Introduction to Android

Topics Covered

Overview for Android

- ✓ Definition of android
- ✓ Why Android
- ✓ History of Android
- ✓ Features of Android

Android Architecture

Environment Setup

- ✓ Installing Android Studio
- ✓ Test AVD
- ✓ Create Simple Android App

Android Core Building Blocks

Android Project Structure

What is Android?

- is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as smart phones and tablets.
- Offers a unified approach to application development for mobile devices.
- Started as an operating system for phones, but it has since made its way into all sorts of places like TVs, Car systems, Watches, E-readers, netbooks, and game consoles etc.
- It’s a powerful development framework that includes everything you need to build great apps using a mix of Java/Kotlin and XML.

- **API level**: is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.
- Each Android platform version supports exactly one API Level.
- The Android platform provides a framework API that applications can use to interact with the underlying Android system.
- The Framework API consist:
  - ✓ A core set of packages and classes
  - ✓ A set of XML elements and attributes for declaring a manifest file.
  - ✓ A set of XML elements and attributes for declaring and accessing resources.
  - ✓ A set of intents.
  - ✓ A set of permissions that application can request, as well as permission enforcements included in the system.

Cont'd

- Applications can use a manifest element provided by the framework API, <uses-sdk> to describe the minimum and maximum API levels under which they are able to run, as well as the preferred API level that they are designed to support.

- <uses-sdk> offers three key attributes:
  - ❖ android:minSdkVersion: Specifies the minimum API level on which the application is able to run. The default value is "1".
    - ✓ It is a hard floor below which the OS should refuse to install the app.
  - ❖ android:targetSdkVersion: Specifies the API level on which the application is designed to run. It indicates how your app is meant to run on different Android versions.
    - ✓ Targeting a recent API level ensures that users benefit from security, privacy and performance improvements while still allowing an app to run on older versions.

- ❖android:maxSdkVersion: Specifies the maximum API level on which the application is able to run.

N.B Declaring maxSdkVersion this attribute is not recommended because after the system update the app may crash.

Compile SDK Version (build target): This setting is not used to update the AndroidManifest.xml file. Whereas the minimum and target SDK versions are placed in the manifest when you build your app to advertise those values to the OS, the compile SDK version is private information between you and the compiler.

- ✓Specifies which version to use when building your own code. the classes and methods you refer to in your imports, will be checked against the build target defined in the Gradle file.

Why Android?

- Is the most popular mobile operating system, with more than 2.5 billion devices activated and it offers a unified approach to application development for mobile devices.
- "unified" meaning developers need only develop for android and their application will be able to run on different devices powered by android.
- According to 2022 report android takes around 70 % of the overall market share worldwide.
- It is Open Source. Google has made the code for all the low-level "stuff" as well as the needed middleware to power and use an electronic device, and gave Android freely to anyone who wants to write code and build the operating system from it.
- Larger developer and community reach.

History of Android

Platform Version	Version Code	API Level
1.5	Cupcake	3
1.6	Donut	4
2.0-2.1	Éclair	5-7
2.2.x	Froyo	8
2.3-2.3.4	Gingerbread	9-10
3.0-3.2	Honeycomb	11-13
4.0-4.0.4	Ice Cream Sandwich	14–15
4.1-4.3	Jelly Bean	16–18
4.4	KitKat	19–20
5.0–5.1	Lollipop	21-22
6.0	Marshmallow	23
7.0	Nougat	24
7.1-7.1.2	Nougat	25
8.0	Oreo	26
8.1	Oreo	27
9	Pie	28
10	Q	29
11	R	30
12	S	31-32
13	Tiramisu	33

- Initially, Andy Rubin founded Android Incorporation in Palo Alto, California, United States in October, 2003.
- In 17th Aug 2005 Google bought Android Inc.
- In 2007, Google announces the development of android OS.
- In 2008, HTC launched the first android mobile
- The Table shows the different Android Versions and their corresponding API level.

Version 1.5 Cupcake & Version 1.6 Donut

An on-screen keyboard, Extensible widgets,

Multi touch capability, Support for front-facing cameras, Screen PIN protection.

