ICT 9025

# Mobile Device Platforms

**Lecture 6**

# Mobile Operating Systems (Mobile Device Platforms)

- As of August 2011, there were 9 popular mobile operating systems:
  - Android OS (Google Inc.)
  - Bada (Samsung Electronics)
  - BlackBerry OS (Blackberry Limited)
  - iOS (Apple)
  - MeeGo OS (Nokia and Intel)
  - Palm OS (Garnet OS)
  - Symbian OS (Nokia)
  - webOS (Palm/HP)
  - Windows Mobile (Windows Phone)

# Mobile Operating Systems

- As of June 2021, Android OS has 72% market share in the mobile OS market
  - i.e. 72% of all consumer mobile devices globally, use Android OS
  - Google's Android and Apple's iOS jointly possess over 99 percent of the global market share

*Note: our scope for Mobile Device Platforms will mostly focus on the Android OS*

# What is Android?

- A software platform and operating system for mobile devices.
- Based on the Linux kernel.
- Developed by Google and later the Open Handset Alliance (OHA).
- The Android platform was announced on 5 November 2007 with the founding of OHA.

# History of Android

- Google acquired the startup company Android Inc. in 2005 to start the development of the Android Platform.

- In late 2007, a group of industry leaders came together around the Android Platform to form the Open Handset Alliance (OHA) (http://www.openhandsetalliance.com).

# History of Android

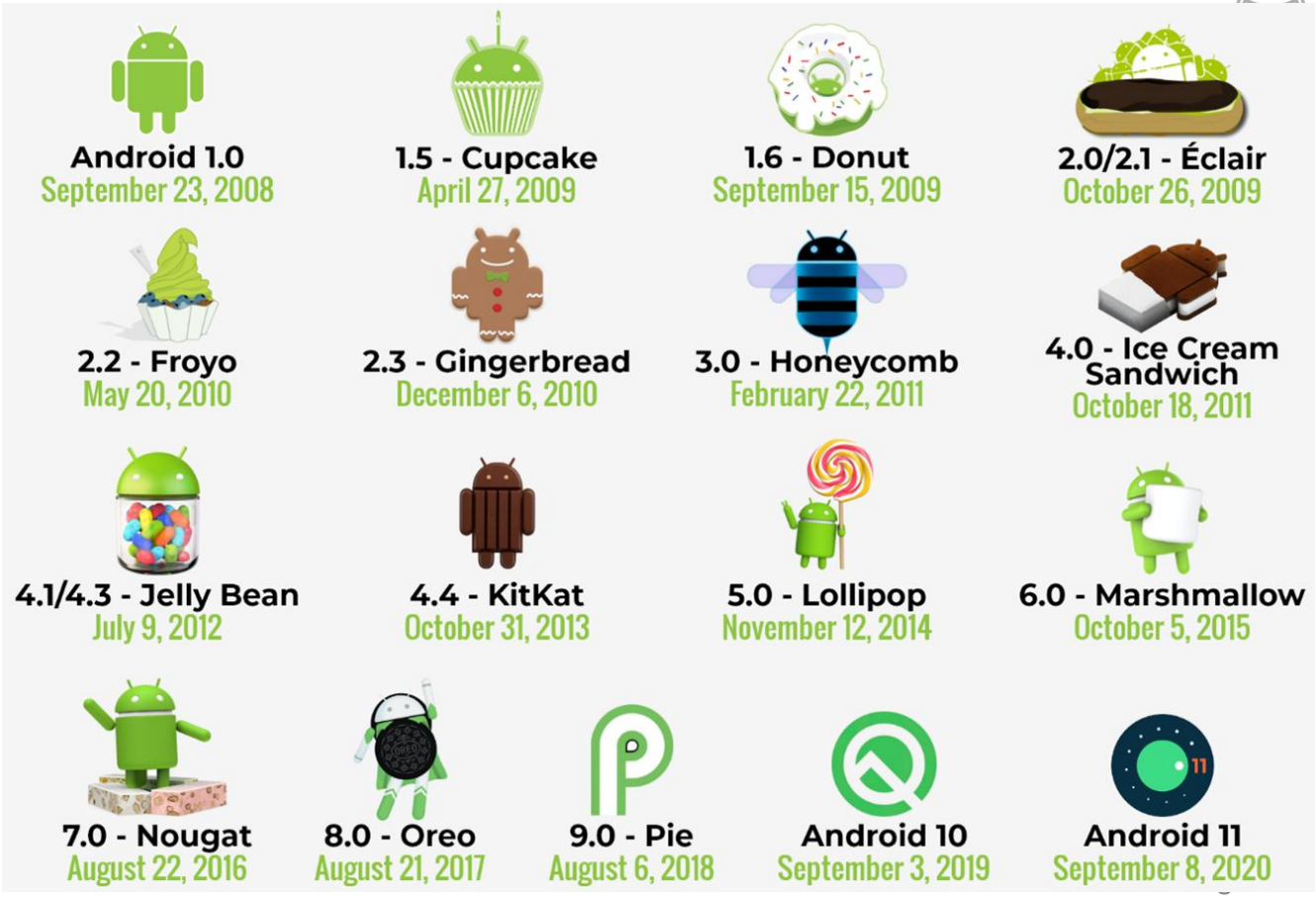
- The Android SDK was first issued as an “early look” release in November 2007.
- In September 2008 T-Mobile announced the availability of the T-Mobile G1, the first smartphone based on the Android Platform.
- A few days after that, Google announced the availability of Android SDK Release Candidate 1.0.
- In October 2008, Google made the source code of the Android Platform available under Apache’s open source license.

