

# History

- The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007, whereas the first commercial version, Android 1.0, was released in September 2008.
- On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and Performance.
- The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Code name <sup>◆</sup>	Version number <sup>◆</sup>	Initial release date <sup>◆</sup>	API level <sup>◆</sup>	Security patches <sup>[1]</sup> <sup>◆</sup>
<a href="#">(No codename)</a> <sup>[2]</sup>	1.0	September 23, 2008	1	Unsupported
<a href="#">(Internally known as "Petit Four")</a> <sup>[2]</sup>	1.1	February 9, 2009	2	Unsupported
Cupcake	1.5	April 27, 2009	3	Unsupported
Donut <sup>[3]</sup>	1.6	September 15, 2009	4	Unsupported
Eclair <sup>[4]</sup>	2.0 – 2.1	October 26, 2009	5 – 7	Unsupported
Froyo <sup>[5]</sup>	2.2 – 2.2.3	May 20, 2010	8	Unsupported
Gingerbread <sup>[6]</sup>	2.3 – 2.3.7	December 6, 2010	9 – 10	Unsupported
Honeycomb <sup>[7]</sup>	3.0 – 3.2.6	February 22, 2011	11 – 13	Unsupported
Ice Cream Sandwich <sup>[8]</sup>	4.0 – 4.0.4	October 18, 2011	14 – 15	Unsupported
Jelly Bean <sup>[9]</sup>	4.1 – 4.3.1	July 9, 2012	16 – 18	Unsupported
KitKat <sup>[10]</sup>	4.4 – 4.4.4	October 31, 2013	19 – 20	Supported; <sup>[11]</sup> See <a href="#">clarification</a>
Lollipop <sup>[12]</sup>	5.0 – 5.1.1	November 12, 2014	21 – 22	Supported
Marshmallow <sup>[13]</sup>	6.0 – 6.0.1	October 5, 2015	23	Supported
Nougat <sup>[14]</sup>	7.0 – 7.1.2	August 22, 2016	24 – 25	Supported
<b>Oreo</b>	<b>8.0</b>	<b>August 21, 2017</b>	<b>26</b>	<b>Supported</b>

# Android version : Features

- [https://en.wikipedia.org/wiki/Android\\_version\\_history](https://en.wikipedia.org/wiki/Android_version_history)

# How to Install

- Eclipse + ADT bundle

Download ADT bundle

<http://void-mainblog.blogspot.in/2015/04/download-eclipse-adt-bundle-for-android.html>

- Android Studio

<https://developer.android.com/studio/index.html>

# ADT

- The ADT Bundle provides everything you need to start developing apps, including a version of the Eclipse IDE with built-in ADT (Android Developer Tools) to streamline your Android app development.
- To add the ADT plugin to Eclipse:
  - > Start Eclipse , then select Help > Install New Software .
  - > Click Add , in the top-right corner

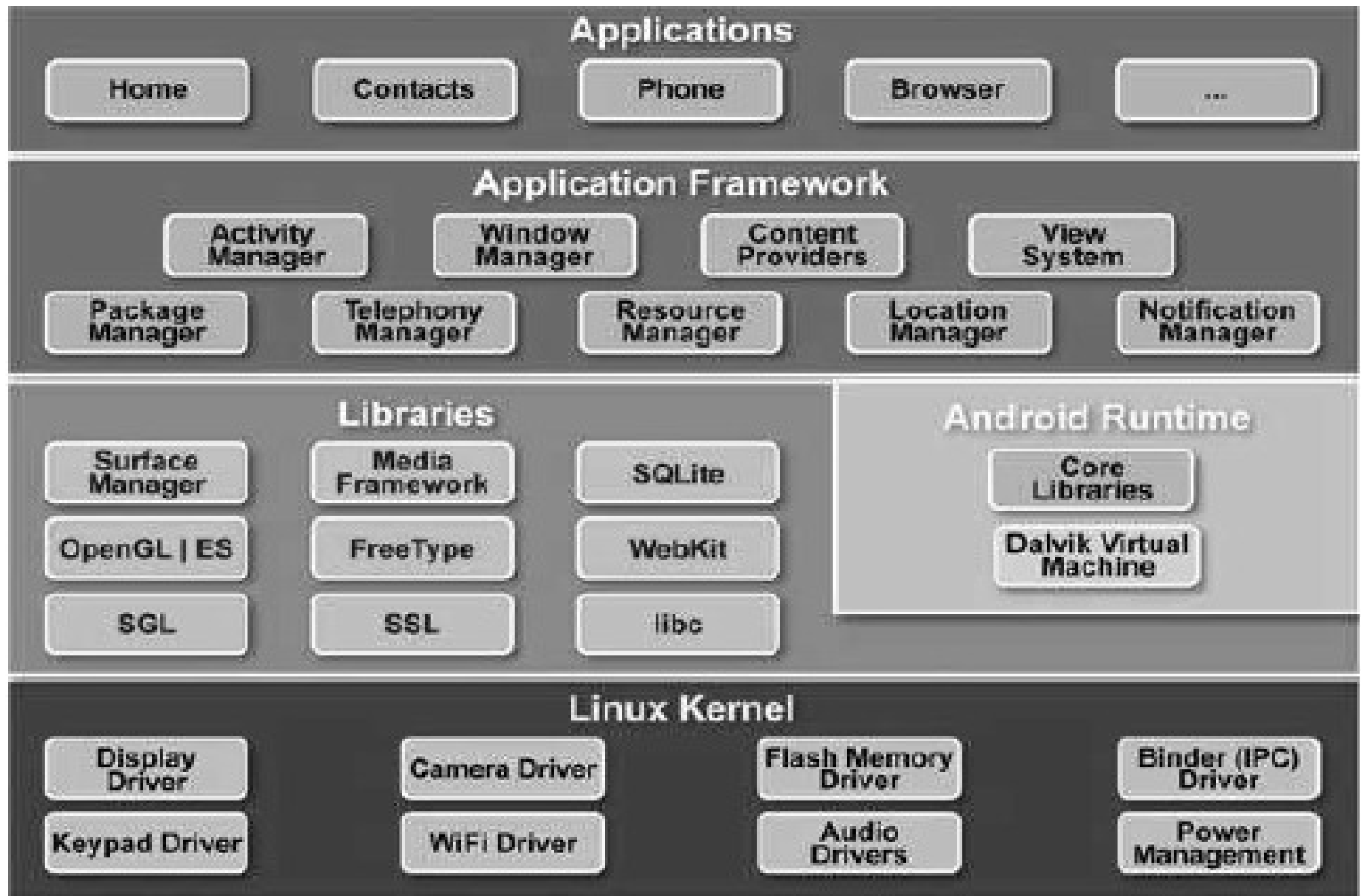
# Android Features

- Beautiful UI
- Connectivity
- Storage
- Media
- Messaging
- Web browser
- Multi-touch
- Multi-tasking
- Resizable widgets
- Multi-Language

# To Setup environment

- JDK
- Eclipse
- SDK (software development kit)
- ADT (Android development kit)

# Architecture





# Linux Kernel

At the bottom of the layers is Linux - Linux 2.6 with

- Process managment
- Memory managment
- Device managment

# **Libraries**

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

# Android Runtime

**Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of **core libraries** which enable Android application developers to write Android applications using standard Java programming language.

# DVM & ART

- ART (Android RunTime) is the next version of Dalvik. Dalvik is the runtime, bytecode, and VM used by the Android system for running Android applications. ART has two main features compared to Dalvik: Ahead-of-Time (AOT) compilation, which improves speed (particularly startup time) and reduces memory footprint (no JIT)

- Android runtime (ART) is the managed runtime used by applications and some system services on Android. ART and its predecessor Dalvik were originally created specifically for the Android project. ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.

