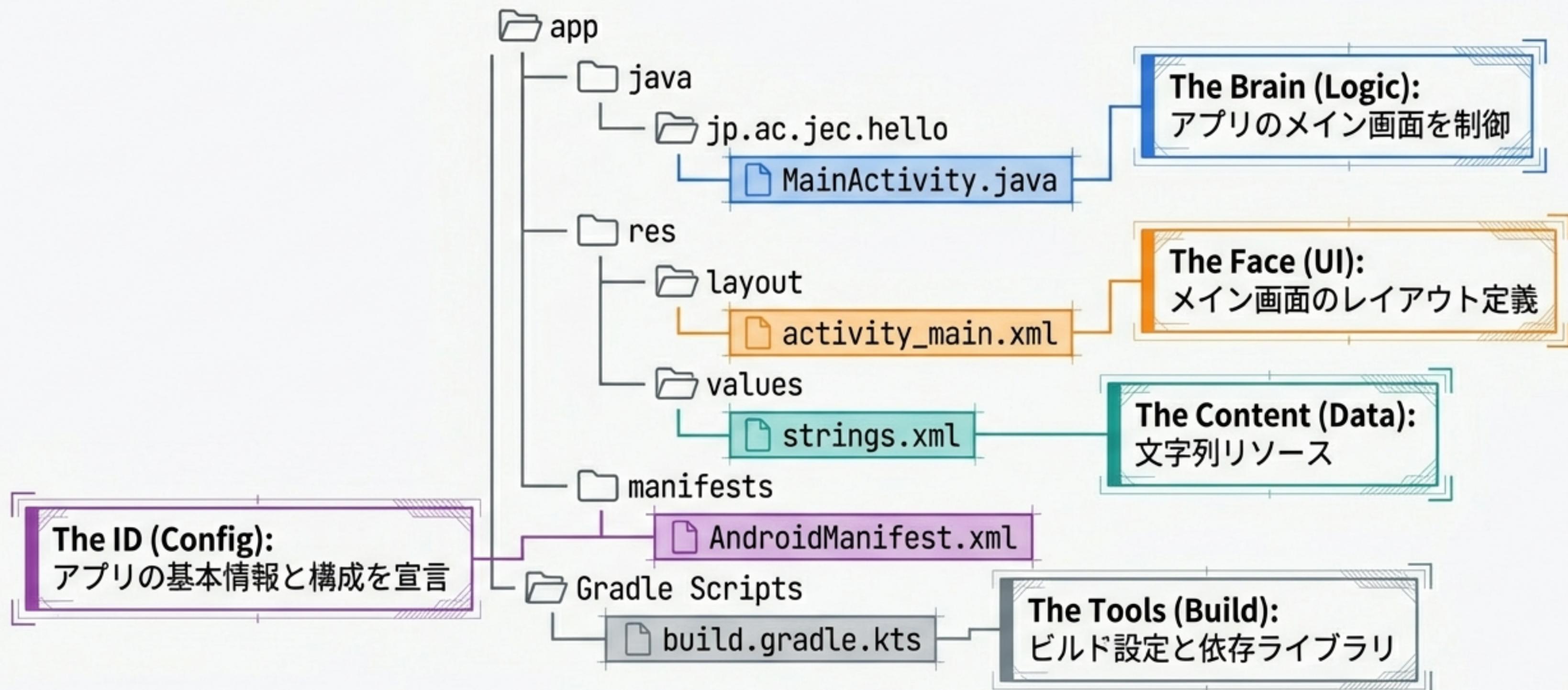
IDE Android Studio
Panda 1
(2025.3.1)+

Language JDK 11

Min SDK 30 (Android 11)

Target SDK 36 (Android 16)

プロジェクト構成



MainActivity.java: クラスの定義

```
package jp.ac.jec.hello;  
  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
    // ...  
}
```

