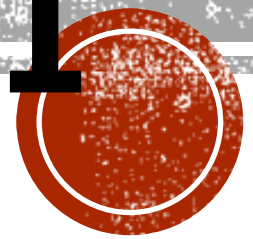
MOBILE APPLICATION DEVELOPMENT



By Ankit Attkan

WHAT IS ANDROID DEVELOPMENT?

- Android is an operating system. That is, it's software that connects hardware to software and provides general services. Like web application development, mobile application development has its roots in more traditional software development.
- It is mobile specific operating system: an OS designed to work on mobile (read: handheld, wearable, carry-able) devices.
- The term "Android" can be referred as the "platform" (e.g., devices that use the OS) as well as the ecosystem that surrounds it.
- "Android Development" technically means developing applications that run on the specific OS, it also gets generalized to refer to developing any kind of software that interacts with the platform.
- The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

ANDROID HISTORY

- **2003**: The platform was originally founded by a start-up “Android Inc.” which aimed to build a mobile OS operating system.
- **2005**: Android was acquired by Google, who was looking to get into mobile
- **2007**: Google announces the Open Handset Alliance, a group of tech companies working together to develop “open standards” for mobile platforms.
  - Members included phone manufacturers like HTC, Samsung, and Sony;
  - Mobile carriers like T-Mobile, Sprint, and NTT DoCoMo;
  - Hardware manufacturers like Broadcom and Nvidia;
  - The Open Handset Alliance is now group of 84 technology and mobile companies

ANDROID HISTORY

- **2008**: First Android device is released: the HTC Dream (a.k.a. T-Mobile G1)
  - Specs: 528Mhz ARM chip; 256MB memory; 320x480 resolution capacitive touch; slide-out keyboard! Author's opinion: a fun little device.
- **2010**: First Nexus device is released: the Nexus One. These are Google-developed "flagship" devices, intended to show off the capabilities of the platform.
  - Specs: 1Ghz Scorpion; 512MB memory; .37" at 480x800 AMOLED capacitive touch.
- **2014**: Android Wear, a version of Android for wearable devices (watches) is announced.
- **2016**: Daydream, a virtual reality (VR) platform for Android is announced

ANDROID VERSIONS, NAME AND API LEVEL

Code name	Version numbers	API level	Release date
No codename	1.0	1	September 23, 2008
No codename	1.1	2	February 9, 2009
Cupcake	1.5	3	April 27, 2009
Donut	1.6	4	September 15, 2009
Eclair	2.0 - 2.1	5 - 7	October 26, 2009
Froyo	2.2 - 2.2.3	8	May 20, 2010
Gingerbread	2.3 - 2.3.7	9 - 10	December 6, 2010
Honeycomb	3.0 - 3.2.6	11 - 13	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	October 18, 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	July 9, 2012

ANDROID VERSIONS, NAME AND API LEVEL

Code name	Version numbers	API level	Release date
KitKat	4.4 - 4.4.4	19 - 20	October 31, 2013
Lollipop	5.0 - 5.1.1	21- 22	November 12, 2014
Marshmallow	6.0 - 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 - 7.1.2	25	October 4, 2016
Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018
Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020