# What is the Open Handset Alliance (OHA)?

- It's a consortium of several companies

# Android Versions

- The Android OS (and iOS) are continuously updated.
- Each updated version of Android adds improvements to the OS




*Each Android version from V1.5 to V9 was given a codename by Google (developers)*

# Android Versions


*(Extra information, will not be in assessments)*
*Android codenames return with Android 12 which has the codename Snow Cone.*
*Note that the Android codenames:*

*a) use words in alphabetical order (beginning from C to S)*
*b) each code name is named after a dessert (or sweet food)*

# Android Versions

*Example of smartphones using Android 1.0 in 2008*

# Android Versions

*Example of smartphones using Android 12 in 2021*

# Android Software Development

- Android Development requirements:
  - Java or Kotlin
  - Android SDK

- **Java** and **Kotlin** are the programming languages used to create Android applications
  - Java was used from Android 1.0
  - Google announced support for Kotlin on Android in November 2017
  - Google announced that the Kotlin programming language is now its preferred language for Android app development in May 2019

# Android Software Development

- Android SDK includes:
  - Class Libraries
  - Developer Tools
  - Emulator and System Images
  - Documentation and Sample Code

# Android IDE

- Top IDEs for Android Development:
  - **Android Studio**
    - Google's official IDE for Android development
    - Supports Java and Kotlin programming
  - **Eclipse**
    - An open source IDE, a popular Android IDE
    - Initially used only for Java desktop apps, expanded to support Android mobile app development
    - *Note, however, in 2015, the Eclipse website recommended using Android Studio, the official IDE for Android development*