# ART Features

- Ahead-of-time

-->ART introduces ahead-of-time (AOT) compilation, which can improve app performance. ART also has tighter install-time verification than Dalvik.

--> At install time, ART compiles apps using the on-device dex2oat tool. This utility accepts DEX files as input and generates a compiled app executable for the target device.

# ART Features

- Improved GC

Garbage collection (GC) can impair an app's performance, resulting in choppy display, poor UI responsiveness, and other problems

- Development and debugging improvements

# ART Features

- Support for sampling profiler
- Historically, developers have used the Traceview tool (designed for tracing application execution) as a profiler. While Traceview gives useful information, its results on Dalvik have been skewed by the per-method-call overhead, and use of the tool noticeably affects run time performance.
- ART adds support for a dedicated sampling profiler that does not have these limitations. This gives a more accurate view of app execution without significant slowdown. Sampling support was added to Traceview for Dalvik in the KitKat release.



# ART Features

- Support for more debugging features
  - See what locks are held in stack traces, then jump to the thread that holds a lock.
  - Ask how many live instances there are of a given class, ask to see the instances, and see what references are keeping an object live.
  - Filter events (like breakpoint) for a specific instance.
  - See the value returned by a method when it exits (using “method-exit” events).
  - Set field watchpoint to suspend the execution of a program when a specific field is accessed and/or modified.

## **Application Framework**

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

## **Applications**

--All the Android application at the top layer.

# Basic Components of Android

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

Components	Description
Activities	They dictate the UI and handle the user interaction to the smartphone screen
Services	They handle background processing associated with an application.
Broadcast Receivers	They handle communication between Android OS and applications.
Content Providers	They handle data and database management issues.

Components	Description
Fragments	Represent a behavior or a portion of user interface in an Activity.
Views	UI elements that are drawn onscreen including buttons, lists forms etc.
Layouts	View hierarchies that control screen format and appearance of the views.
Intents	Messages wiring components together.
Resources	External elements, such as strings, constants and drawable pictures.
Manifest	Configuration file for the application.

# Hello World Program in Android





# New Project

Android Studio

## Configure your new project

Application name:

Company Domain:

saira\_000@example.com

Package name:

com.example.saira\_000

[Edit](#)

Project location:

C:\Users\saira\_000\AndroidStudioProjects



Please enter an application name (shown in launcher)

Previous

Next

Cancel

Finish





## Target Android Devices

## Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK API 23: Android 6.0 (Marshmallow)

Lower API levels target more devices, but have fewer features available.

By targeting API 23 and later, your app will run on approximately 4.7% of the devices that are active on the Google Play Store.

[Help me choose](#)

☐ Wear

Minimum SDK API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass

Minimum SDK Glass Development Kit Preview (API 19)

Previous

Next

Cancel

Finish

# Project Structure

- `AndroidManifest.xml`

This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

- `Java`

This contains the `.java` source files for your project. By default, it includes an `MainActivity.java` source file having an activity class that runs when your app is launched using the app icon.

# Project Structure

- `res/drawable`

This is a directory for drawable objects that are designed for high-density screens.

- `res/layout`

This is a directory for files that define your app's user interface.

- `res/values`

This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.

# Project Structure

- Build.gradle

This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

# Code Structure

- MainActivity.java

```
super.onCreate(savedInstanceState);  
setContentView(R.layout.activity_main);
```
- R.layout.activity\_main refers to the activity\_main.xml file located in the res/layout folder

# Code Structure

- Manifest File
  - You must declare all its components in a manifest.xml which resides at the root of the application project directory.
  - This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS.

# Code Structure : Manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

# Code Structure

- Manifest File
  - Here `<application>...</application>` tags enclosed the components related to the application.
  - `android:icon`
  - `<activity>` tag



# Intent-filter contents

- The action for the intent filter is named `android.intent.action.MAIN` -- indicate that this activity serves as the entry point for the application.
- The category for the intent-filter is named `android.intent.category.LAUNCHER` to indicate that the application can be launched from the device's launcher icon.

# String Reference

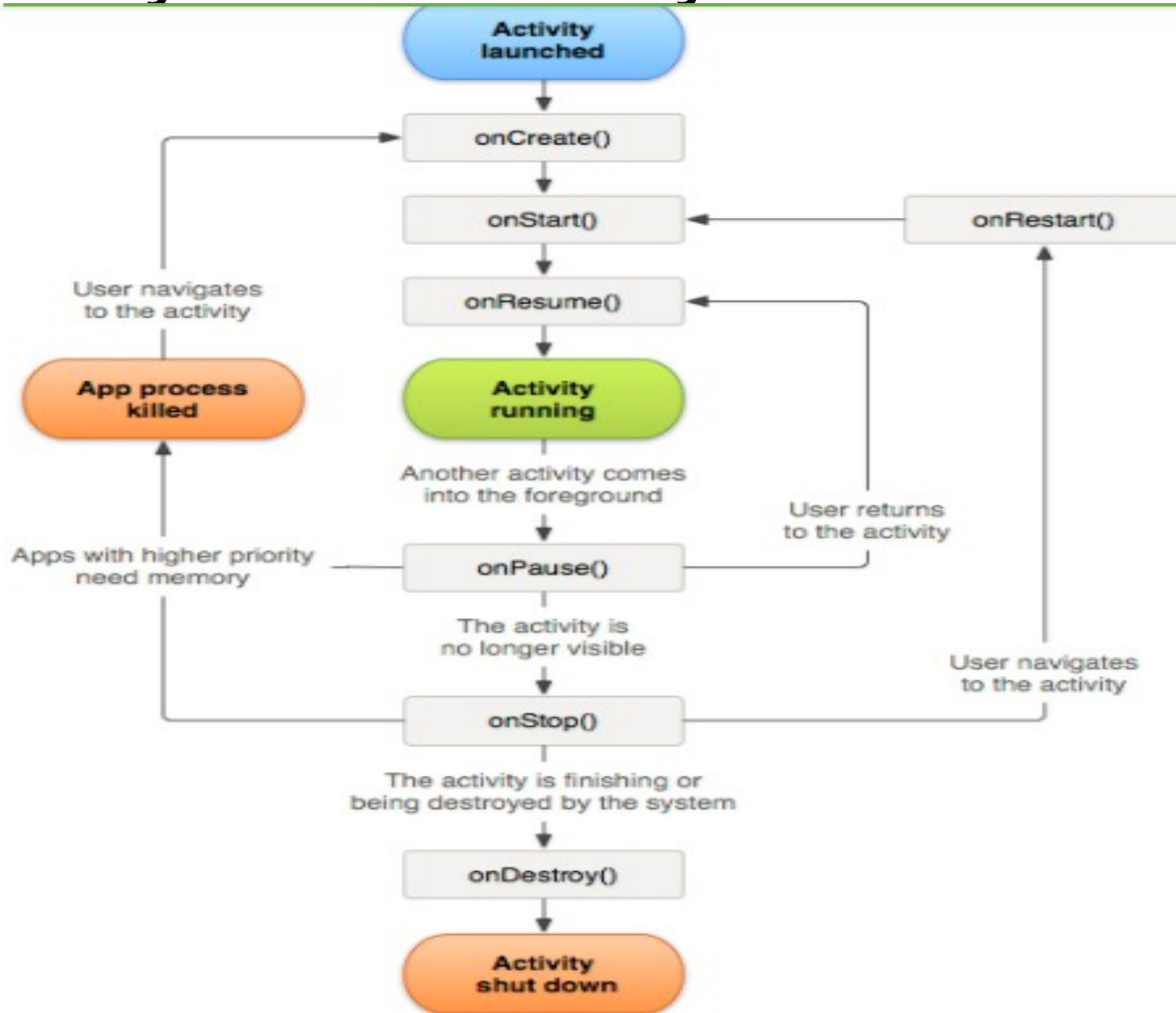
- The @string refers to the strings.xml file explained below. Hence, @string/app\_name refers to the app\_name string defined in the strings.xml file
- Strings.xml file

```
<resources>  
<string name="app_name">Welcome</string>  
</resources>
```