FEATURE OF ANDROID

As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

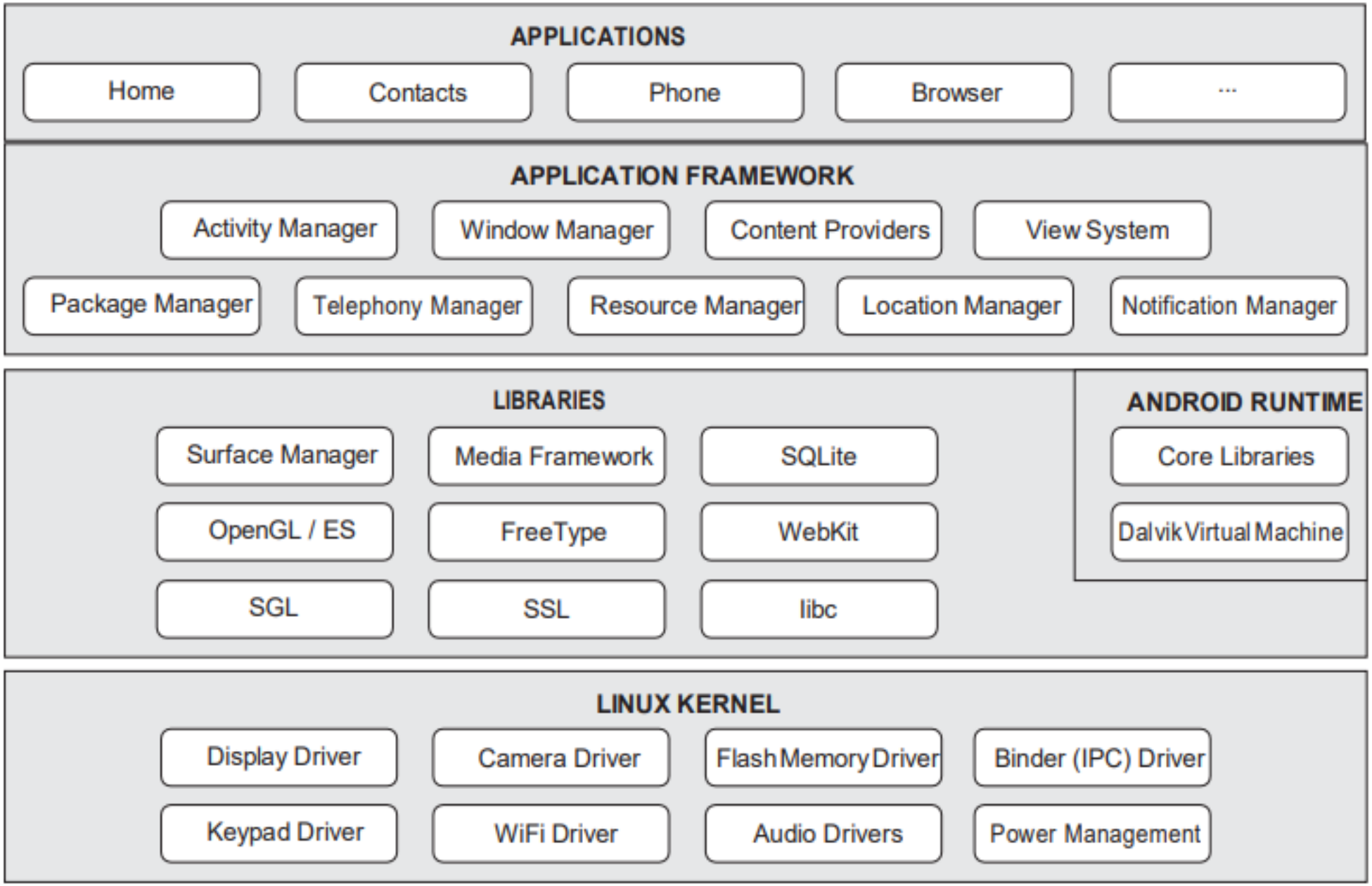
- Storage- Uses SQLite, a light weight relational database, for data storage.
- Connectivity- Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE and WiMAX.
- Messaging—Supports both SMS and MMS.
- Web browser- Based on the open-source WebKit, together with Chrome‘s V8 JavaScript engine.
- Flash support- Android supports Flash 10.1.

FEATURE OF ANDROID

- Media support- Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, WAV, JPEG, PNG, GIF, BMP, etc.
- Hardware support- Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
  - Multi-touch- Supports multi-touch screens,
  - Multi-tasking- Supports multi-tasking applications
- Tethering- Supports sharing of Internet connections as a wired/wireless hotspot Architecture of Android .

ANDROID ARCHITECTURE:

- Android architecture is roughly divided into five sections in four main layers to explain the working of android that make up the Android operating system (OS), namely:
  1) Applications
  2) Application Framework
  3) Libraries
     - Android Runtime
  4) Linux kernel
- The primary hardware platform is the ARM architecture.