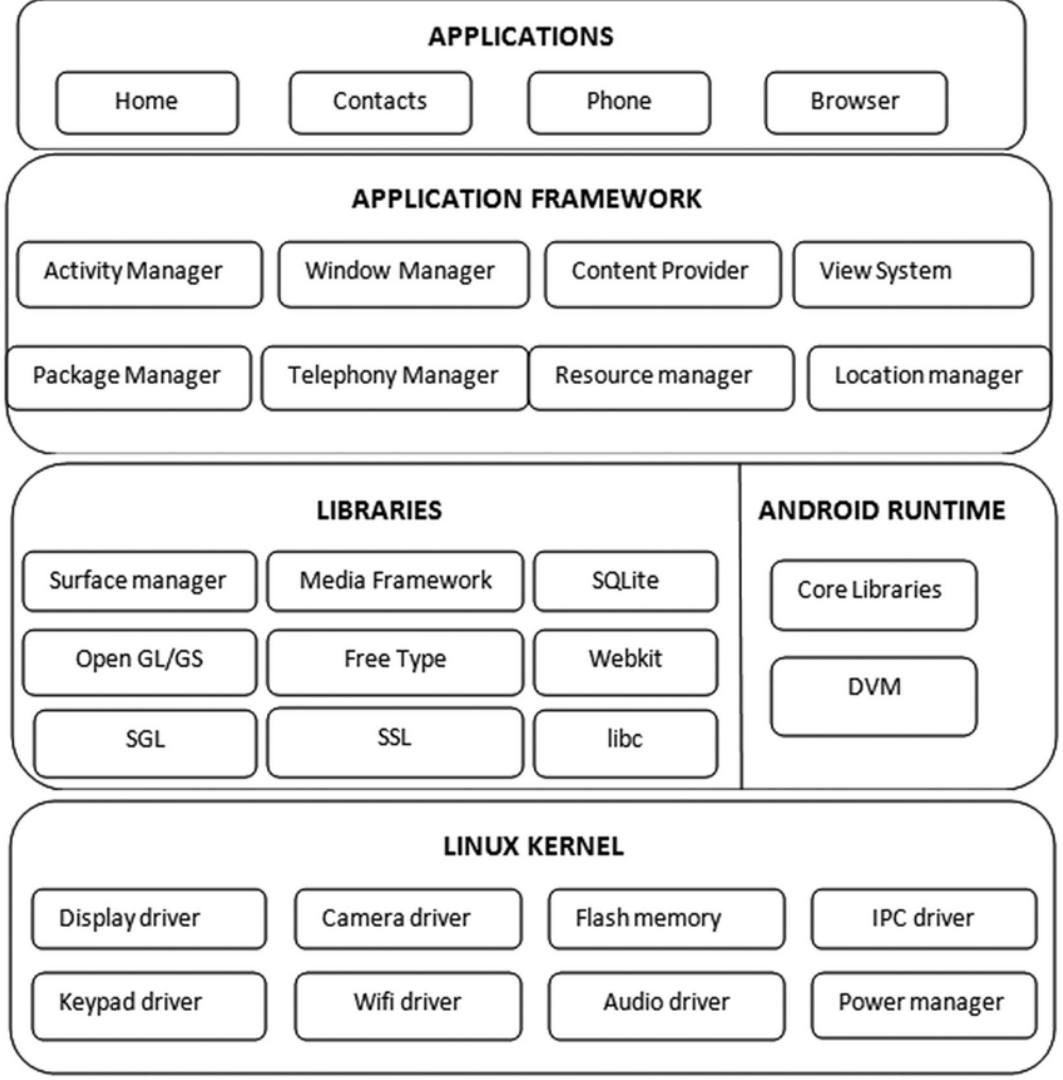
# Android IDE

- Top IDEs for Android Development (cont.) :
  - **Visual Studio (with Xamarin)**
    - Microsoft's IDE which comes in multiple versions
    - when integrated with Xamarin, it enables cross-platform native app development
  - **IntelliJ IDEA**:
    - (Similarly like Eclipse) Initially used only for Java desktop apps, expanded to support Android mobile app development

*Note that there are many more IDEs for Android development including: Netbeans, Komodo, Cordova, PhoneGap, Appcelerator Titanium etc.*

# Android System Architecture Overview

- Based on the **Linux kernel** (bottom layer which controls hardware and software connectivity)
- Includes various **libraries** (some of which are cross-platform libraries)
- Includes application **frameworks** (including classes (code) exclusive to Android)
- **Applications** (the top layer, the user carries tasks via the applications GUI (user interface)



# Android System Architecture - Libraries

- Software Library
  - a collection of programs, resources and software packages used by programs whereby the library is stored and loaded locally/on disk for immediate use.

- **Libraries**
  - **Free Type** – free software library for rendering fonts
  - **Surface Manager** – library that provides display functionality.
  - **Media Framework** – a library used for playing, recording, and editing audio and video formats.

# Android System Architecture - Libraries

- **Libraries (cont.)**
  - **SQLite** – a library that implements a SQL database providing database functionality.
  - **Open GL** (Open Graphics Library) – a graphics library that manages displaying 2D and 3D graphics
  - **SGL** (Scalable Graphics Library) – a library that handles displaying 2D graphics