# Different Android application components

- <activity>elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

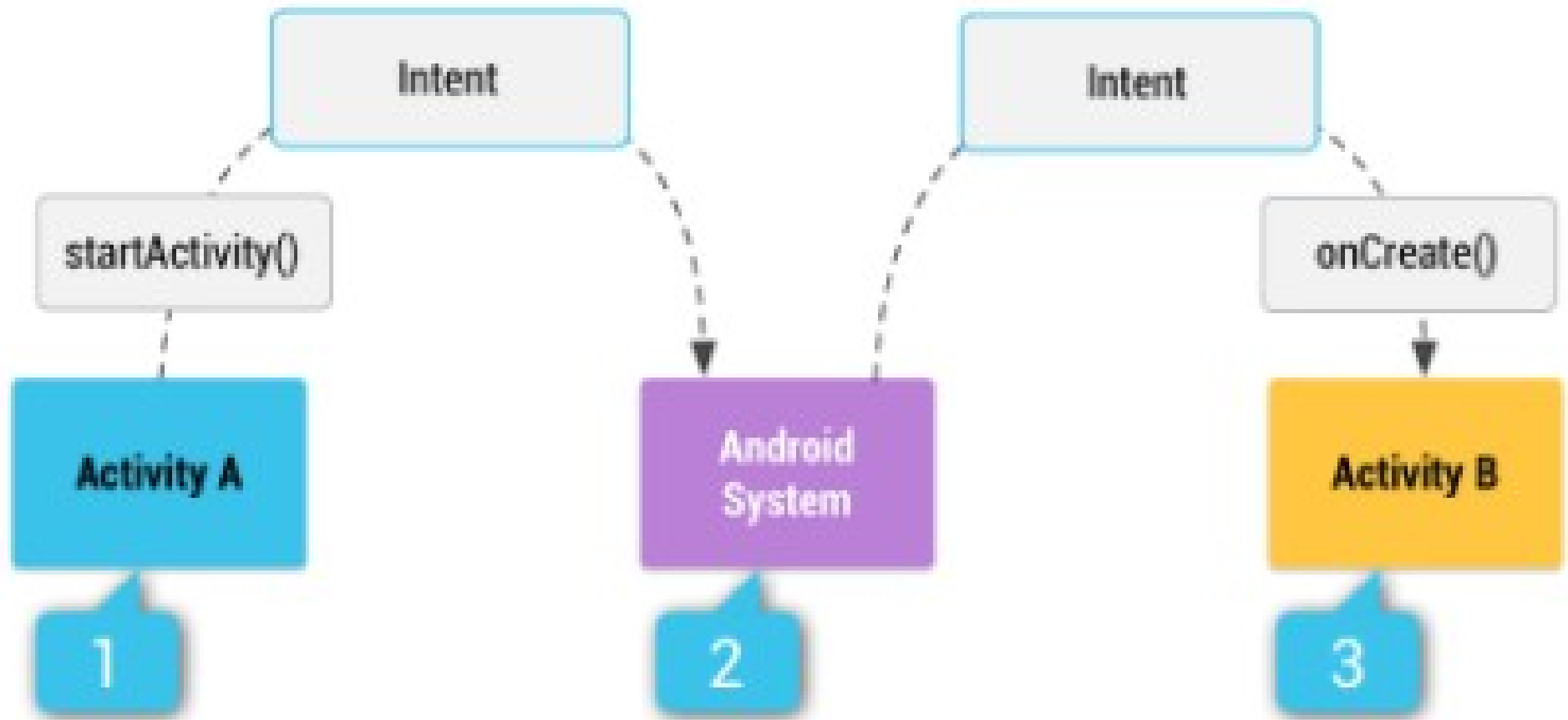
# Life Cycle of Activity



# Intent & Intent Filter

- Explicit
- Implicit

# Intent & Intent Filter



# System Permissions

- To maintain security for the system and users, Android requires apps to request permission before the apps can use certain system data and features. Depending on how sensitive the area is, the system may grant the permission automatically, or it may ask the user to approve the request.

# Compatibility

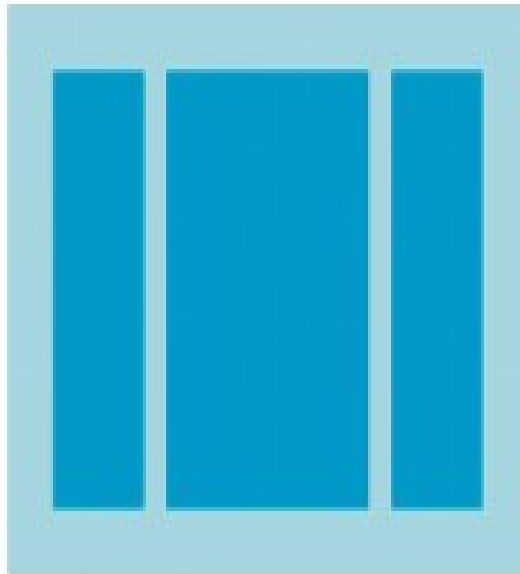
- Device
- App



# Layouts

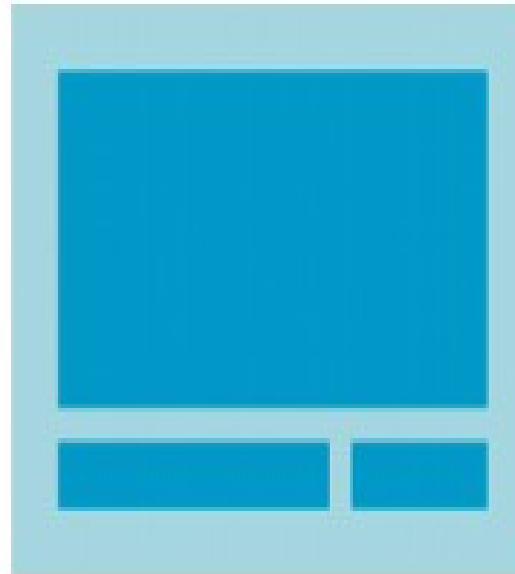
- Linear
- Table Layout
- Relative Layout

## Linear Layout



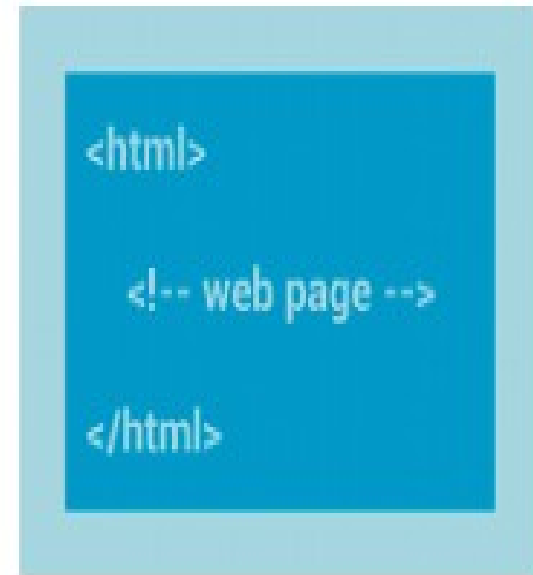
A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.