Package Declaration
アプリを一意に識別するID。逆ドメイン形式(jp.ac.jec...)を使用。

Imports
必要なクラスの読み込み。

Inheritance (継承)
AppCompatActivity を継承することで、古いAndroidバージョンでも最新機能を使用可能にする(後方互換性)。

MainActivity.java: ライフサイクル



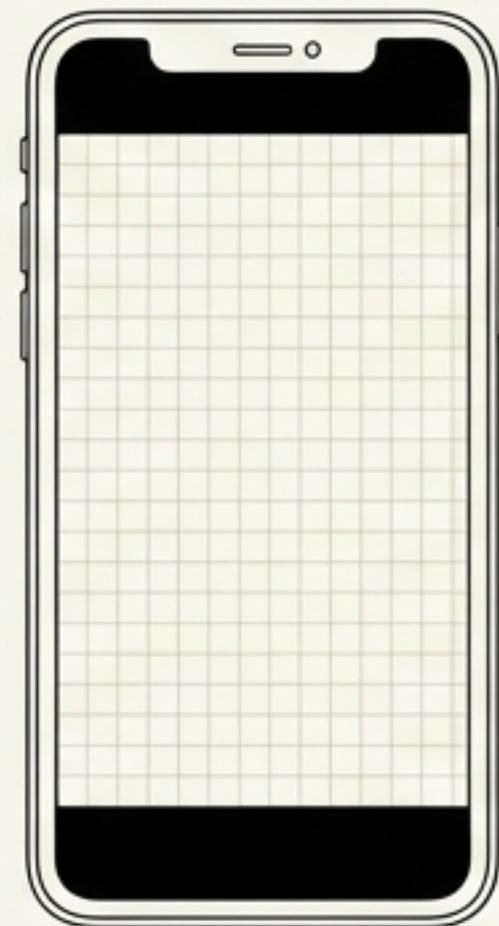
```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
}
```

親クラスの初期化処理 (必須)
レイアウトの設定
(Logic connects to UI)

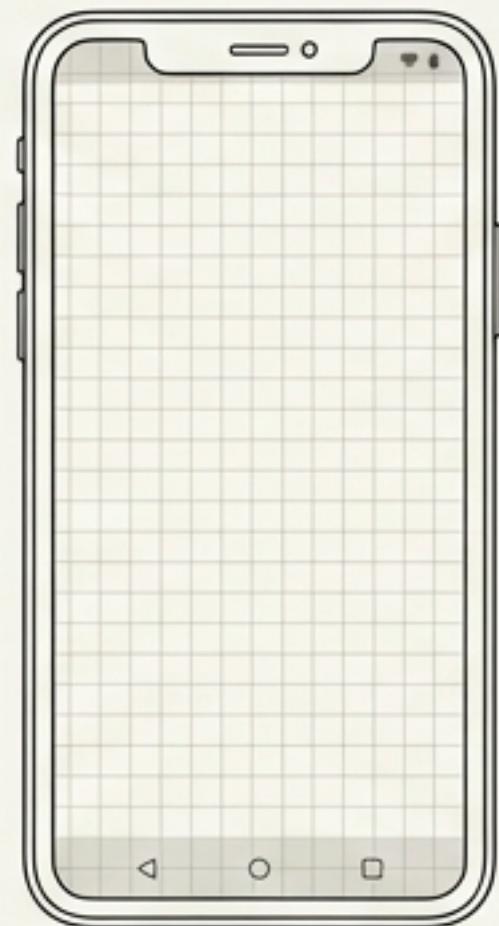


The Pulse

Modern UI: EdgeToEdge & Insets



Legacy Layout



Edge-to-Edge

```
enableEdgeToEdge();
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
    // システムバーの重なりを防ぐための余白設定
    Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
    return insets;
});
```

システムバーの領域分だけ、
内側に余白(Padding)を持たせる

レイアウトファイルの解説 (activity_main.xml)

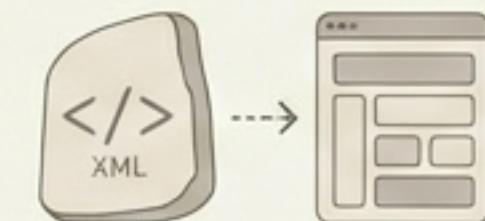
android: Android標準の属性。幅、高さ、テキストなど基本的な設定に使用。