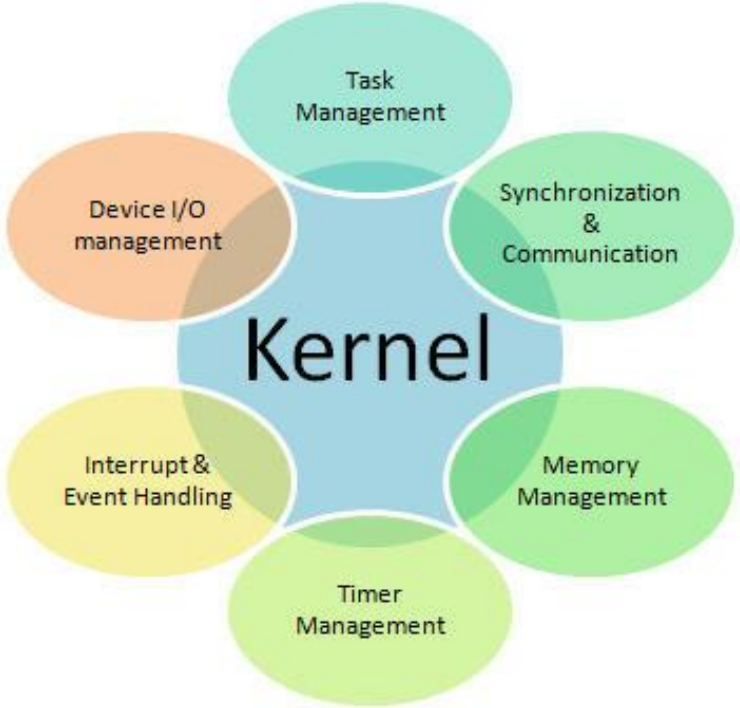
# Android System Architecture - Libraries

- **Libraries (cont. )**
  - **Webkit** – a library which provides browser support
  - **SSL** (Secure Socket Layer) – provides Internet security
  - **libc** – a standard C library which provides functionality for string handling, mathematical computations, input and output processing and memory management, working with the Linux kernel.
    - **Bionic** – a library developed by Google for Android, Bionic is a variant of libc designed for devices with less memory and processor power than traditional Linux-based devices.

# Android System Architecture - Kernel

**Linux Kernel**

- A kernel:
  - The main component of an operating system, the kernel is a system program that
    - controls all programs running on the computer/machine.
    - manages the connection between the software and hardware of the system.
    - The kernel provides process management, task management and disk management (including other operating system functions).



# Android System Architecture - Kernel

- (Beyond the functionality of the kernel)
  - The operating system includes **protection and security** functionality of the machine.
  - The operating system provides a **user interface** between the user and the machine whereas the kernel converts user commands into machine language and process management (and other operating system functions)

# Android Architecture

- Android Runtime - Delvik Virtual Machine (DVM)
  - **JVM**
    - The Java Virtual Machine (JVM) compiles Java code into bytecode such that Java programming becomes platform independent (various different systems can run Java code so long as JVM is on the system.
    - JVM is for Java desktop application
  - **DVM**
    - Like JVM, however, specifically designed for mobile Java apps, DVM focuses on running on **low memory mobile devices** and loading quicker compared to any JVM

# Android Architecture

- **Android Runtime (ART)**
  - Note that DVM was discontinued and no longer is use (was important to earlier versions of Android)
  - **ART** is used in Android OS after DVM was discontinued, ART provided performance improvements which the end users may not be aware of

*On the Android Architecture diagram (Slide 17, DVM can be replaced with ART)*

# Android Development Process

- Setup develop environment (SDK, Android Stduio)
  - SDK: compiler, debugger, device emulator
    - Multiplatform support: Windows, Mac, Linux
  - Java programming: has its own Java Virtual Machine (known as ART) and special byte code
- Create app
  - Android project containing java files and resource files
- Test app
  - Pack project into apk  format (via SDK and IDE)
  - Install, run and debug on emulator or device
- Publish app in Android market
  - Earn profit!