Version 3.x

Targeted exclusively at tablets, No physical buttons, Improved Multitasking

Version 4.0 Ice cream Sandwich

NFC Support, Face unlock, Data usage analysis

Version 4.1 to 4.3 Jelly

Support panoramic image, projecting text, support OpenGL ES 3.0

Version 4.4 KitKat

Full screen apps (Immerse yourself), Google Cloud print, Improved Quickoffice app

Version 5.0 – 5.1 Lollipop (SDK 22)

Device Protection- Smarter Wi-Fi improved Priority mode – changing the volume. Easier way to connect Wi-Fi, Bluetooth – Alarm volume is easier to change. …

Version 6.0 Marshmallow (SDK 23)

Android Pay -Adopted Storage -USB Type-C - System UI Tuner -Improved Copy and Pasting - Custom Google tabs - Clear permissions system.

Version 7.0 Nougat (SDK 24)

Split-Screen Mode -split-screen multitasking - More Powerful Notifications. …

A Better Doze for Longer Battery Life - An Easier, More Customizable Quick Settings Menu. ... - Data Saver, Call Blocking, and More.

Version 8.0 – 8.1 Oreo

Picture-in-Picture -Password Autofill - Notification Channels -Snooze Notifications. Notification Dots -New Emoji Styling -Smart Text Selection.

Auto-Enable Wi-Fi.

Version 9.0

Enhance messaging experience and Notifications ….

…..

You can read more from the official android documentation.

Features of Android

- Beautiful UI

Android OS basic screen provides a beautiful and intuitive user interface.

- Connectivity

Along with some standard and basic network connectivity, Android provides Bluetooth, Wi-Fi, NFC, P2P, SIP and USB to interact and connect with other devices and WiMAX.

- Storage

SQLite, a lightweight relational database, is used for data storage purposes.

- Wi-fi Direct

This technology is used to connect devices directly to each other via Wi-Fi without an intermediate access point.

Cont’d

- Messaging

Android OS provides SMS (Short Message Service) and MMS (Multimedia Messaging Service) to send short text messages, videos, photos or audios to other devices.

- Multi-tasking

This operating system comes with a feature of multi-tasking so that the users can use multiple applications simultaneously at the same time.

- Android Beam

With the Android Beam file transfer feature, you can share small or large files to other devices. This is a NFC-based technology with lets you share files just by touching the two NFC-enabled devices together.

- Firebase Cloud Messaging (FCM)

With FCM you can send notification messages or data messages, send messages from client apps and distribute messages to your client app to single devices/ to groups of devices.

Android Architecture

- First Let discuss about Operating System.