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    ... >
```

android: Android標準の属性。幅、高さ、テキストなど基本的な設定に使用。

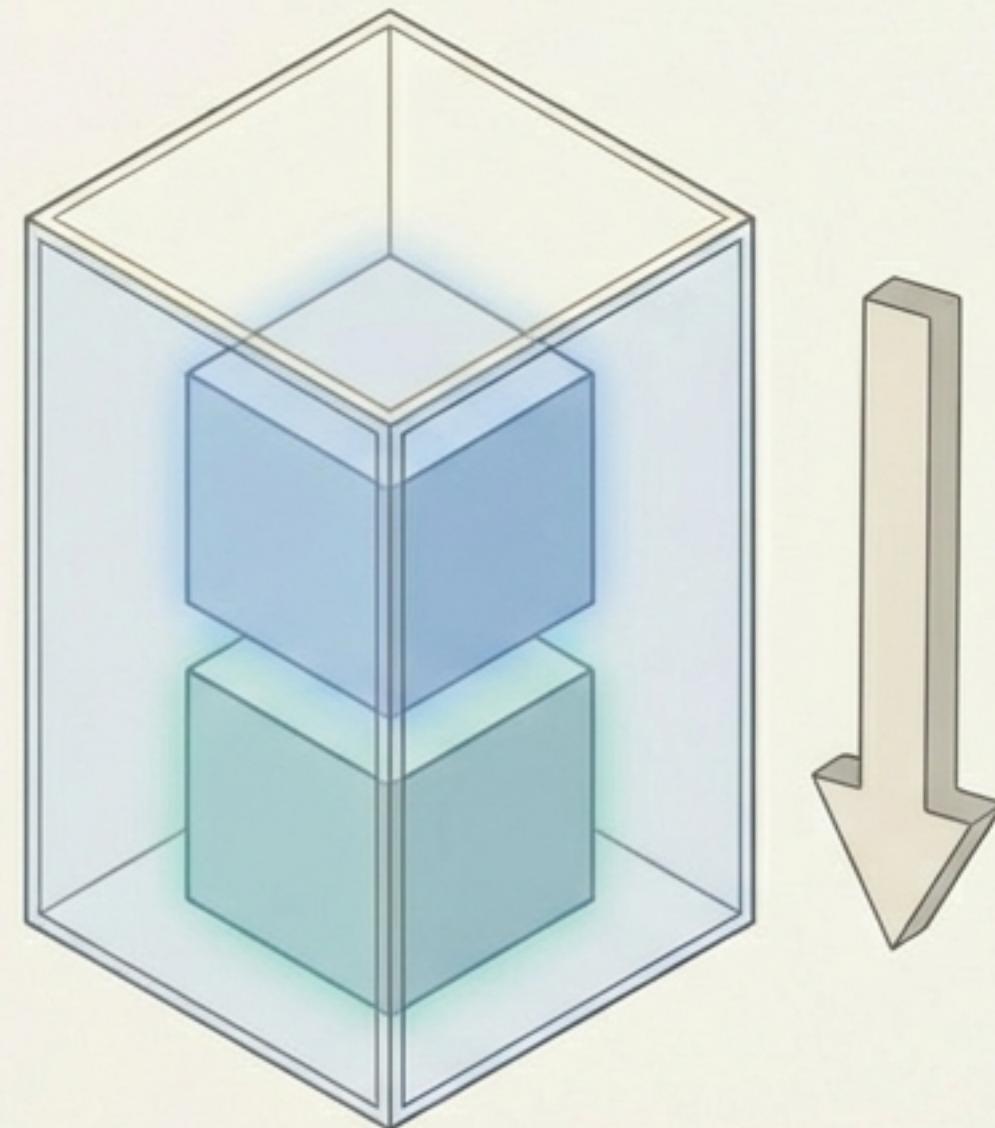
app: ライブラリ固有の属性。カスタム機能や互換ライブラリの設定に使用。

tools: 開発時のみ使用。プレビュー画面でのみ有効で、実行時には無視される。



Android speaks XML to draw the screen.

LinearLayout (The Container)



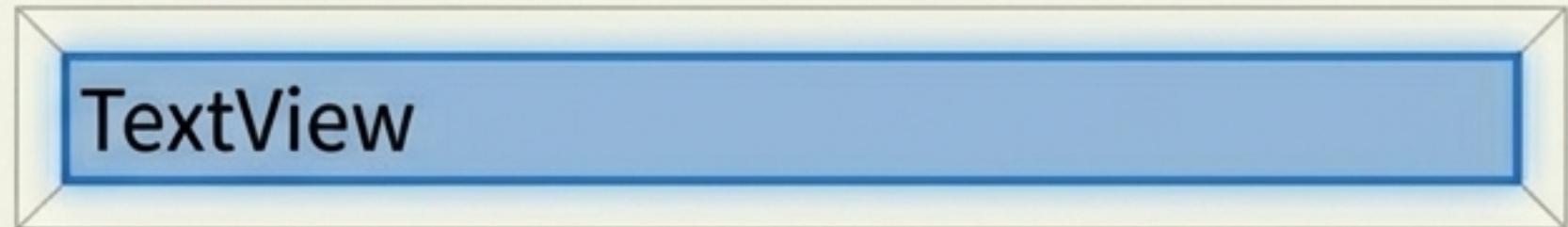
android:orientation="vertical"