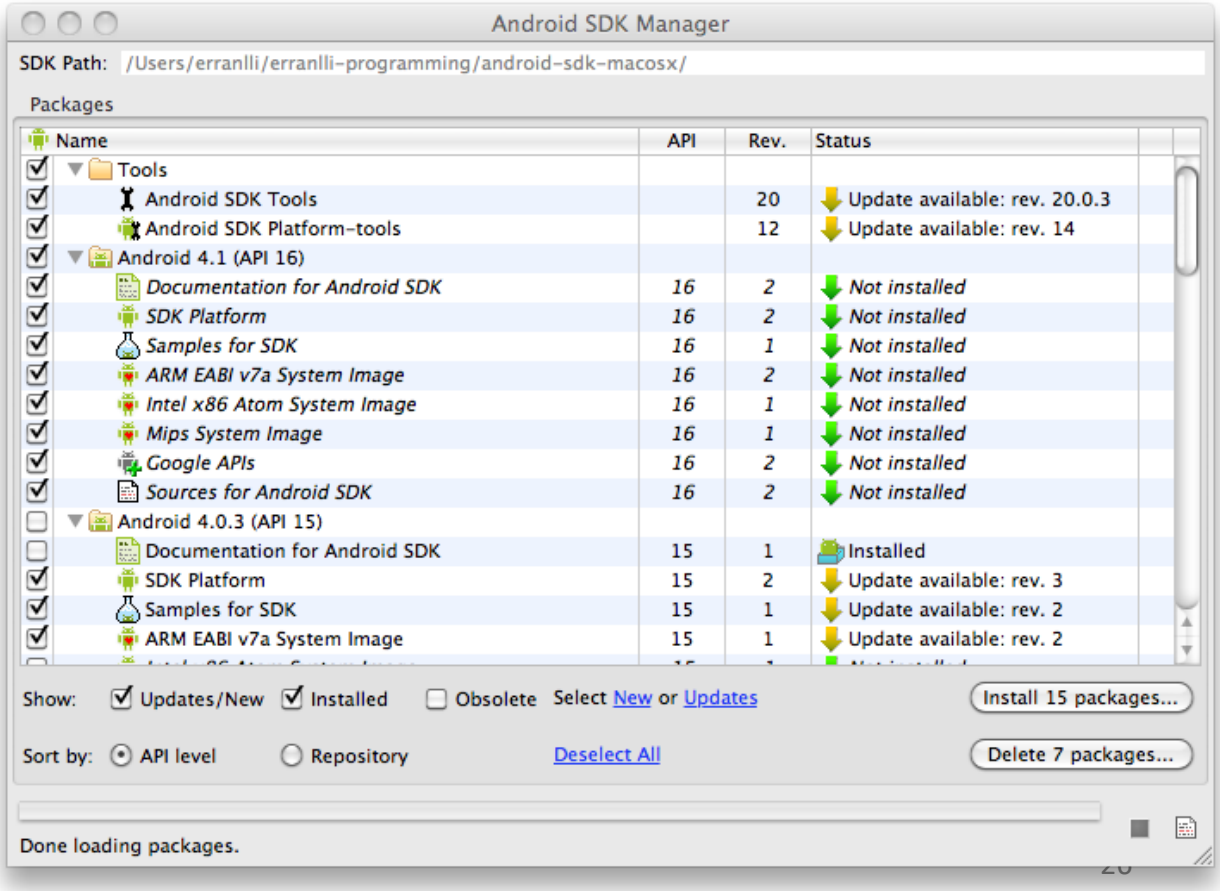
# Setup Android Studio

- Download and install Android SDK:
  - https://developer.android.com/studio/#downloads
- Scroll down and select the Android Studio Package for your OS (where android apps will be developed)
- After the download, run the Android Studio setup file and follow the installation instructions

***Our scope: note that you will not be required to download and install Android Studio for ICT 9025, however you should familiarize yourself with some Android concepts***

# Android SDK Manager

- The Android SDK (software development kit) allows developers to choose which android version and other features they want to work with, Android SDK can be further downloaded via Android Studio's installation