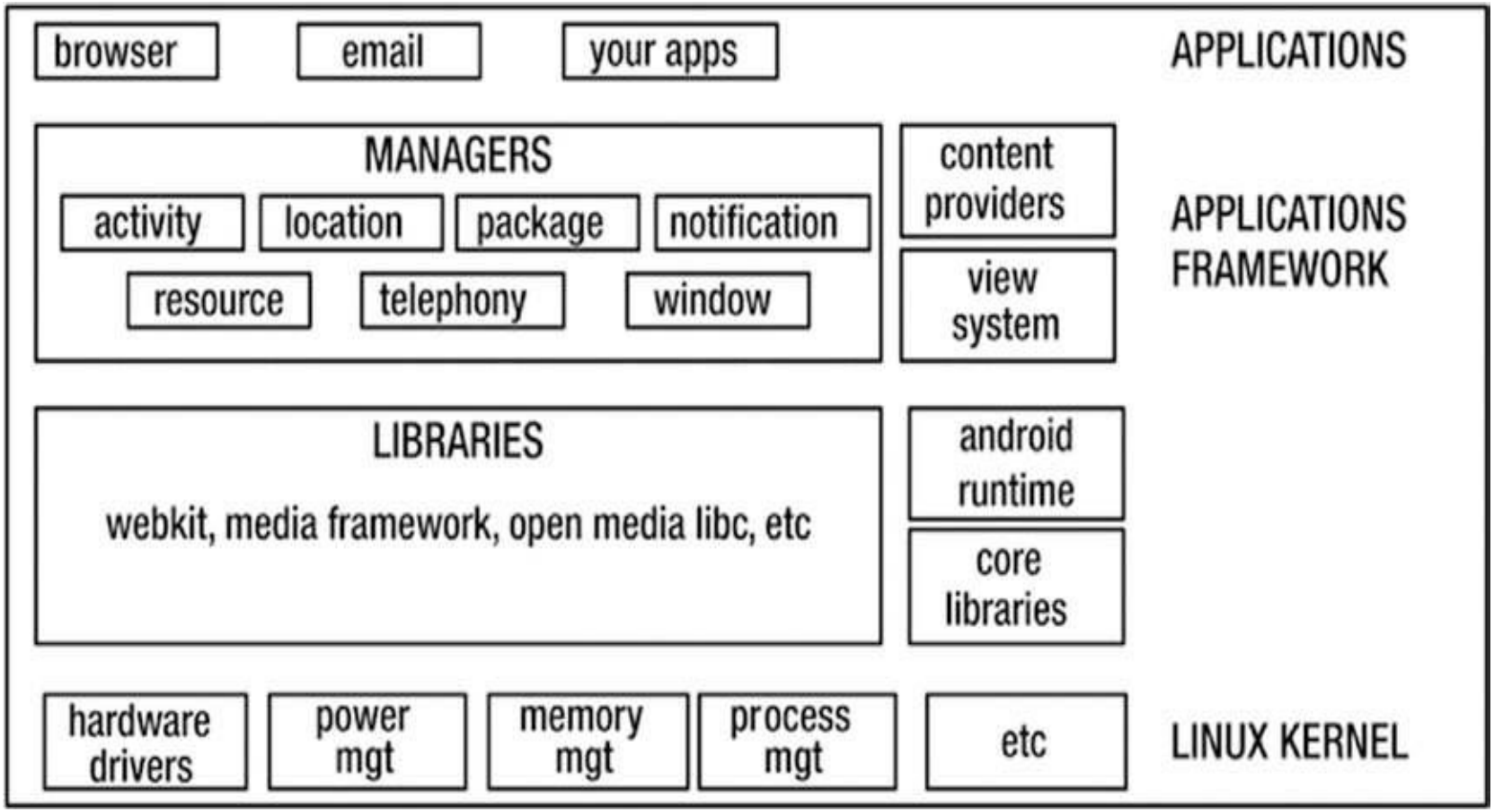
Operating System: The most visible part of Android, at least for developers, is its operating system.

- ❖ It stands between the user and hardware. By “User” not only indicate end user or person rather it can refer to an application, a piece of code that a programmer creates; like a word processor or an email client.
- ❖ For instance, Take the **email app**,

  As we type each character, the app needs to communicate to the hardware for the message to make its way to your screen and hard drive and eventually send it to the cloud via your network.

- Generally, Operating System does three things:
  - ❖ Manages hardware on behalf of applications.
  - ❖ Provides services to applications like networking, security, memory management, and so forth.
  - ❖ Manages execution of applications; this is the part that allows us to run multiple applications (seemingly) almost at the same time.

- Next, we try to see the High-Level Android System architecture, it consists of five parts.
  - ❖ Linux Kernel
  - ❖ Native Library (C/C++ libraries)
  - ❖ Android Runtime
  - ❖ Application Framework
  - ❖ Applications


browser
email
your apps
APPLICATIONS
MANAGERS
activity
location
package
notification
resource
telephony
window
content providers
view system
APPLICATIONS FRAMEWORK
LIBRARIES
webkit, media framework, open media libc, etc
android runtime
core libraries
hardware drivers
power mgt
memory mgt
process mgt
etc
LINUX KERNEL

- Linux Kernel: is the lowest layer in the diagram responsible for interfacing with the hardware, various services like memory management, power management and executions of processes. It provides a level of abstraction between the device hardware and the upper layers of the Android software stack.

  Linux is very stable OS and is quite ubiquitous itself. we can find it in many places like server hardware on data centers, appliances, medical devices, and so forth.

❖ It is the heart of android architecture that exists at the root of android architecture.

- Native Libraries: on the top of Linux Kernel low-level libraries like SQLite, OpenGL, WebKit, FreeType, and so on. These are not part of the Linux kernel but are still low level and as such are written mostly in C/C++.

  Webkit Library: For Browser Support

  SQLite: For Database Support

  FreeType: For Font Support

The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps.