## Relative Layout



Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

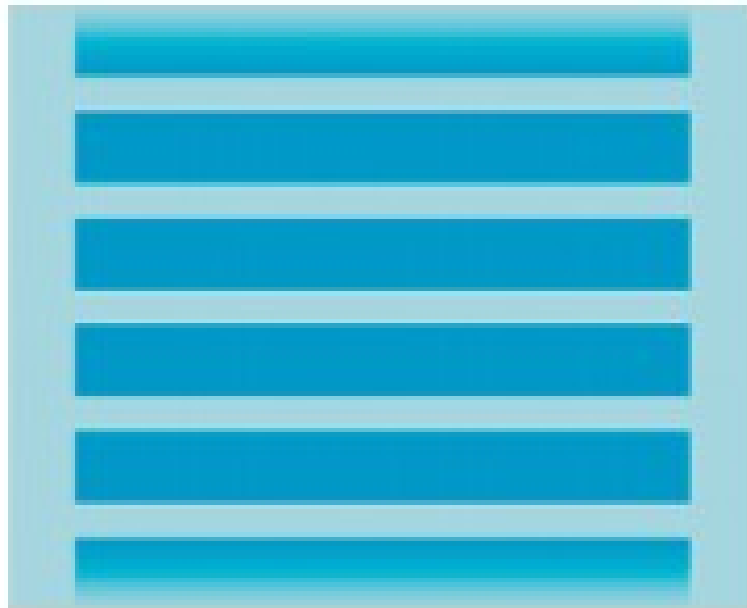
## Web View



Displays web pages.

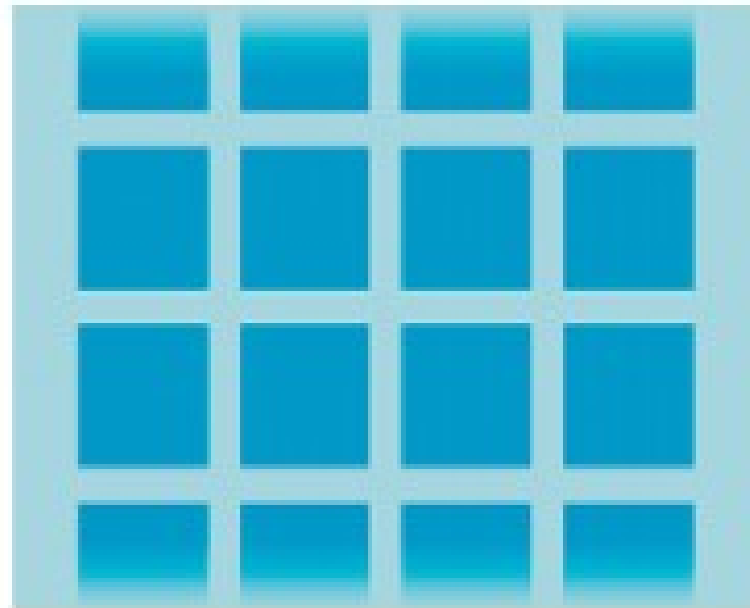
Not-pre-determine - subclasses AdapterView to populate the layout with views at runtime.

## List View



Displays a scrolling single column list.

## Grid View



Displays a scrolling grid of columns and rows.

