

UNIT 2

Introduction to Android Programming

(4 Hours) — Mobile Programming (CAC351)

2.1 Android Platform

Android is an open-source, Linux-based mobile operating system and application platform developed by Google (originally by Android Inc.). It powers smartphones, tablets, TVs (Android TV), cars (Android Auto) and wearables (Wear OS). Apps are written mainly in **Kotlin** or **Java**, using XML for layouts, and are built with the Android SDK.

Android Software Stack

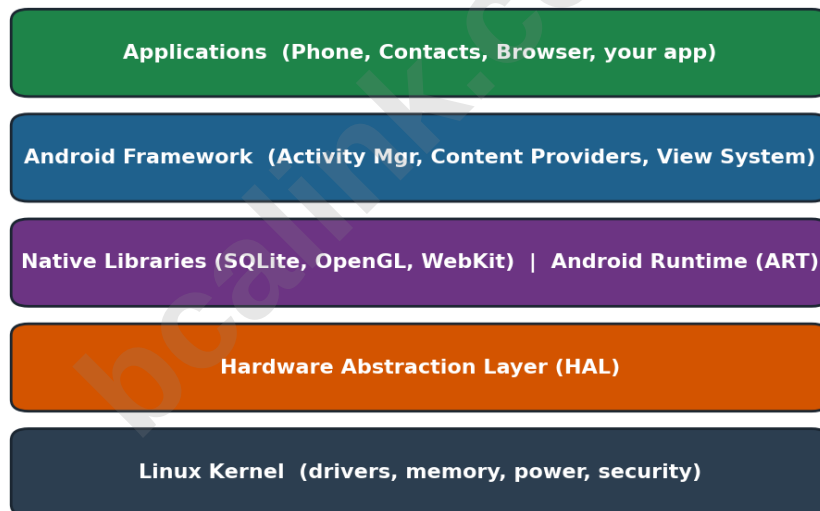


Figure 2.1: Layers of the Android platform (top to bottom)

Layer	Role
Linux Kernel	Process/memory management, security, power management and hardware drivers (camera, display, Wi-Fi).
HAL	Standard interfaces that expose hardware capabilities to the higher layers.
Native libraries and ART	C/C++ libraries (SQLite, OpenGL ES, WebKit) and the Android Runtime that executes compiled app code (DEX bytecode).
Application Framework	Java/Kotlin APIs: Activity Manager, Content Providers, Notification Manager, Resource Manager, View System.

Layer	Role
Applications	System and user-installed apps.

Main App Components

- **Activity** – a single screen with a user interface.
- **Service** – runs in the background without UI (e.g., music playback).
- **Broadcast Receiver** – responds to system-wide events (battery low, boot completed).
- **Content Provider** – shares app data with other apps (e.g., Contacts).
- **Intent** – message used to start components and pass data.

2.2 History of Android

Year	Event
2003	Android Inc. founded by Andy Rubin, Rich Miner, Nick Sears and Chris White.
2005	Google acquires Android Inc.
2007	Open Handset Alliance (OHA) formed; Android announced.
2008	Android 1.0 released with the HTC Dream (T-Mobile G1).
2009–2013	Named after desserts: Cupcake (1.5), Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat (4.4).
2014	Lollipop (5.0) introduces Material Design and the ART runtime.
2015–2018	Marshmallow (runtime permissions), Nougat, Oreo, Pie (9).
2019	Dessert names dropped; Android 10 onward use numbers (11, 12, 13, 14, 15...).
2017/2019	Kotlin declared an official language (2017) and the preferred language for Android (2019).

2.3 Environment Setup

The official IDE for Android is **Android Studio** (based on JetBrains IntelliJ IDEA). Steps to set up:

- Check requirements: 64-bit Windows/macOS/Linux, at least 8 GB RAM (16 GB recommended), ~8 GB free disk.
- Download and install Android Studio from developer.android.com/studio. The installer bundles the JDK (JetBrains Runtime).

- Run the Setup Wizard; it downloads the Android SDK, platform-tools and emulator.
- Open **Tools → SDK Manager** to install SDK platforms (API levels) and build tools.
- Open **Tools → Device Manager** to create an Android Virtual Device (AVD), or enable **Developer options → USB debugging** on a real phone.

Important SDK Terms

Term	Meaning
SDK	Software Development Kit – libraries, tools and docs for building Android apps.
API level	Integer identifying an Android version's framework API (e.g., API 34 = Android 14).
minSdk / targetSdk / compileSdk	Lowest supported version / version app is tested for / version used to compile.
Gradle	Build system that compiles code, manages dependencies and packages the APK/AAB.
APK / AAB	Installable package / Android App Bundle (required for Google Play).
AVD	Android Virtual Device – an emulated phone.

2.4 Creating an Android Project

- Choose **File → New → New Project**.
- Select a template, e.g., *Empty Views Activity* (XML layouts) or *Empty Activity* (Jetpack Compose).
- Enter Name (e.g., GeoQuiz), Package name (com.example.geoquiz), Save location, Language (Java/Kotlin), Minimum SDK.
- Click **Finish**; Gradle syncs and builds the project.