Android Runtime:

- The Android Runtime (ART) is the system that runs your compiled code on an Android device. It first appeared on Android with the release of KitKat and became the standard way of running code in Lollipop.
- It is the successor of DVM (Dalvik virtual machine).
- Dalvik is a JIT (Just in time) compilation-based engine i.e., instead of compiling the whole app to machine code it compiles small chunk (part of) code during runtime.
- The advantage is it will require low memory usage.
- ART is equipped with Ahead-of-Time compiler (AOT) i.e. During the app's installation phase, it statically translates the DEX bytecode into machine code and stores in the device's storage. This format is known as Executable and Linkable Format (ELF).

❖ The advantage

- ✓ Apps run faster as DEX bytecode translation done during installation.
- ✓ Reduces startup time of applications as native code is directly executed.
- ✓ Improves battery performance as power utilized to interpreted byte codes line by line is saved.

Application Framework

❖ It sits on top of both the low-level libraries and the Android Runtime because it needs both. This is the layer that we will interact with as an application developer because it contains all the libraries, we need to write apps.

❖ It includes **Java API's** such as View System, Content Provider, Resource Manager, Activity Manager, Package Manager etc. It provides a lot of classes and interfaces for android application development.

Applications:

- On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using Linux kernel.

Android- Environment Setup

- In order to write an Android application, we are going to need a **development environment**.
- Google has made a very useful tool for all Android Developers, the Android Studio.
- Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on JetBrains' IntelliJ. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:
  - A flexible Gradle-based build system
  - A fast and feature-rich emulator
  - A unified environment where you can develop for all Android devices etc.

System Requirement

- Microsoft Windows 7, 8, or 10 (32- or 64-bit)
- macOS 10.10 (Yosemite or higher)
- Linux (Gnome or KDE Desktop), Ubuntu 14.04 or higher; 64-bit capable of running 32-bit applications
  - ✓GNU C Library (glibc 2.19 or later) if you're on Linux

Hardware Requirement

For the hardware, your workstation needs to be at least

- 4GB RAM minimum (8GB or more recommended)
- 2GB of available HDD space (4GB is recommended)
- 1280*800 minimum screen resolution.

Install Android Studio

- Go to the official android website

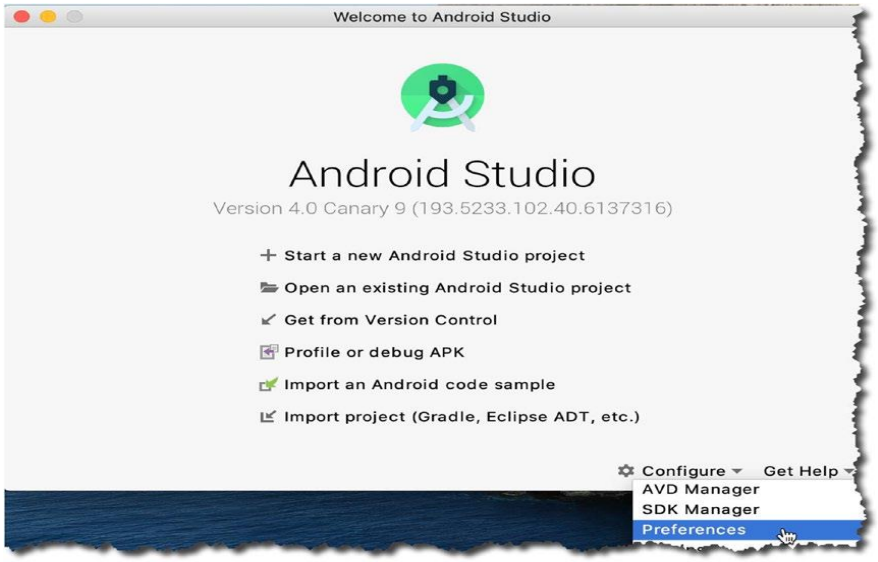
https://developer.android.com/studio/

If you're on Windows, do the following:

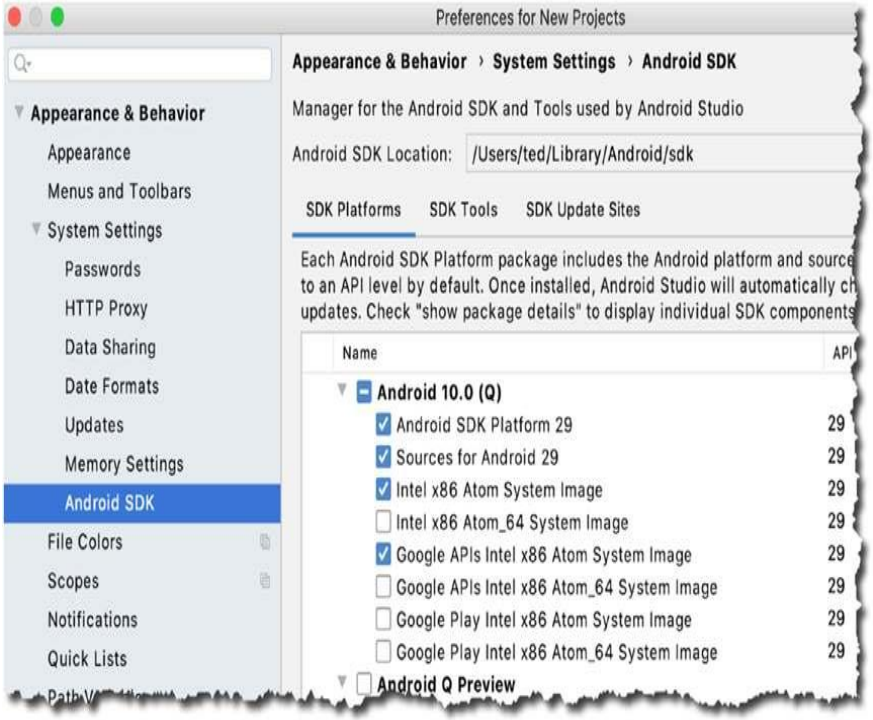
- Unzip the installer file.
- Move the unzipped directory to a location of your choice, for example, C:\Users\myname\AndroidStudio.
- Drill down to the "AndroidStudio" folder; inside it, you'll find "studio64.exe". This is the file you need to launch. It's a good idea to create a shortcut for this file—if you right-click studio64.exe and choose "Pin to Start Menu," you can make Android Studio available from the Windows Start menu; alternatively, you can also pin it to the Taskbar.

Configuring Android Studio

- Launch android studio and click "Configure", then choose "preferences" from the drop-down list.



- The "Preferences" option opens the Preferences dialog. On the left-hand side, go to Appearance & Behavior System Settings ⇨ Android SDK
- Download the API Level you want to target for your application. In this case Android 10 is used.



- Make sure you install the following tools in SDK Tools:
  - ✓ Android SDK Build tools
  - ✓ Android SDK Platform tools
  - ✓ Android SDK tools
  - ✓ Android Emulator
  - ✓ Support Repository
  - ✓ HAXM Installer (Hardware Acceleration)

Hardware Acceleration

- An Android Virtual Device or AVD is an emulator where you can run your apps.
- Running on an emulator can sometimes be slow; this is the reason why Google and Intel came up with HAXM.
- HAXM It is an emulator acceleration tool that makes testing your app a bit faster.

N.B If you are on the Linux platform, you cannot use HAXM even if you have an Intel processor. KVM (Kernel-based Virtual Machine) will be used in Linux instead of HAXM.

Test Our App

- We can use
  - ✓ Android Emulator
  - ✓ Physical device (our phone)
    - Android Emulator: allows us to run our app on an Android virtual device (AVD), which behaves just like a physical device.

AVD running in the Android emulator looks like the above figure.
The AVD here is setup to Nexus 5x and API level 25.

- ❖ If we want to know how the app behaves in our physical device.

✓Enable USB debugging on your device

On your device, open “Developer options” [greater than 4.0 version]

- ▪ To enable “Developer options,”

go to Settings → About Phone and tap the build number seven times

- • Turn on USB debugging

✓Setup your system to detect your device

Install a USB driver

- ✓ Plug your device into your computer with a USB cable
  - Your device may ask you if you want to accept an RSA key that allows USB debugging with your computer. If it does, you can check the “Always allow from this computer” option and choose OK to enable this.

