Developer Guides

①

Build your first App

* Apps provide multiple entry point

⇒ Android app are built as a **Combination of Components** that can be invoked individually.

Example: **Activity** is a type of app component that provides UI.

⇒ The **"main" activity** starts when the user taps your app's icon.

↳ You can also direct the user to an activity from elsewhere, such as from a notification or even from a different app.

⇒ Other components such as **broadcast receivers** and **Services**, allow your app to perform background tasks without a UI.

* Android allows you to provide different resources for different devices

⇒ If any of your app's features need specific hardware, such as a camera, you can query at runtime whether the device has that hardware or not and then disable the corresponding features if it doesn't.

Main Activity . java

→ This is the entry point of your app

→ When you build and run your app, the System launches an instance of this Activity and load its layout.

activity_main . xml

→ Layout for the main activity's UI.

Android Manifest . xml

→ Describes the fundamental characteristics of the app and defines each of its Component.

build . gradle

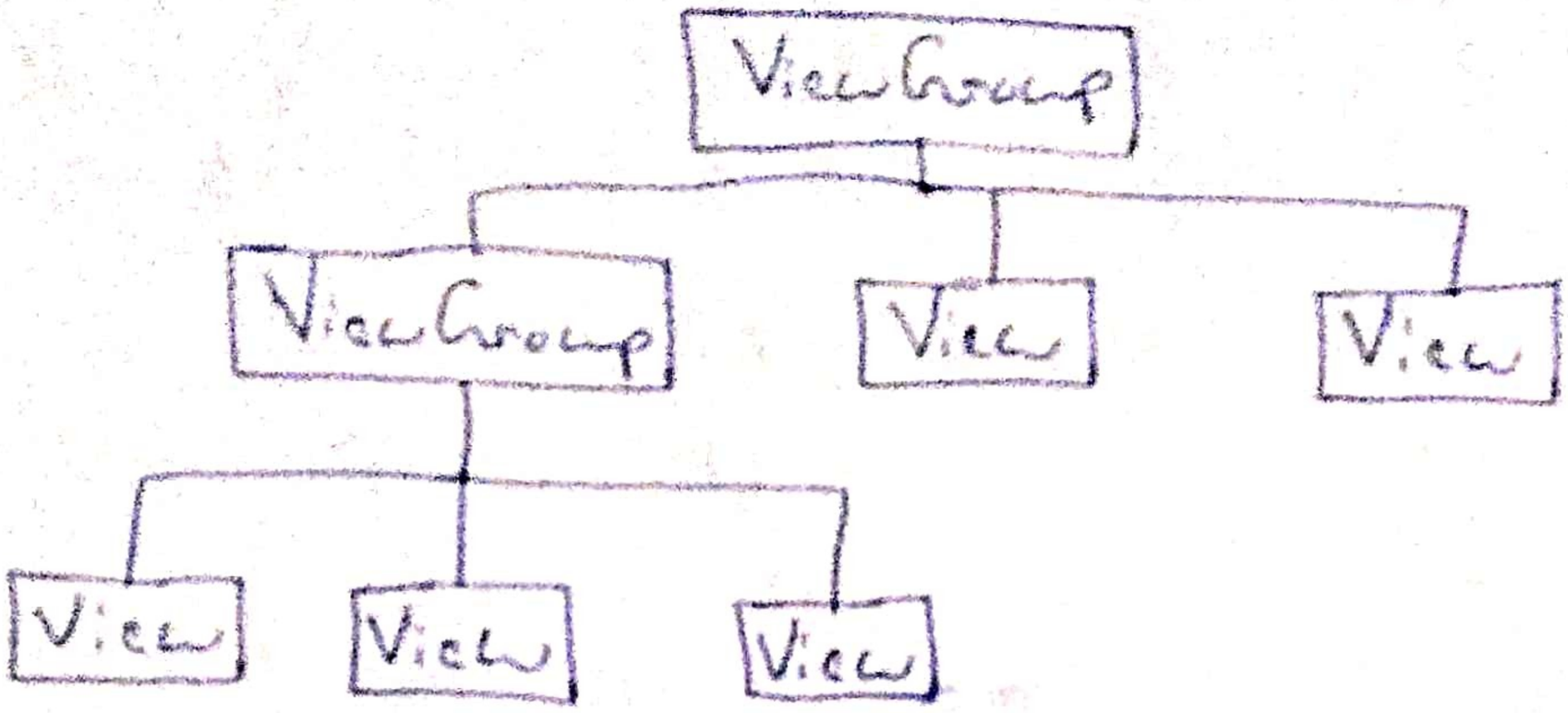
→ There are two files with this name
  → one for the project
  → one for the app module