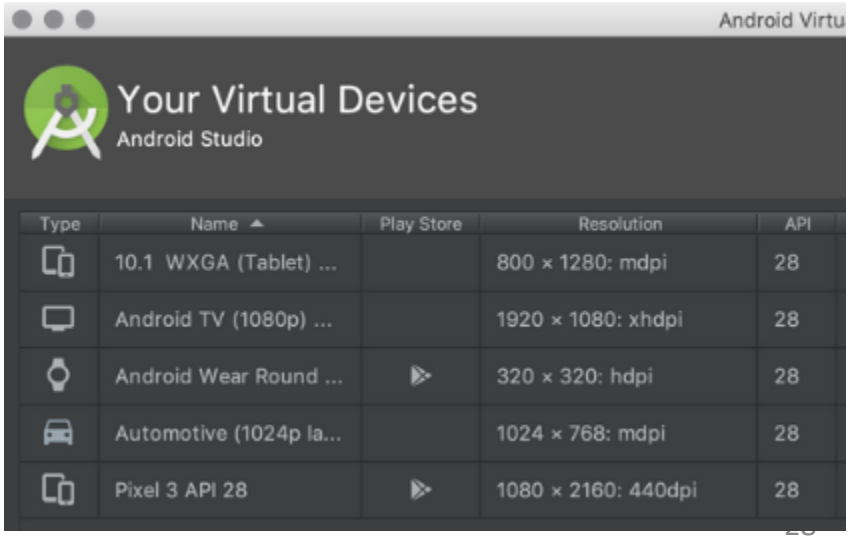


# Android Emulator

- The **Android Emulator** simulates Android devices on your computer so that you can test your application on a variety of devices and Android versions without needing to have each physical device

# Android Emulator – Install via Android Studio

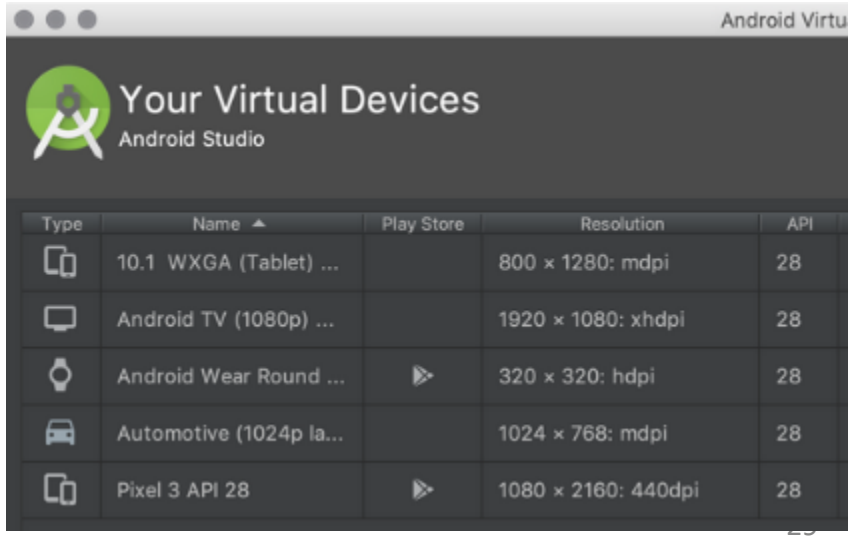
- An **Android Virtual Device (AVD)**
  - software that is designed to simulate an Android phone, tablet, Wear OS, Android TV (etc.) which works in the Android Emulator
  - The **AVD Manager** is an interface you can launch from Android Studio that helps create and manage AVDs.



# Android Emulator – Install via Android Studio

- To open the AVD Manager, do one of the following in Android Studio:
  - ➢ Select Tools > AVD Manager
  - ➢ Click AVD Manager AVD Manager icon in the toolbar.

*Android Virtual Devices can be installed from the AVD Manager*



# Android Application Framework

- A software framework which manages the basic components for Android application
- App components can be activated when any of the components need to be executed


*Some of the Android components in the Application Framework (there are other components as well)*

# Android App Components

| Basic Components | Description |
| --- | --- |
| Activity | Deals with UI aspects. Typically corresponds to a single screen |
| Service | Background tasks (e.g. play music in background while user is web surfing) that typically have no UI |
| BroadCastReceiver | Can receive messages (e.g. “low battery”) from system/apps and act upon them |
| ContentProvider | Provide an interface to app data. Lets apps share data with each other. |

# Activity

- UI portion of an app
- One activity typically corresponds to a single screen of an app
- Activity stack:
  - Activity is visible in the foreground, considered to be at the top of the stack
  - Background activities are stopped but their state is retained
  - Back button resumes previous Activity in the stack
  - HOME button moves an app and its activity to the background