Android Core Building Blocks

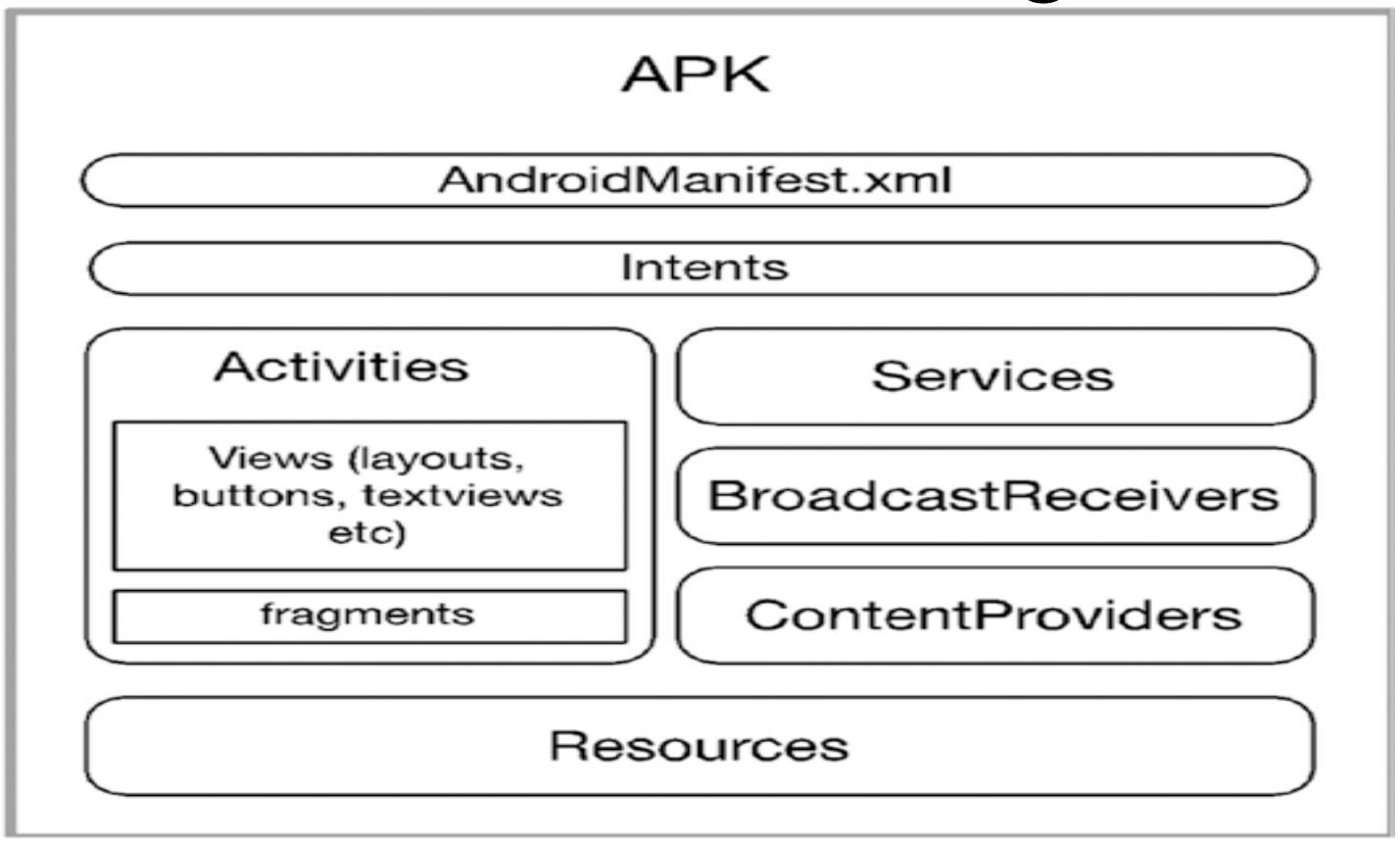


Fig. Logical Representation of Android App

Cont'd

- The Key building block of Android:
  - ❖Activity
  - ❖Services
  - ❖Broadcast Receivers
  - ❖Content Providers

Activity:

❖ Specifies what the app does and how it should interact with the user. E.g., An activity can be made so a user can Dial the phone, take a photo, send email or it can View a photo.

❖ If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

Services:

- ❖ They handle background processing associated with an application. The Program can be run without freezing the user interface.
- ❖ A service doesn't provide a User interface.