→ Each module has its own build . gradle file.

⇒ The user Interface (UI) for an android app is built as a hierarchy of layouts and widgets.

→ The layouts are `ViewGroup` objects, Contains that controls how their child views are positioned on the Screen.

→ Widgets are view objects, UI Component such as buttons and text boxes.



⇒ Android provides an XML vocabulary for ViewGroup and View classes, so most of your UI is defined in XML files.

★ Intent

⇒ An Intent is an object that provides runtime binding between separate components, such as two activities.

⇒ The Intent represents an app intent to do something.

②

Application Fundamentals

⇒ Android apps can be written using Kotlin, Java, and C++ language.

⇒ The Android SDK tools compile your code along with any data & resource files into an APK.

↓

{ Android package, which is an archive file with an .apk suffix }

⇒ Each Android app lives in its own security sandbox, Protected by the following Android security features:

- → The Android operating system is a multi-user Linux system in which each app is a different user.
- → The System sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
- → Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.
- → By default, each app runs its own Linux process.

⇒ The Android system implements the principle of least privilege.

- → This is, each app by default, has access only to the components that it requires to do its work and no more.

⇒ However, there are ways for an app to share data with other apps and for an app to access system services.

- → It's possible to arrange for two apps to share the same Linux user ID, in which case they are able to access each other's files.
  - → To conserve system resources, apps with same user ID can also arrange to run in the same Linux process and share the same VM.
  - → The App must be signed with the same certificate.

*

* → An app can request permission to access device data such as the device's location, camera, and Bluetooth connection.

* App Components

⇒ App components are the essential building blocks of an Android app.

⇒ Each component is an entry point through which the system or a user can enter your app.

⇒ There are four different types of app components:

- → Activities
- → Services
- → Broadcast receivers
- → Content providers

Activities

- → An activity is the entry point for interacting with the user.
- → It represents a single screen with a user interface.

⇒ Although the activities work together to form a cohesive user experience, each one is independent of the other.

⇒ As such a different app can starting one of those activities if the app allows it.

⇒ You implement an activity as a subclass of the Activity class.

Services

→ A Service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons.

→ It is a Component that runs in the background to perform long-running operations.

Example: A Service may play music in the background.

⇒ Another Component, such as an activity, can bind to it in Order to interact with it.

⇒ A Service is implemented as a Subclass of Service.

Broadcast receivers

→ It is a Component that enables the System to deliver events to the app outside of a regular user flow, allowing the app to respond to System-wide broadcast announcements.

Example: App can schedule an alarm to post a notification to tell the user about an upcoming event.

⇒ Broadcasts can originate from the System
Example: Screen has turned off, battery is low etc.

⇒ App can also initiate broadcast:
Example: data has been downloaded to device.

⇒ Although broadcast receivers don't display a user interface, they may create a status bar notification.

⇒ A broadcast receiver is implemented as a subclass of BroadcastReceiver.

Content providers

↳ A content provider manages a shared set of app data that you can store in a file system,

↳ In a **SQLite database**

↳ On the web or any other persistent storage location that your app can access.

⇒ An app can decide how it wants to map the data it contains to a URI namespace.

↳ handing out those URIs to other entities which can in turn use them to access the data.

⇒ Assigning a URI doesn't mean that the app remain running, so URI can persist after their owning apps have exited.