# Android Intent Filter

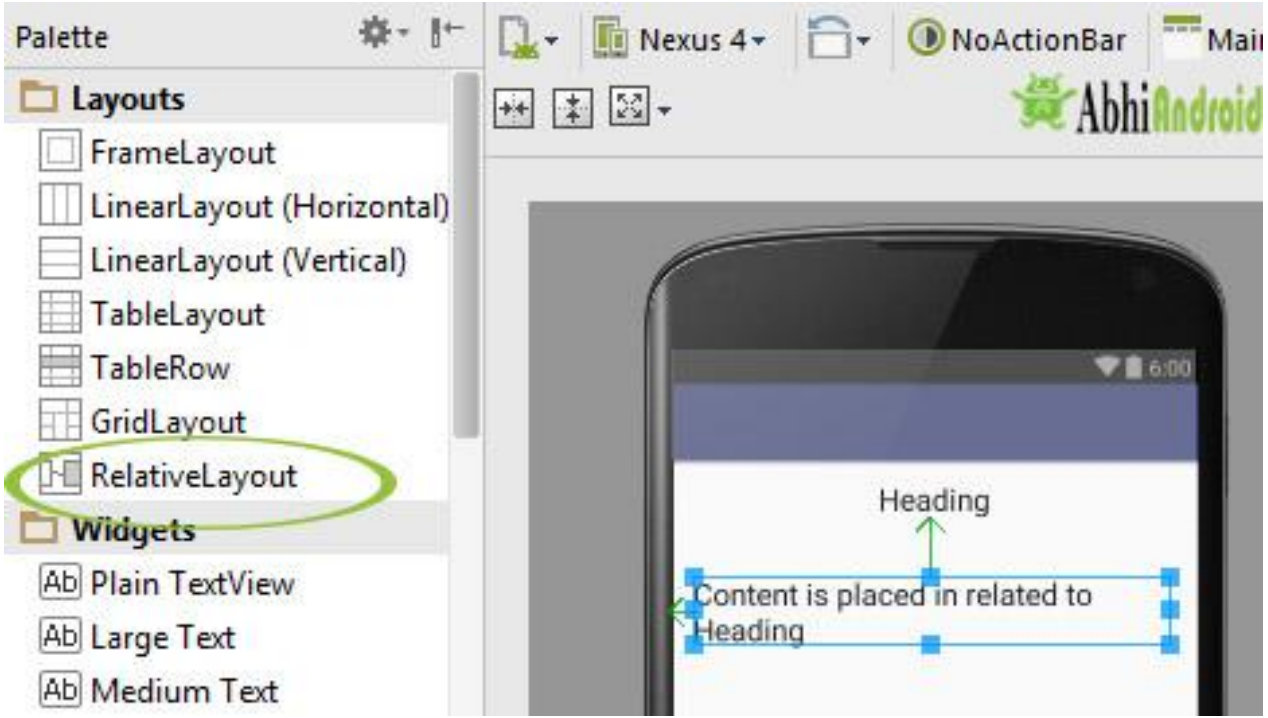
- The app needs a way of knowing what an Activity/Service can handle
- The Intent Filter is used such that Activities/Services/Receivers can declare what they want to receive via the Intent Filter

# Views

- Views are building blocks of UI (user interface)
- Two types of views:
  - Leaf:
    - TextView, EditText, Button, Form, TimePicker,ListView
  - Composite (ViewGroup):
    - LinearLayout, Relativelayout etc.

# Views

- Android Studio includes drag and drop functionality for Android Views



# Views

- XML code will be generated for each View (ie code for the layout of buttons, text etc.) Java or Kotlin work with XML for the Android user interface)

```
<!-- textView is alignParentBottom -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView"
    android:text="Text Here is AlignParentBottom with bottom margin of 120dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="120dp" />
```