APPLICATIONS
Home
Contacts
Phone
Browser
...
APPLICATION FRAMEWORK
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
LIBRARIES
Surface Manager
Media Framework
SQLite
OpenGL / ES
FreeType
WebKit
SGL
SSL
libc
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine
LINUX KERNEL
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

ANDROID ARCHITECTURE LAYERS:

Like so many other systems, the Android platform is built as a layered architecture as:

Linux Kernel:

- At it's base, Android runs on a Linux kernel for interacting with the device's processor, memory, etc. Thus an Android device can be seen as a Linux computer.
- Android uses the Linux kernel, which is also open source, for device drivers, memory management, process management, file system and network management. The kernel has some features added, some dropped from the Linux kernel.
- Android does not support Sys V IPC mechanisms but instead uses IPC Binder. The binder driver that is used for IPC is included to the kernel along with low memory killer, power management, logger and ASHMEM (Anonymous SHared MEMory).

LINUX KERNEL:

I. **Low Memory Killer-** Kills processes as available RAM becomes low. The lowmemorykiller driver lets user-space specify a set of memory thresholds where processes within a range can be killed when the available memory reduces. This is necessary as when an application is closed in Android, the process does not get killed but instead stays in memory. This reduces the start-up time when the application has to be used again. This driver is built on top of the Linux.

II. **Logger** – system logging facility

III. **ASHMEM** – is mechanism to share blocks of data across processes, similar to POSIX shm. The modifications to POSIX shm help Android to reclaim memory blocks not currently in use.

IV. **Power Management** – Android has to operate with substantially limited energy footprint. Consequentially, Android implements a power management driver on top of standard Linux power management. Applications use user space libraries to inform Power Management framework of their constraints.

V. **PMEM** – allocates physically contiguous memory

LIBRARIES:

Android includes C / C++ native libraries which are exposed via the Application framework. So, these are called through Java interfaces. By native libraries, we mean code that is hardware specific. Most of these libraries are open source libraries which have been used without any changes.

- The **Bionic library** is an exception; it is the optimized version of the Standard C library with the Apache license, like rest of Android, instead of GPL license of the GNU libc. The GNU libc has been rewritten taking into consideration the energy constraint of mobile devices that run Android.
- **WebKit**: Layout engine software designed to allow web browsers to render web pages. WebKit also powers the Apple Safari and Google Chrome browsers.

LIBRARIES:

- **Media Framework**: the libraries support playback and recording of many popular audio and video formats including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **SQLite**: A powerful yet light-weight SQL database engine. The applications use this to store and retrieve data.
- **OpenGL**: graphics libraries, OpenGL ES is a subset of OpenGL meant for Embedded Systems
- **SGL**: acronym for Scalable Graphics Libraries, renders 2D graphics
- **OpenSSL**: the open source tool kit that implements Secure Socket Layer (SSL).
- **Surface Manager** to monitor display functionalities and text manipulation during display.
- Readily available Widgets such as buttons, layouts, radio button, lists.