E.g., we can use Services when our app is supposed to download a file from the web or maybe play music.

Broadcast Receivers:

- ❖ respond to broadcast messages from other applications or from the system.
- ❖ you want your app to react in some way when a system event occurs.

A simple scenario can be having built a music app, and you want it to stop playing music if the headphones are removed. How can your app tell when these events occur?

Events example: device running low on power, a new incoming phone call, or the system getting booted. You can listen for these events by creating a broadcast receiver.

Content Providers:

- A content provider component supplies data from one application to others on request.
- Interface that allows apps to share data in a controlled way.
- It allows you to perform queries to read the data, insert new records, and update or delete existing records.

Using Intent, you can forward data in an app. For example, Messaging app to send the text you pass to it. But what if you want to use another app's data in your own app?

For example, what if you want to use Contacts data in your app to perform some tasks, or insert a new Calendar event?

You can't access another app's data by interrogating its database,

Instead, you use a content provider.

Additional Components

Intent:

- refers the message that is passed between components such as activities, content providers, broadcast receivers, services etc.
- Intent meaning is intention or purpose. it can be described as the intention to do action.

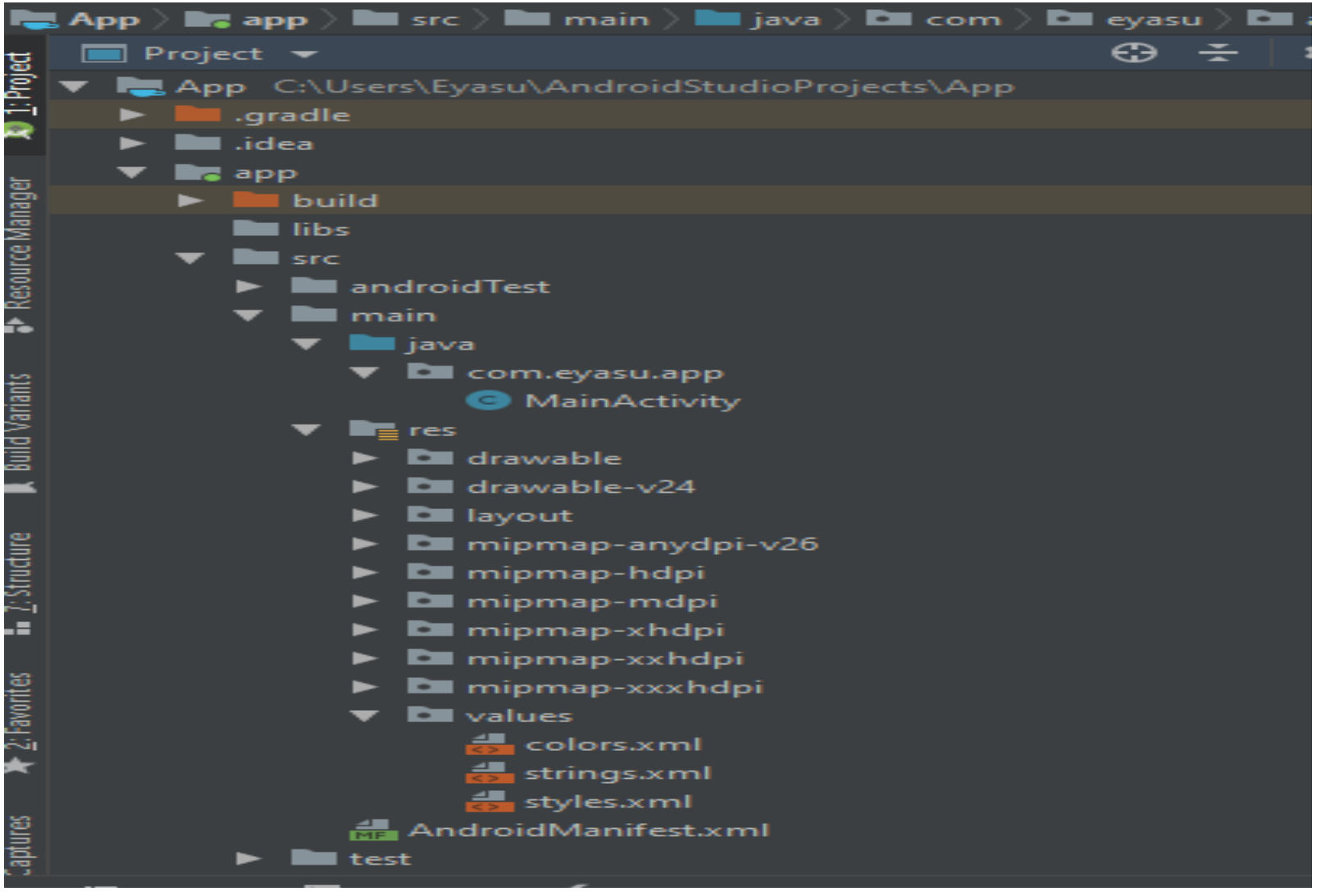
Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a webpage
- Broadcast a message etc.