Project Structure

Folder / File	Purpose
app/src/main/java/...	Java/Kotlin source files such as MainActivity.
app/src/main/res/layout/	XML layout files (activity_main.xml).
res/values/	strings.xml, colors.xml, themes.xml.
res/drawable/, res/mipmap/	Images and launcher icons.

Folder / File	Purpose
AndroidManifest.xml	Declares package, components, permissions and the launcher activity.
build.gradle (app)	Module build settings: SDK versions and dependencies.

2.5 Laying Out the User Interface

The View Hierarchy

Everything on screen is a **View**. Widgets (buttons, text) are views; layouts are special views called **ViewGroup** that contain other views. The views form a tree called the **view hierarchy**, with a root layout at the top and widgets as leaves. Android draws the hierarchy from parent to children.

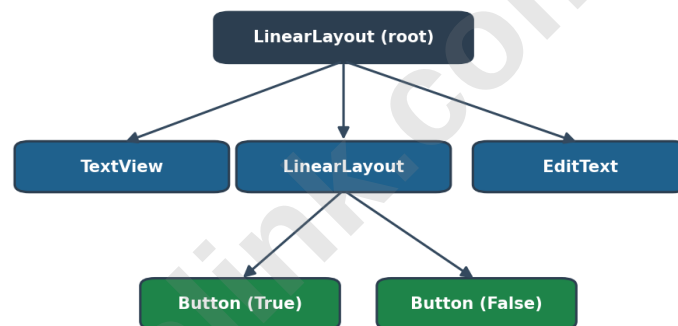


Figure 2.2: A sample view hierarchy for a quiz screen

Widget Attributes

Attributes in XML configure each view. Common ones:

Attribute	Meaning
android:id	Unique identifier, e.g., @+id/true_button; used from code.
android:layout_width / height	match_parent, wrap_content or a fixed size such as 100dp.
android:text	Text shown, normally a reference such as @string/true_button.
android:padding / layout_margin	Space inside / outside the view.
android:orientation	vertical or horizontal for LinearLayout.
android:textSize, textColor	Text appearance (use sp for text size).

Creating String Resources

Hard-coded text should be moved into **res/values/strings.xml** so that it can be reused and translated. Refer to it in XML with *@string/name* and in code with *R.string.name*.

```
<resources>
  <string name="app_name">GeoQuiz</string>
  <string name="question_text">Canberra is the capital of Australia.</string>
  <string name="true_button">True</string>
  <string name="false_button">False</string>
</resources>
```

Listing 2.1: strings.xml

Layout XML and Previewing

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:gravity="center"
  android:padding="16dp">

  <TextView
    android:id="@+id/question_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/question_text" />

  <Button
    android:id="@+id/true_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/true_button" />
</LinearLayout>
```

Listing 2.2: res/layout/activity_main.xml

Android Studio's layout editor offers three views: **Code** (XML), **Split** and **Design** (visual preview with palette and attributes panel). The preview lets you see the layout on different screen sizes without running the app.

Note: Use dp (density-independent pixels) for sizes and sp for text so that the UI scales on all screens.

2.6 Creating a New Class and Setting Up the Project

In the GeoQuiz example, a model class *Question* stores a question's text resource id and its answer. Create it with **File → New → Java Class**.

```

public class Question {
    private int mTextResId;
    private boolean mAnswerTrue;

    public Question(int textResId, boolean answerTrue) {
        mTextResId = textResId;
        mAnswerTrue = answerTrue;
    }
    public int getTextResId() { return mTextResId; }
    public boolean isAnswerTrue() { return mAnswerTrue; }
}

```

Listing 2.3: *Question.java* (model class)

Then wire the layout in *MainActivity*: *setContentView()* inflates the XML, and *findViewById()* gets references to widgets.

```

public class MainActivity extends AppCompatActivity {
    private Button mTrueButton;

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

        mTrueButton = findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(v ->
            Toast.makeText(MainActivity.this,
                R.string.correct_toast,
                Toast.LENGTH_SHORT).show());
    }
}

```

Listing 2.4: *MainActivity.java*

2.7 Running on the Emulator

- Create an AVD in Device Manager (choose hardware profile, system image/API level).
- Select the AVD in the toolbar device list and press **Run** ► (Shift+F10).
- Gradle builds the APK, the emulator boots, and the app is installed and launched.
- Use **Logcat** to read logs (*Log.d(TAG, "message")*) and the Debugger for breakpoints.
- To run on a real device: enable USB debugging, connect by USB or Wi-Fi, and select the device.

Example 1 — Fixing a build problem

The app crashes at start with *NullPointerException* on *mTrueButton*. What are the likely causes?

Solution

(1) *findViewById()* was called before *setContentView()*. (2) The id in code does not match *android:id* in the layout. Call *setContentView()* first and verify the ids.

Key Takeaways

- ✓ Android is a Linux-based open-source platform; its stack has Kernel, HAL, Libraries/ART, Framework and Apps.
- ✓ Android Studio + SDK + AVD form the development environment; Gradle builds the app.
- ✓ Layouts are XML view hierarchies; text belongs in string resources; use dp/sp units.
- ✓ `setContentView()` and `findViewById()` connect XML layouts with Java/Kotlin code.
- ✓ Run apps on the emulator (AVD) or a real device with USB debugging.

— *End of Unit 2* —

bcalink.com