```
<LinearLayout  
    android:id="@+id/main"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">
```

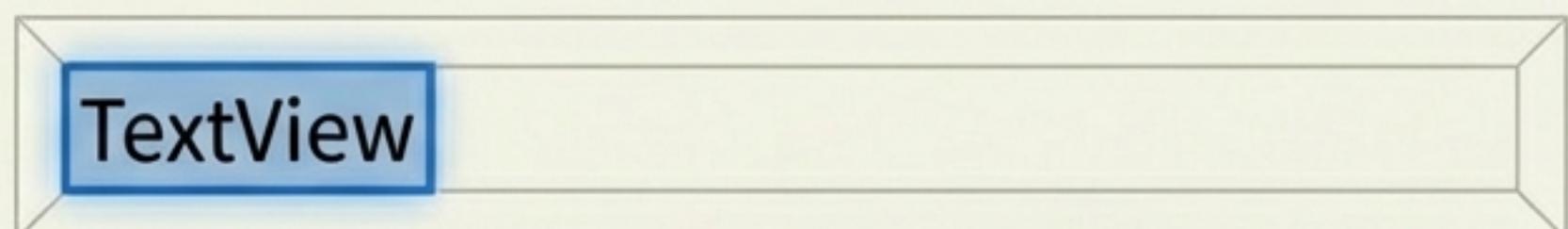
- **@+id/main** : "識別子 (ID) を新規作成"
- **match_parent** : "親要素 (画面全体) の幅・高さいっぱいに広がる"
- **vertical** : "子要素を上から下へ順番に配置"

TextView & Dimensions

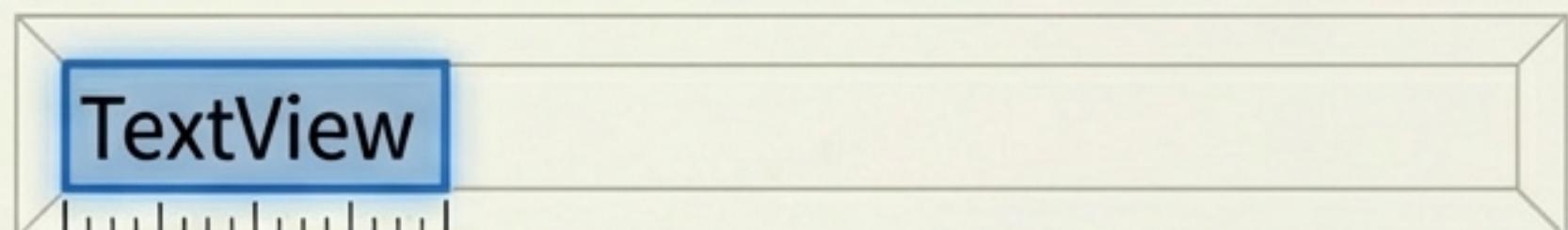
android:layout_width="match_parent"



android:layout_width="wrap_content"



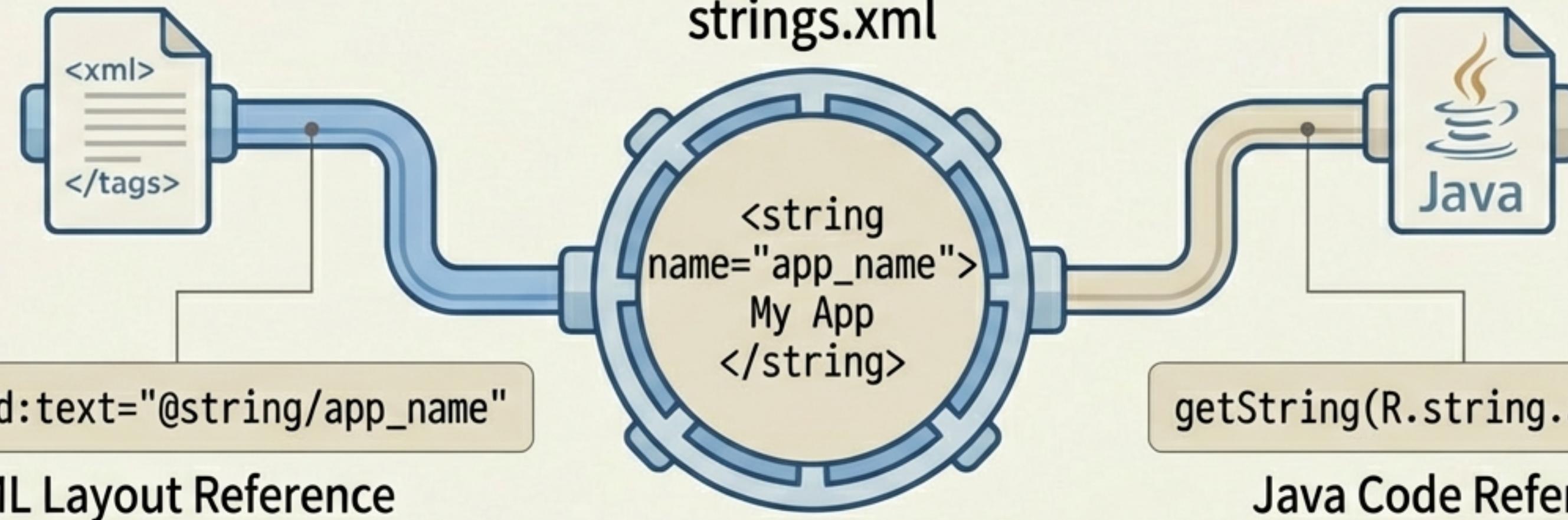