Fragments:

- Represents a portion of user interface in an Activity.
- We use fragments to create reusable codes in an activity.

Android Project Structure



Useful File in the project

- In the previous diagram
- The root folder "App" refers the name of your project. The "app" folder is a module in your project. Inside this there is "build" folder it contains files that Android Studio creates for you. You don't usually edit anything in this folder.
- R. Java contained in the folder: build/generated/source
- Every Android project needs a file called R.java, which is created for Android uses this file to keep track of resources in the app.
- "Src" directory contains source code you write and edit. It contains two important directories.
  - ✓ Java
  - ✓ Res

Java:

- The java folder contains any Java code you write. Any activities you create live here.
- inside this directory we get MainActivity.java
- MainActivity.java defines an activity. An activity tells Android how the app should interact with the user.

Res:

- ❖ inside this directory we have layout and values.
- ❖ In Layout folder there is activity_main.xml. This file defines a layout. A layout tells Android how your app should look.
- ❖ In Values folder we found different files like strings.xml, colors.xml, styles.xml
  - ✓ Strings.xml: is a String resource file. It includes Strings such as the app's name and any default text value. Other files such as layouts and activities can look up text values from here.
  - ✓ Colors.xml: contains color resources of the Android application. Different color values are identified by a unique name that can be used in the Android application program.
  - ✓ Styles.xml: file contains resources of the theme style in the Android application.

Additional folders in res include Mipmap and drawable

Mipmap: folder contains the Image Asset file that can be used in Android Studio application.

- ✓ A mipmap is an image that can be used for application icons, and they're held in mipmap* folders in

app/src/main/res. You can generate the following icon types like Launcher icons, Action bar and tab icons, and Notification icons.

Drawable:

- A Drawable folder contains resource type file (something that can be drawn). Drawable may take a variety of file like Bitmap (PNG, JPEG), Nine Patch, Vector (XML), Shape, Layers, States, Levels, and Scale.

AndroidManifest.xml:

- Every Android app must include a file called AndroidManifest.xml at its root.
- The manifest file defines the structure and metadata of our application, its components, and its requirements. This file includes nodes for each of the Activities, Services, Content Providers and Broadcast Receiver that make the application and using Intent Filters and Permissions, determines how they co-ordinate with each other and other applications.

Package name in Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp"
    android:versionCode="1"
    android:versionName="1.0" >
    ...
</manifest>
```

- The following image shows the root <manifest> element with the package name “com.example.myapp”.
- The next images show how we declare Activity

```
<manifest package="com.example.myapp" ... >
    <application ... >
        <activity android:name=".MainActivity" ... >
            ...
        </activity>
    </application>
</manifest>
```

❖ The activity in the above figure is resolved to: "com.example.myapp.MainActivity"

Build.gradle:

❖ Every Android project needs a gradle for generating an apk from the *.java* and *.xml* files in the project.

E.g. Converts the java files into dex files and compresses all of them into a single file known as apk that is actually used.

❖ Whenever you click on **Run** button in android studio, a gradle task automatically triggers and starts building the project and after gradle completes its task, app starts running in AVD or in the connected device.

❖ This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion,

versionCode and versionName.

- ❖There are two types of build.gradle scripts:
  - ✓Top-level build.gradle
  - ✓Module-level build.gradle

Top-level build.gradle:

It is located in the root project directory and its main function is to define the build configurations that will be applied to all the modules in the project.

Module-level build.gradle:

It is a place where all dependencies are defined and where the SDK versions are declared.