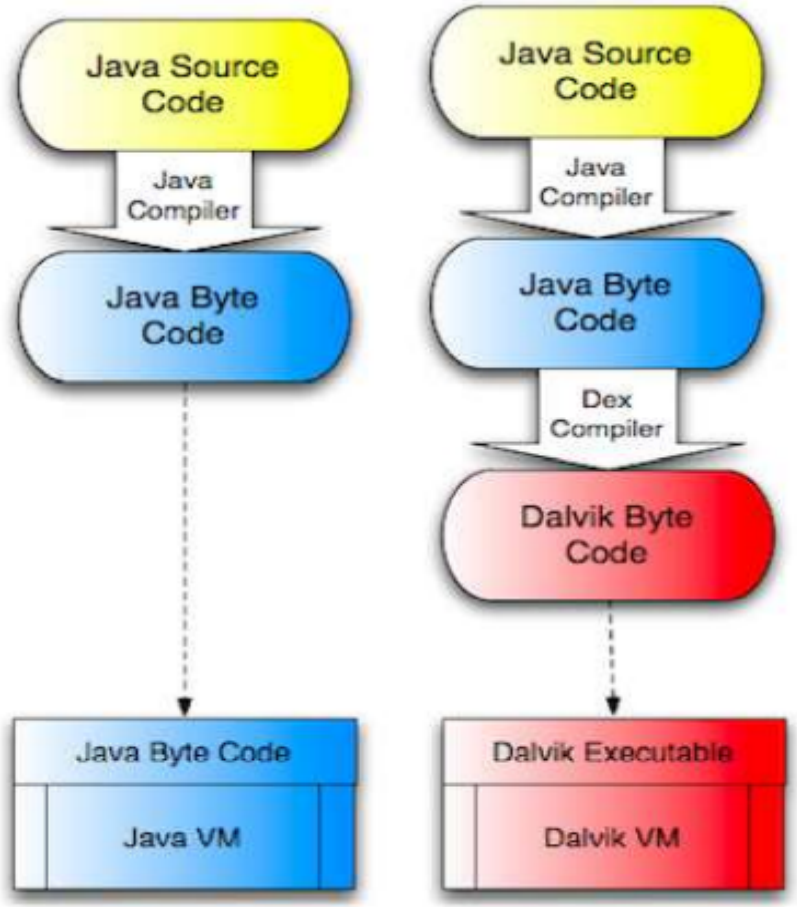
ANDROID RUNTIME

- At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.
- **Core Libraries**: Android re-implements core Java libraries that the layers above use. Android includes most of Java 5 Standard Edition functionalities in its implementation.

DALVIK VIRTUAL MACHINE (DVM):

- Android does not use the standard Java Virtual Machine but uses the DVM which has been tailor made for mobile devices and is an integral part of Android.
- DVM ensures that applications can run in systems with relatively smaller RAM, slower processors and without swap space in comparison to desktops.
- This is especially relevant as the mobile devices are battery powered and memory efficiency of programs is very important. As a consequence, Dalvik itself is small and operates with tighter overhead constraints.
- DVM is a register-based architecture unlike JVM which is stack-based. Dalvik has its own byte code and hence the standard Java compiler generated byte code has to be converted to a .dex file. The process of converting from Jave code to dex files is called **“dexing”.**
- The .dex is the DVM compatible byte code which is compact and efficient. The uncompressed .dex file is smaller than a compressed .jar file.
- All applications run as separate processes with their own instance of the DVM. This adds to the security of Android.

DALVIK VIRTUAL MACHINE (DVM):



APPLICATION FRAMEWORK:

- The second top layer is the Application framework layer implemented in Java which offers a repertory of APIs catering to application developers.
- This layer exposes the operating system's capability in the form of services for effective and innovative application development.
- The components like Activity Manager, Resource manager etc., are through which applications interact with OS and other applications.
- **Content Providers** a primary building block providing encapsulated content to applications from a central repository of data. Content providers are necessary when the content or data is to be shared across applications. The content provider works along with ContentResolver to provide this data. The ContentResolver object in the client application's process and the ContentProvider object in the application that owns the provider automatically handle inter-process communication.

public abstract class ContentProvider

APPLICATION FRAMEWORK:

- **Activity Manager:** The Activity Manager essentially handles the activity life cycle of an application. For example, it is responsible for creating the activity when an application starts, move an activity off screen when use navigates to a different screen. It interacts with all activities running in the system.

public class ActivityManager

- **Window Manager:** The window manager is responsible for organizing the screen. It decides which application goes where and layering on the screen.
- **Package Manager:** It is a class for retrieving various kind of information related to the application packages that are currently installed on the device.

public abstract class PackageManager

APPLICATION LAYER:

- Application layer is the top most layer of android architecture, exclusively written in Java. At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.
- These Java applications execute on the Dalvik Virtual Machine (DVM), a virtual machine much like JVM but for Android.
- **Application Anatomy:**

Activity: An activity is a single screen of the application that the user sees at one time, for example, the Contacts screen. However, the activity does not have to paint the entire screen. It displays the UI component and responds to a user or system initiated action. Activity code runs only when the Activity GUI is visible and has focus. An application comprises of many such activities and Android maintains a history stack, 'back stack' of all the activities which are spawned. Maintaining this history enables Android to put out the activity on the screen quickly when a user returns to a previous displayed screen.