android:layout_width="100dp"



固定サイズ (**dp** = density-independent pixels)

dp guarantees consistent physical size across different screen densities.

リソースファイル: データとデザインの分離



Why Use This?

多言語対応 (Localization)

Switch languages easily.

一元管理 (Centralized)

Change text in one place, update everywhere.

No Hardcoding

Cleaner, maintainable code.

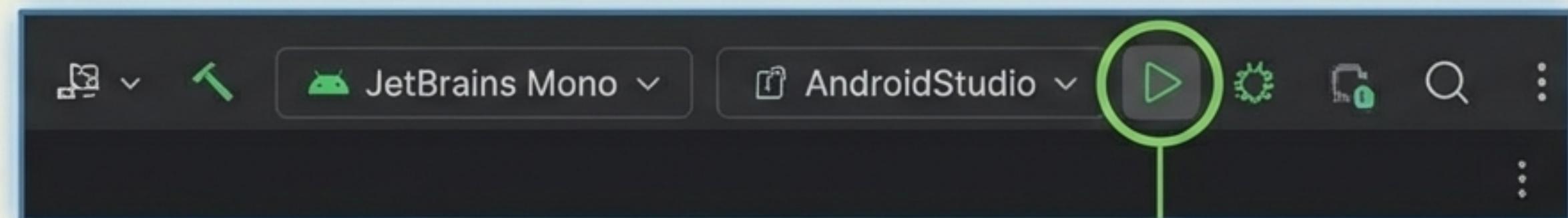
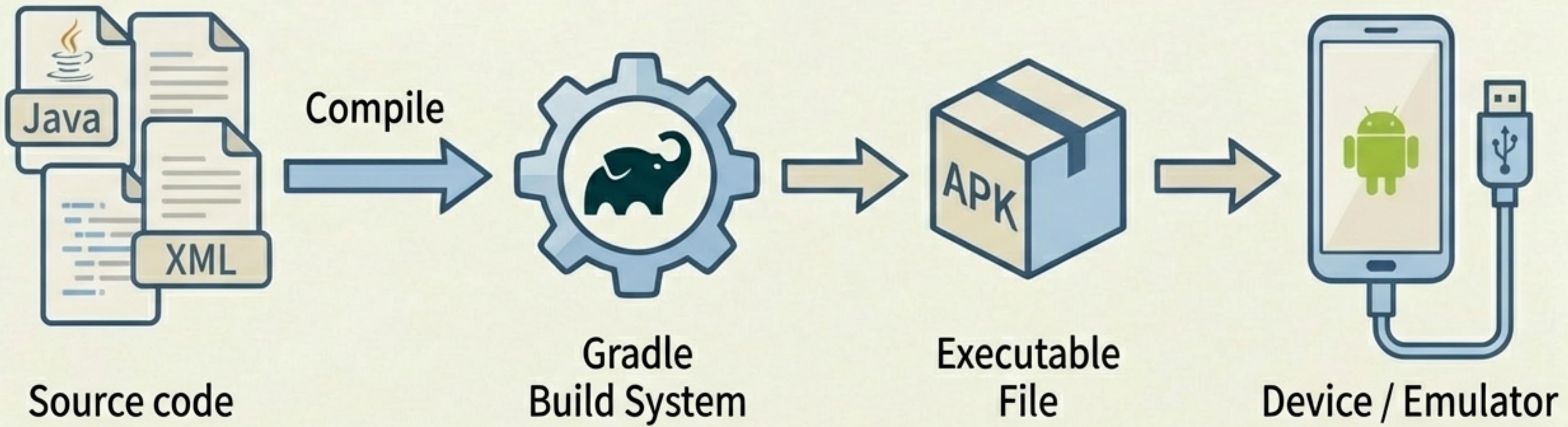
"AndroidManifest.xml"