# RecyclerView

## Recycler View

**The Martian**

2015

Science Fiction & Fantasy

**Mission: Impossible Rogue Nation**

2015

Action

**Up**

2009

Animation

**Star Trek**

2009

Science Fiction



## Recycler View

**Mad Max: Fury Road**

2015

Action & Adventure

**Inside Out**

2015

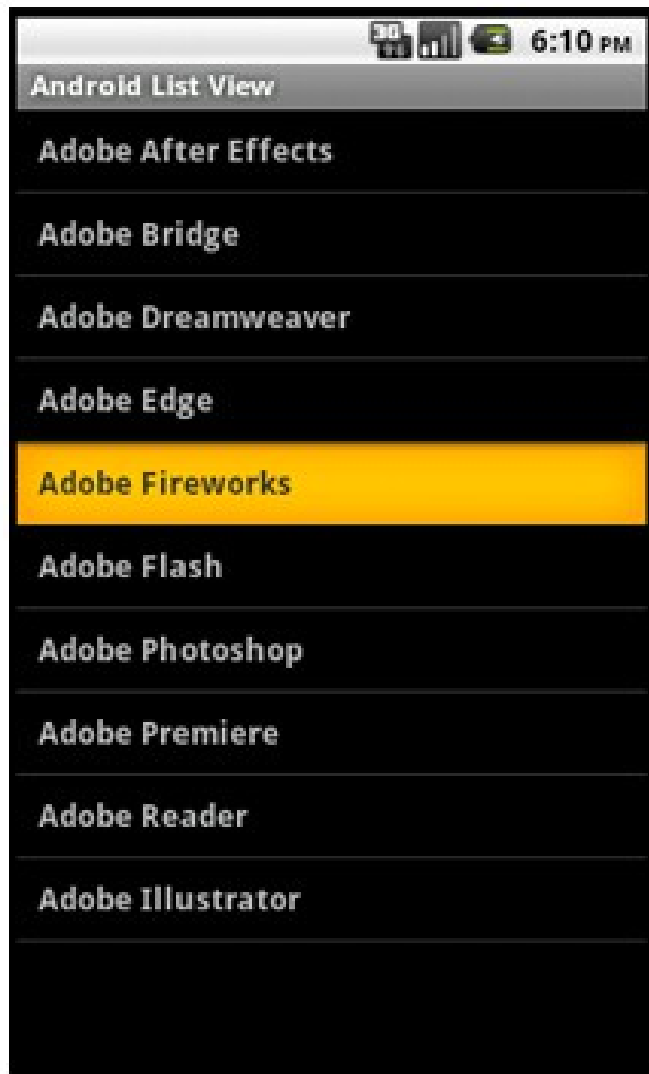
Animation, Kids & Family

**Star Wars: Episode VII - The Force Awakens**

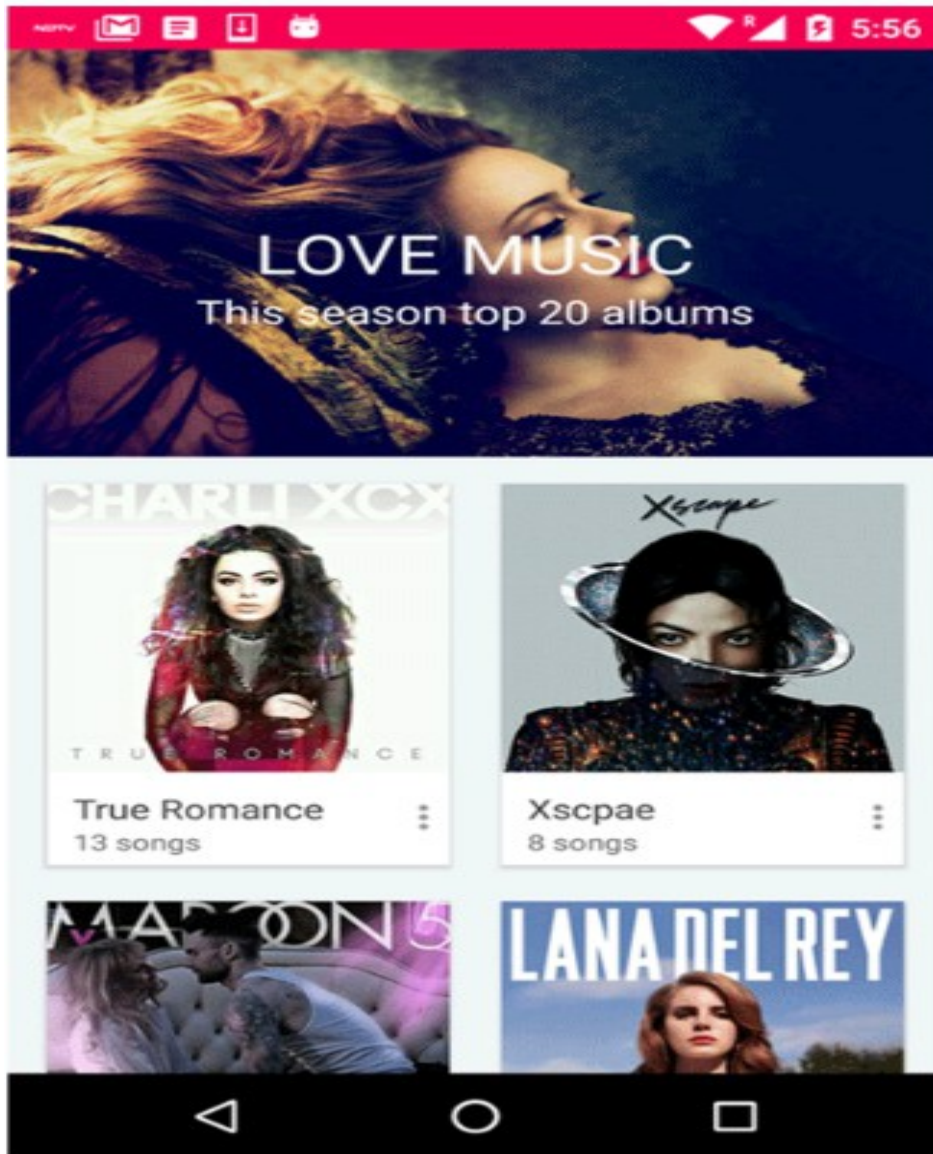
2015

Action

# ListView

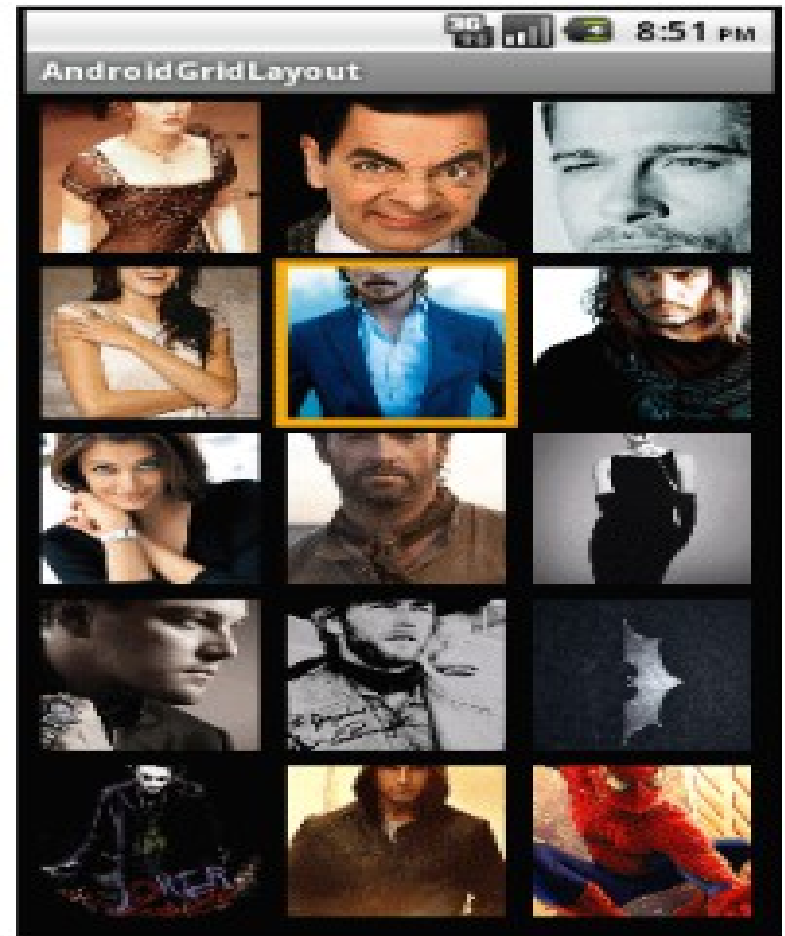


# CardView & RecyclerView



# Grid Layout

- GridView layout is one of the most useful layouts in android. GridView is mainly useful when we want to show data in grid layout like displaying images or icons. This layout can be used to build applications like image viewer, audio or video players in order to show elements in grid manner.



# Web View

- Android's WebView allows you to integrate a webpage as a part of the app. WebView comes with all the features that of a desktop browser like managing history, cookies, HTML5 support and lot more. Using webview you can build very cool apps like integrating HTML5 games in the app.





# Input Controls

- Buttons
- CheckBox
- RadioButton
- Toggle Button
- Spinners
- Input Events
- Menus
- Toast
- Dialogs
- Styles
- Themes

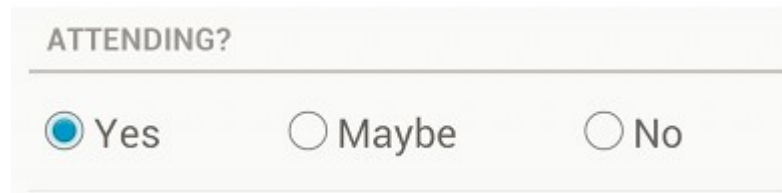
# Toggle Button

- A ToggleButton displays checked/unchecked states as a button. It is basically an on/off button with a light indicator.



# Radio Button

A RadioButton has two states: either checked or unchecked. This allows the user to select one option from a set.

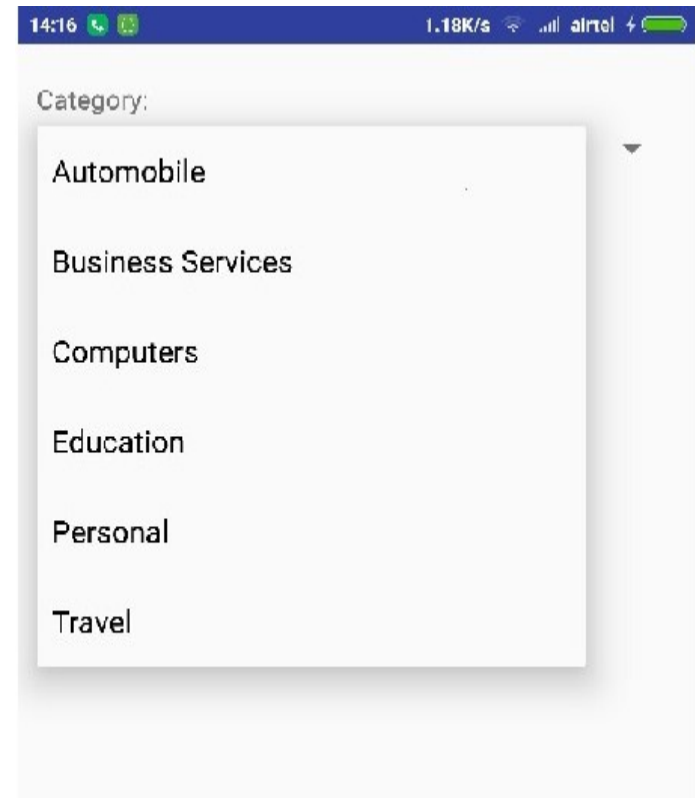


ATTENDING?

☒ Yes ☐ Maybe ☐ No

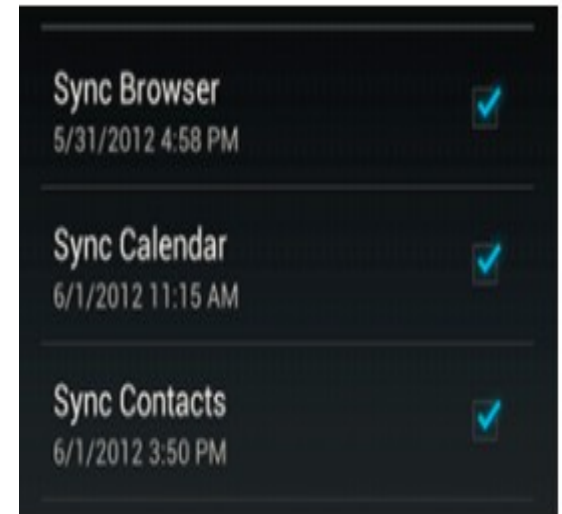
# Spinner

- Spinner allows you to select an item from a drop down menu



# Check Box

- Checkboxes allow the user to select one or more options from a set. Typically, you should present each checkbox option in a vertical list.