⇒ Content providers are also useful for reading and writing data that is private to your app and not shared.

⇒ A Content Provider is implemented as a subclass of ContentProvider.

⇒ A unique aspect of the Android system design is that any app can start another app's Component.

⇒ When the System starts a Component, it starts the Process for that app if it's not already running and instantio the classes needed for the Component.

→ Unlike apps on most other systems, Android apps don't have a single entry point.
(There is no main() function)

⇒ To activate a Component in another app, deliver a message to the system that specifies your intent to start a particular Component.

↳ The System then activates the Component for you

★ Activating Components

⇒ Activities, Services and broadcast receivers are activated by an asynchronous message called an 'intent.

⇒ An 'intent is created with an Intent object.

⇒ Content providers are not activated by intent.

⇒ Rather, they are activated when targeted by a request from a (ContentResolver)

{The Content resolver handles all direct transactions with the Content Provider}

⇒ There are separate methods for activating each type of component:

* You can start an activity or give it something new to do by passing an Intent to startActivity() or startActivityForResult().

* With Android 5.0 and later, you can use the JobScheduler class to schedule actions.
  ↳ For earlier Android versions, you can start a Service by passing an Intent to startService().

* You can initiate a broadcast by passing an Intent to methods such as sendBroadcast(), sendOrderedBroadcast() or sendStickyBroadcast()

* You can perform a query to a content provider by calling query() on a ContentResolver.

★ The manifest file

⇒ Before the Android system can start an app component, the system must know that the component exists by reading the app's manifest file.

↓

AndroidManifest.xml .

⇒ Your app must declare all its components, in this file, which must be at the root of the app project directory.

⇒ The manifest does a number of things in addition to declaring the app's components, such as the following:

- Identifies any user permissions that app requires, such as Internet access or read-access to the user's contacts..
- Declares the minimum API Level required by the app, based on which APIs the app uses.
- Declares hardware and software features used or required by the app, such as a camera, bluetooth services, or a multitouch screen.
- Declares API Libraries the app needs to be linked against, such as the Google Maps Library.

* Declaring Components

- <application> element
  - android:icon attribute
    {Points to resource for an icon that identifies the app}
  - <activity> element
    - android:name attribute
      {Attribute specifies the fully qualified class name of the Activity Subclass}
    - android:label attribute
      {Specifies a string to use as the user-visible label for the activity}

⇒ You must declare all app components using the following elements:

* <activity> element for activities
* <Service> element for services
* <receiver> element for broadcast receivers.
* <provider> element for Content providers.

⇒ Activities, services and content providers that you include in your source but don't declare in the manifest are not visible to the System and, consequently, can never run.

* Declaring Component Capabilities

⇒ You can use an Intent to start activities, Services, and broadcast receivers.

⇒ You can use an Intent by explicitly naming the target component (using the component class name) in the intent.

⇒ You can also use an implicit intent, which describes the type of action to perform and, optionally, the data upon which you'd like to perform the action.

→ The implicit intent allows the system to find a component on the device that can perform the action and start it.

→ If there are multiple components that can perform the action described by the intent, the user selects which one to use.

⇒ The System identifies the Components that can respond to an intent by Comparing the intent received to the [intent filters] provided in the manifest file of other apps on the device.

⇒ When you declare an activity in your app's manifest, you can optionally include intent filters and declare the Capabilities of the activity so it can respond to intents from other apps.

★ Declaring App requirement

⇒ There are a variety of devices powered by Android and not all of them provides the same features and Capabilities.

⇒ To prevent your app from being installed on devices that lack features needed by your app, its important that you clearly define a profile for the type of device your app supports by declaring device and Software requirements in your manifest file.

★ App resources

⇒ An Android app is Composed of more than just code - it requires resources that are Seperate from the Source code, such as image, audio file etc..

⇒ For every resource that you include in your Android Project, the SDK build tools define a unique integer ID:

↳ which you can use to reference the resource from your app code or from other resources defined in XML.