*For example*

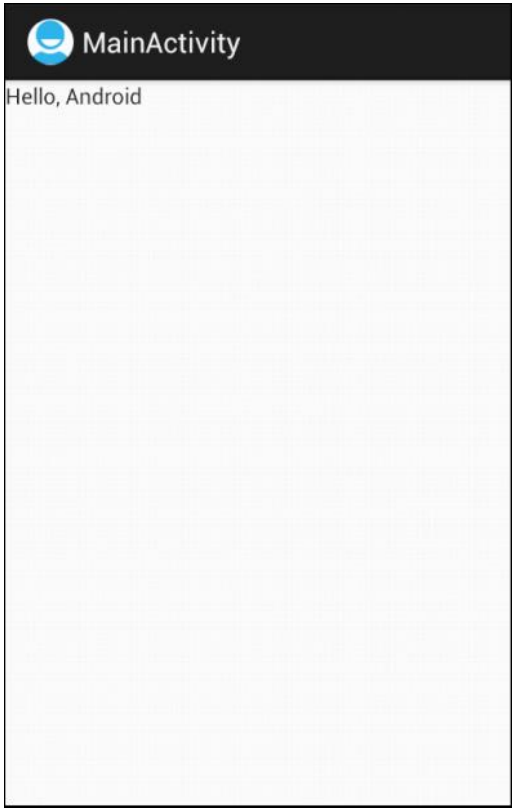
# Views

Views can also be created programmatically

**MainActivity.java**

```
public class MainActivity extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
            // savedInstanceState holds any data that
may have been saved
            // for the activity before it got killed by the
system (e.g.
            // to save memory) the last time

      super.onCreate(savedInstanceState);
       //setContentView(R.layout.main);
      TextView tv = new TextView(this);
      tv.setText("Hello, Android");
      setContentView(tv);
```



# Layouts

- Controls how Views are laid out / arranged:
  - LinearLayout,
  - TableLayout,
  - RelativeLayout

# Resources

- Resources are the additional files and static content that your code uses, such as bitmaps, images, layout definitions etc.

| | |
|---|---|
| `res/anim/` | XML files for frame-by-frame animation |
| `res/drawable/` | images compiled and optimized |
| `res/layout/` | XML files for screen layouts |
| `res/values/` | compiled XML files into different resource |
| `res/xml/` | arbitrary XML files |
| `res/raw/` | raw, uncompiled files |

# Services

- Components that typically run in the background
  - Music player, network download, etc

- Services can be started in two ways
  - A component can start the service by calling startService()
  - A component can call bindService() to create the service
- Service started using startService() remains running until explicitly killed
- Service started using bindService() runs as long as the component that created it is still “bound” to it.

- The Android system can force-stop a service when memory is low
  - However “foreground” services are almost never killed

# Broadcast Receivers

- Components designed to respond to broadcast messages (called Intents)
- Can receive broadcast messages from the system. For example when:
  - A new phone call comes in
  - There is a change in the battery level or cell ID
- Can receive messages broadcast by apps
  - Apps can also define new broadcast messages

# Content Providers

- Enable sharing of data across apps
  - Address book, photo gallery, etc.
  - Allows the app to share data online and with other apps
- Provides uniform APIs for
  - Query, delete, update, and insert rows
- API: extends ContentProvider implement methods such as insert, delete, query, update, oncreate

# Telephony APIs / Telephony Manager

- Includes components which support:
  - Send and receive SMS
  - Get mobile network info (network type, operator,…)
  - Get current value of network parameters (cellID, signal strength, roaming state etc.)
  - Monitor state changes (cellID, call state, connectivity …)
  - Get current device state (connected, idle, active)
  - Get device parameters (IMSI, IMEI, device type)

# Android - Gradle

- Gradle:
  - As set of tools used to automate and manage the build process while letting the user define flexible and custom build configurations
  - *Note that Gradle is installed (or downloaded) via Android Studio*

# Android – Kotlin and Java

- Kotlin:
  - A programming language designed to work with the JVM (Java Virtual Machine)

- Advantages of Kotlin:
  - Faster compile time: generally, Kotlin has faster compilation times than Java based projects
  - Improved readability: Kotlin will use less lines of code than equivalent Java projects
  - Additionally, Kotlin is designed to be fully **interopretational** with Java (such that Kotlin can be used along side Java in the same project)

# Online Resources for Android Developers

- Android API: http://developer.android.com/reference/packages.html
  - information on Android classes (Java examples included)

- Basics http://developer.android.com/guide/components/index.html
  - Introduction to Android Activities

- Firebase Cloud Messaging https://firebase.google.com/docs/cloud-messaging/
  - a cross-platform messaging solution for apps to receive notifications, and syncing user data

*Many more resources are available online,*
*Our scope is to understand some Android concepts*

# Mobile Apps vs Websites


## MOBILE WEB

Accessed via Mobile Browser

**PROS**
Content is Web Indexable
Device independent
Easier/Faster Development & Update
Easier to Measure

**CONS**
Connection Dependence
Limited Hardware Integration
Bandwidth & Browser Limited UX

## MOBILE APPS

Downloaded, and Executed in Mobile

**PROS**
Connection Independence
Better Hardware Integration
Better Graphic Performance & UX

**CONS**
Content is not Web Indexable
Device Dependent
More Expensive Development & Update
More Complex to Measure