```
<activity android:name=".MainActivity" android:exported="true">
    <intent-filter> ←
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

The Front Door: This tells Android that this Activity is the first screen to launch when the user taps the icon.

ビルドと実行 (Build & Run)



Click Run to start the process.

演習問題: UIの変更 (Challenge Level 1)

Goal: テキストを変更する

Exercise 1:

activity_main.xml を編集して、表示されるテキストを「こんにちは！」に変更してください。

Visual Hint:

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="こんにちは！" />
```

Goal: 文字列リソースを使用する

Exercise 2:

1. strings.xml に新しい文字列を追加。
2. TextView で @string/ を使って参照。

Visual Hint:

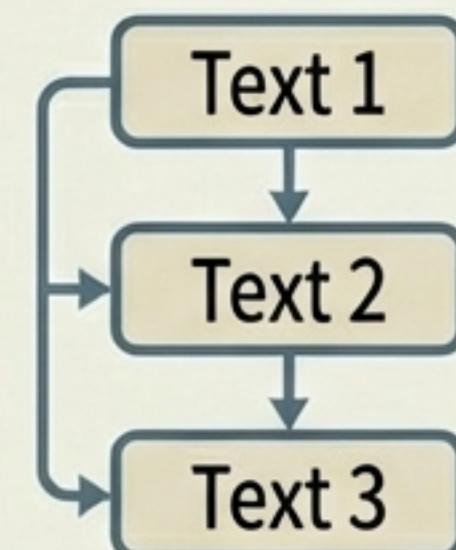
```
strings.xml:  
<resources>  
    <string name="hello">こんにちは！</string>  
</resources>
```

```
layout:  
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/hello" />
```

演習問題: 構造とロジック (Challenge Level 2)

Exercise 3: Goal: 複数の TextView

LinearLayout 内に 3 つの TextView を追加し、それぞれ異なるテキストを表示してください。



Cheat Sheet

```
TextView tv = findViewById(R.id.my_text);
tv.setText("Javaから変更しました");
```

Exercise 4: Goal: Java から操作する

TextView に ID を付け、Java コードでテキストを変更する。

1. XML: <TextView
 android:id="@+id/my_text" ... />
2. Java: findViewById(R.id.my_text)
3. Java: setText("New Text")

まとめ (Summary)

Recap

- **Project Structure:** The Skeleton (Directory Tree).
- **Activity Lifecycle:** The Brain (onCreate).
- **Layouts:** The Face (xml, 'LinearLayout').
- **Resources:** The Soul (strings.xml)
- **Build:** The Spark (APK Generation)



Next Steps

ボタンの追加やユーザー入力の処理 (Interaction)

You have built the foundation. Now you are ready to make it interactive.