# Input Events

- On Android, there's more than one way to intercept the events from a user's interaction with your application. When considering events within your user interface, the approach is to **capture the events from the specific View object** that the user interacts with. The View class provides the means to do so.

# Event Handling

There are following three concepts related to Android Event Management –

- **Event Listeners** – An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
- **Event Listeners Registration** – Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.
- **Event Handlers** – When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

# Event Listeners & Event Handlers

Event Handler	Event Listener & Description
<code>onClick()</code>	<b><code>OnClickListener()</code></b>  This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use <code>onClick()</code> event handler to handle such event.
<code>onLongClick()</code>	<b><code>OnLongClickListener()</code></b>  This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use <code>onLongClick()</code> event handler to handle such event.



onFocusChange()

### **OnFocusChangeListener()**

This is called when the widget loses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.

onKey()

### **OnFocusChangeListener()**

This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.

onTouch()

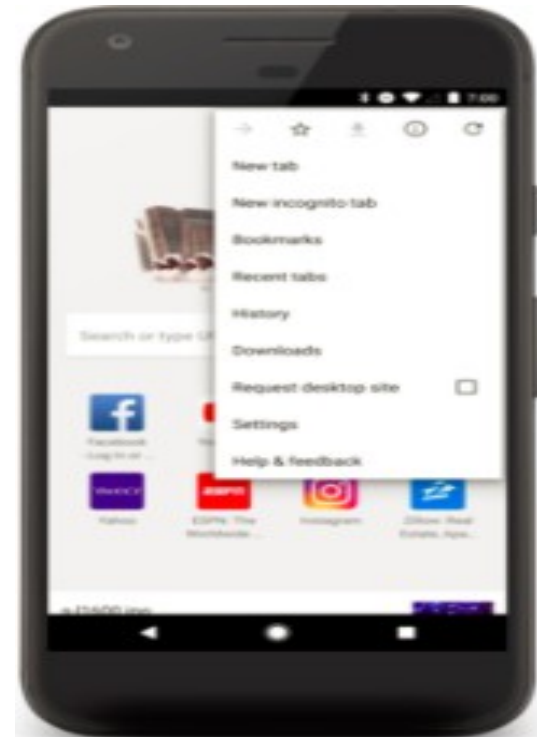
### **OnTouchListener()**

This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.

<code>onMenuItemClick()</code>	<b>OnMenuItemClickListener()</b>  This is called when the user selects a menu item. You will use <code>onMenuItemClick()</code> event handler to handle such event.
<code>onCreateContextMenu()</code>	<b>onCreateContextMenuListener()</b>  This is called when the context menu is being built(as the result of a sustained "long click")

# Menus

- Menus are a common user interface component in many types of applications. To provide a familiar and consistent user experience
- A) Options menu and app bar

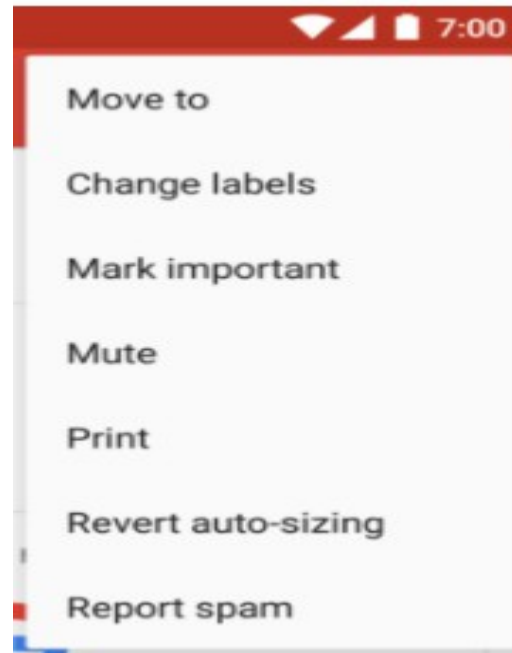


# Menus

- Context menu and contextual action mode

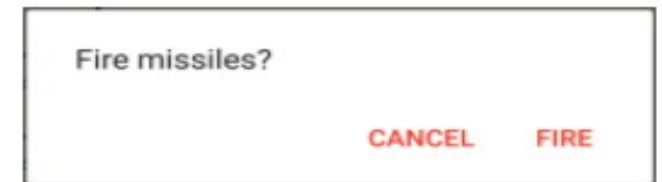
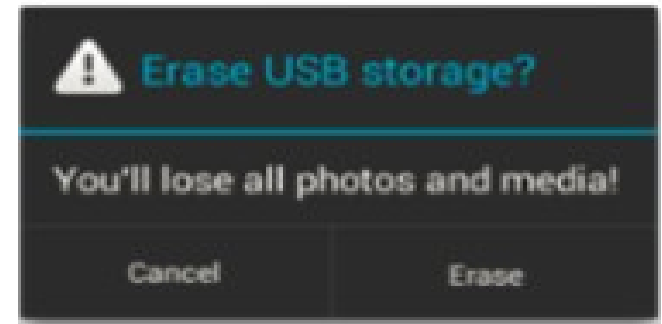


- Popup menu



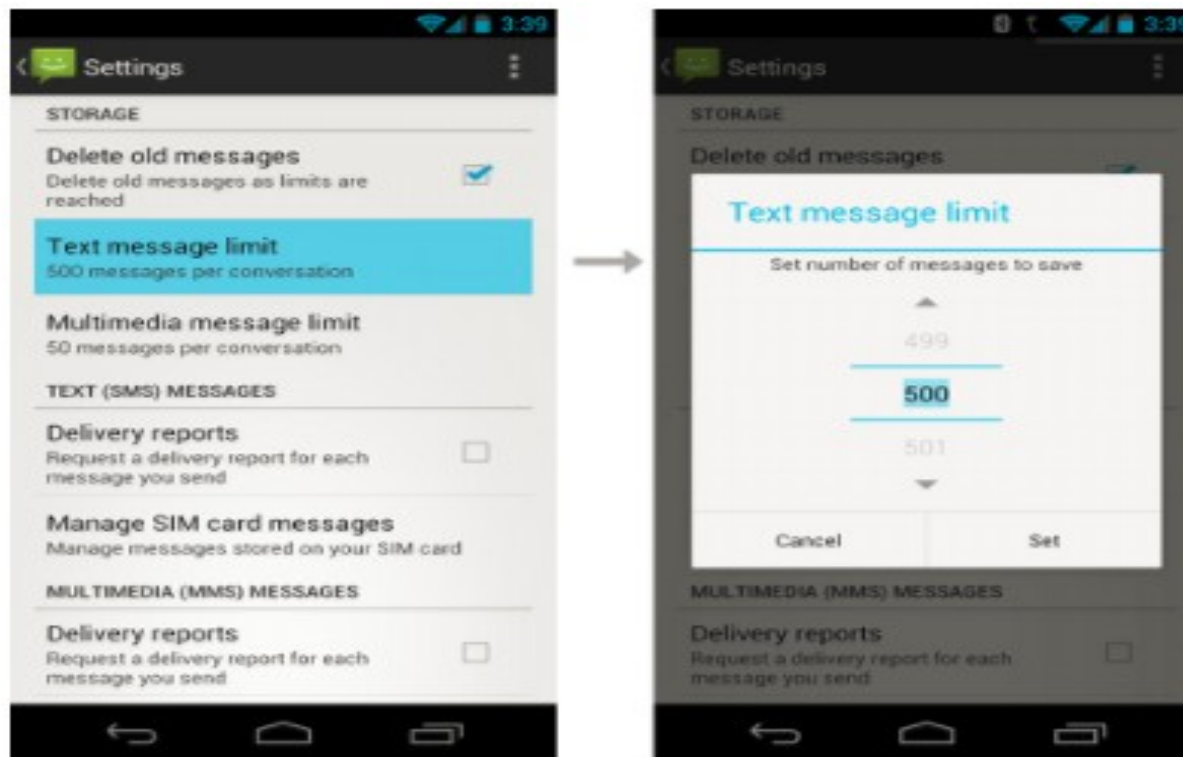
# Dialogs

- A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.
- AlertDialog
- DatePickerDialog or TimePickerDialog