APPLICATION LAYER:

- **Services:** Service is an application component to enable an application to perform long running tasks in the background. This code runs when user interaction is needed, for e.g., applications continue to play music when you have navigated away to the home screen.
- **Content Provider:** The data storage and retrieval in Android applications is done via content providers. It is implemented as a subclass of ***ContentProvider.***
- **Broadcast Receiver:** A broadcast receiver is a component that responds to system-wide broadcast announcements. This part of the application decides what the application has to do, for instance, when the battery is low. The battery low message is broadcast. A broadcast receiver is implemented as a subclass of ***BroadcastReceiver***

PROGRAMMING LANGUAGES:

- **Java:** Android code (program control and logic, as well as data storage and manipulation) is written in Java.
  - Writing Android code will feel a lot writing any other Java program: you create classes, define methods, instantiate objects, and call methods on those objects. But because you're working within a **framework**, there is a set of code that *already exists* to call specific methods.
- **XML:** Android user interfaces and resources are specified in XML (E**X**tensible **M**arkup **L**anguage).
  - To compare to web programming: the XML contains what would normally go in the HTML/CSS, while the Java code will contain what would normally go in the JavaScript. XML is just like HTML, but you get to make up your own tags. Except we'll be using the ones that Android made up; so it's like defining web pages, except with a new set of elements.

ANDROID APPLICATION:

- Android applications (the source, dexed bytecode, and any resources) are packaged into **.apk** files.
- The .apk files are then cryptographically signed to specify their authenticity, and either "side-loaded" onto the device or uploaded to an App Store for deployment.
  - The signed .apk files are basically the "executable" versions of your program!
- While building an App you need to:
  1. Generate Java source files (e.g., from resource files, which are written XML used to generate Java code)
  2. Compile Java code into JVM bytecode
  3. "dex" the JVM bytecode into Dalvik bytecode
  4. Pack in assets and graphics into an APK
  5. Cryptographically sign the APK file to verify it
  6. Load it onto the device

ANDROID DEVELOPMENT TOOL:

- **Hardware**: Android code is written for a virtual machine anyway, Android apps can be developed and built on any computer's operating system.
  - Android apps will need to be run on Android devices.
  - Physical devices are the best for development (they are the fastest, easiest way to test), though you'll need USB cable to be able to wire your device into your computer.
  - Don't even need cellular service (just WiFi should work)
  - Need to turn on developer options in order to install development apps on the device.
  - Use the Android Emulator if you don't have a physical device, which is a "virtual" Android device. It does have some limitations (e.g., no cellular service, no bluetooth, etc).

ANDROID DEVELOPMENT TOOL:

- **Software** needed to develop Android applications includes:
  - **Java 7 SDK** (not just the JRE!), because you're writing Java code.
  - **Gradle or Apache ANT:** These are *automated build tools*—in effect, they let you specify a single command that will do a bunch of steps at once (e.g., compile files, dex files, move files, etc).
    - ANT is the "old" build system
    - Gradle is the "modern" build system
  - **Android Studio & Android SDK** is the official IDE for developing Android applications. IDE comes bundled with the SDK.
    - Android Studio provides the main build system: all of the other software (Java, Gradle) goes to support this.

ANDROID DEVELOPMENT TOOL:

SDK comes with a number of useful command-line tools. These include:

- adb, the "Android Device Bridge", which is a connection between your computer and the device (physical *or* virtual). This tool is used for console output!
- emulator, which is a tool used to run the Android emulator
- *deprecated/removed* android: a tool that does SDK/AVD (Android Virtual Device) management. Basically, this command-line utility did everything that the IDE did, but from the command-line! It has recently been removed from the IDE.

- It is recommended to make sure that the SDK command-line tools are installed.
- Put the tools and platform-tools folders on your computer's PATH;
  - you can run adb to check that everything works.
- All of these tools are built into the IDE, but they can be useful fallbacks for debugging.

THANK YOU