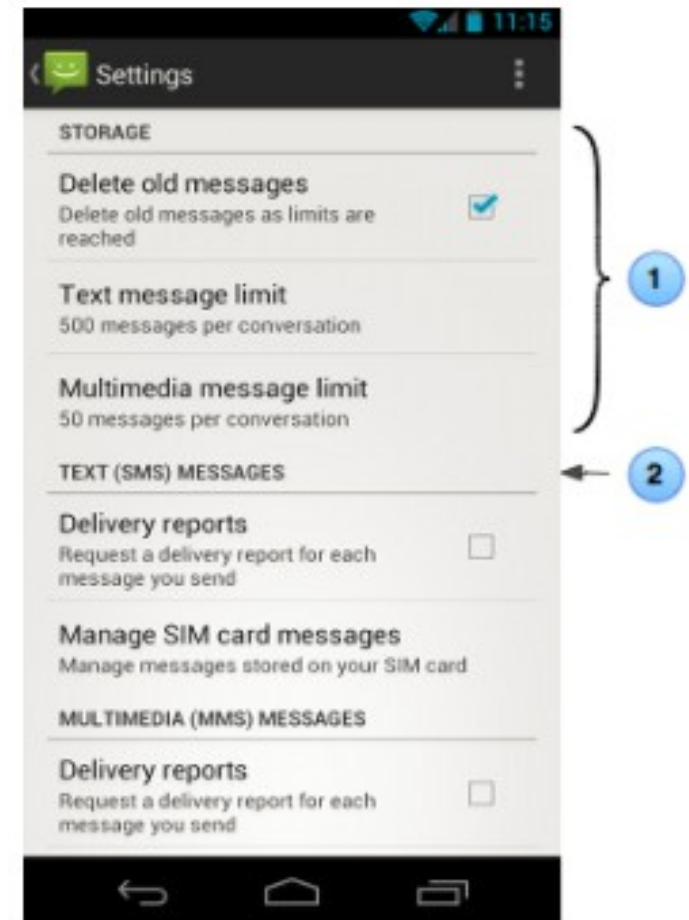
# Settings

- Apps often include settings that allow users to modify app features and behaviors.



# Settings

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="@string/pref_sms_storage_title"
        android:key="pref_key_storage_settings">
        <CheckBoxPreference
            android:key="pref_key_auto_delete"
            android:summary="@string/pref_summary_auto_delete"
            android:title="@string/pref_title_auto_delete"
            android:defaultValue="false"... />
        <Preference
            android:key="pref_key_sms_delete_limit"
            android:dependency="pref_key_auto_delete"
            android:summary="@string/pref_summary_delete_limit"
            android:title="@string/pref_title_sms_delete"... />
        <Preference
            android:key="pref_key_mms_delete_limit"
            android:dependency="pref_key_auto_delete"
            android:summary="@string/pref_summary_delete_limit"
            android:title="@string/pref_title_mms_delete"... />
    </PreferenceCategory>
    ...
</PreferenceScreen>
```

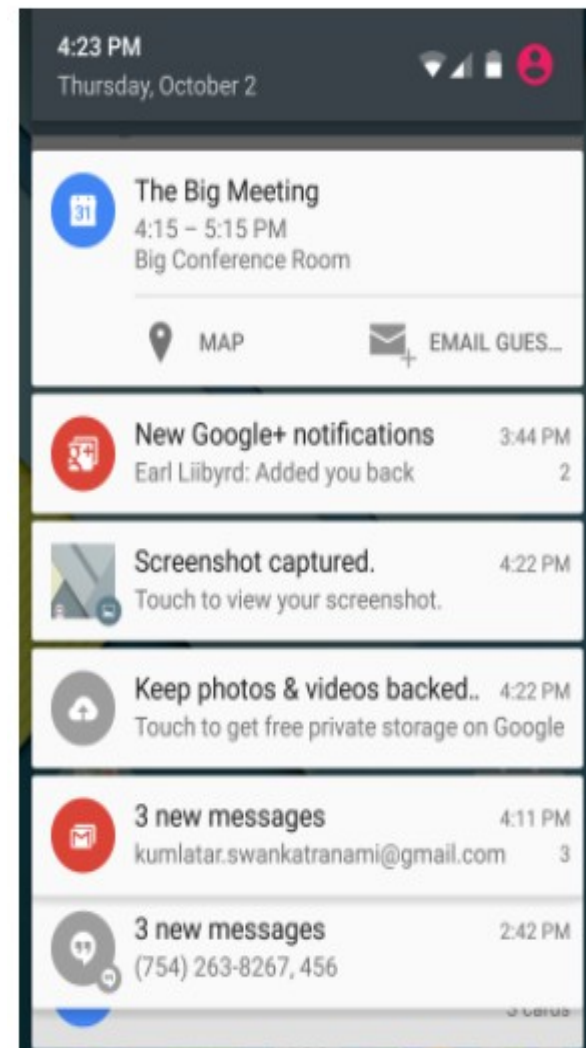


# Notifications

- A notification is a message you display to the user outside of your app's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.



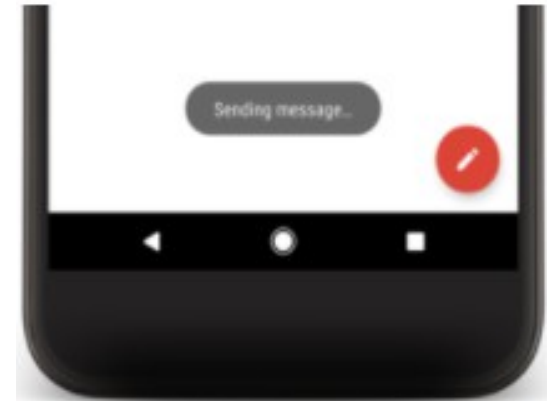
Figure 1. Notifications in the notification area.





# Toasts

- A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.



# Tooltips

- A tooltip is a small descriptive message that appears near a view when users long press the view or hover their mouse over it. This is useful when your app uses an icon to represent an action or piece of information to save space in the layout.



# Styles and Themes

- A style is a collection of attributes that specify the look and format for a View or window. A style can specify attributes such as height, padding, font color, font size, background color, and much more. A style is defined in an XML resource that is separate from the XML that specifies the layout.

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textColor="#00FF00"

android:text="@string/hello" />

# Styles and Themes

- Apply a theme to an activity or app

To set a theme for all the activities of your app, open the `AndroidManifest.xml` file and edit the `<application>` tag to include the `android:theme` attribute with the style name

- e.g

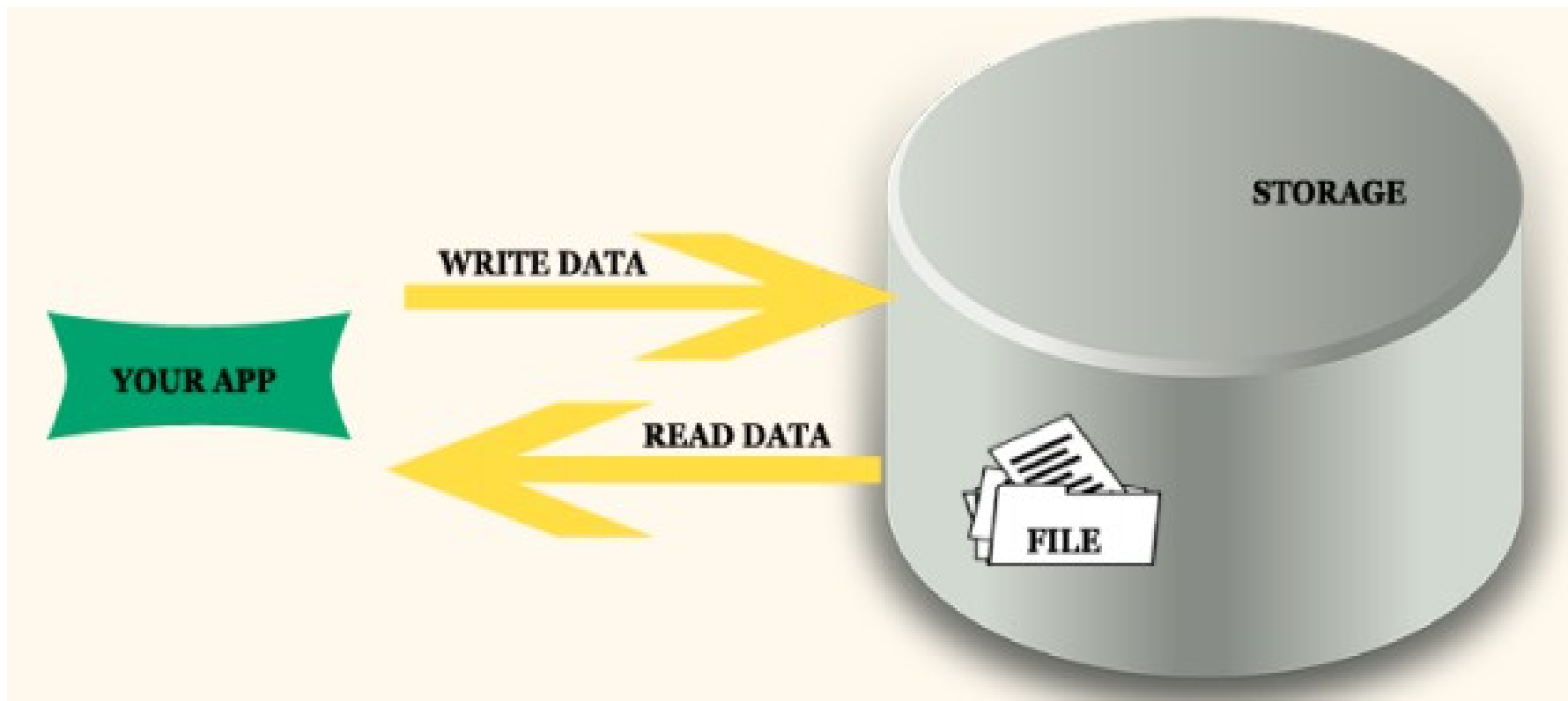
```
<application android:theme="@style/CustomTheme">
```

# Data Storage

- Shared Preferences
- Internal Storage
- External Storage
- SQLite Database
- Content Providers
- Remote Databases

# Internal Storage

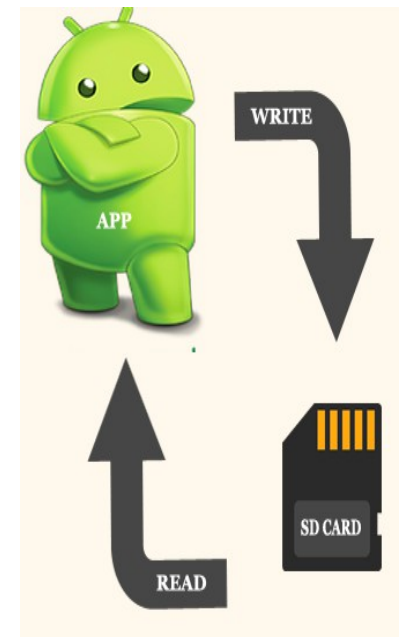
- primary memory of your phone.



# Internal Storage

1. The stored data in memory is allowed to read and write files.
2. When files are stored in internal storage these file can only be accessed by the application itself not by other applications.
3. These files in storage exist till the application stays over the device, as you uninstall associated files get removed automatically.
4. The files are stored in directory data/data which is followed by the application package name.
5. User can explicitly grant the permission to other apps to access files.

# External Storage



- secondary memory/SD card of your phone.

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

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
```



# Shared Preferences

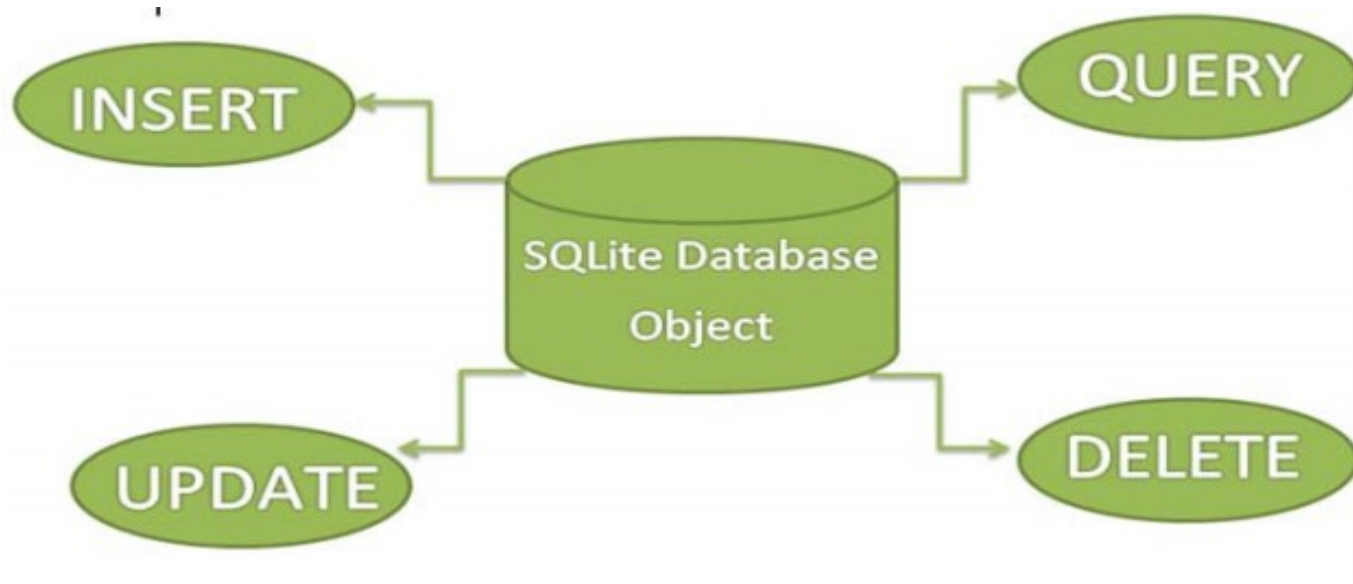
- Shared Preference in Android are used to save data based on key-value pair.

Shared Preference can be used to save primitive data type: string, long, int, float and Boolean.

- A preference file is actually a xml file saved in internal memory of device

# SQLite Database

- SQLite is a Structure query base database, open source, light weight, no network access and standalone database. It support embedded relational database features.



# SQLite Database

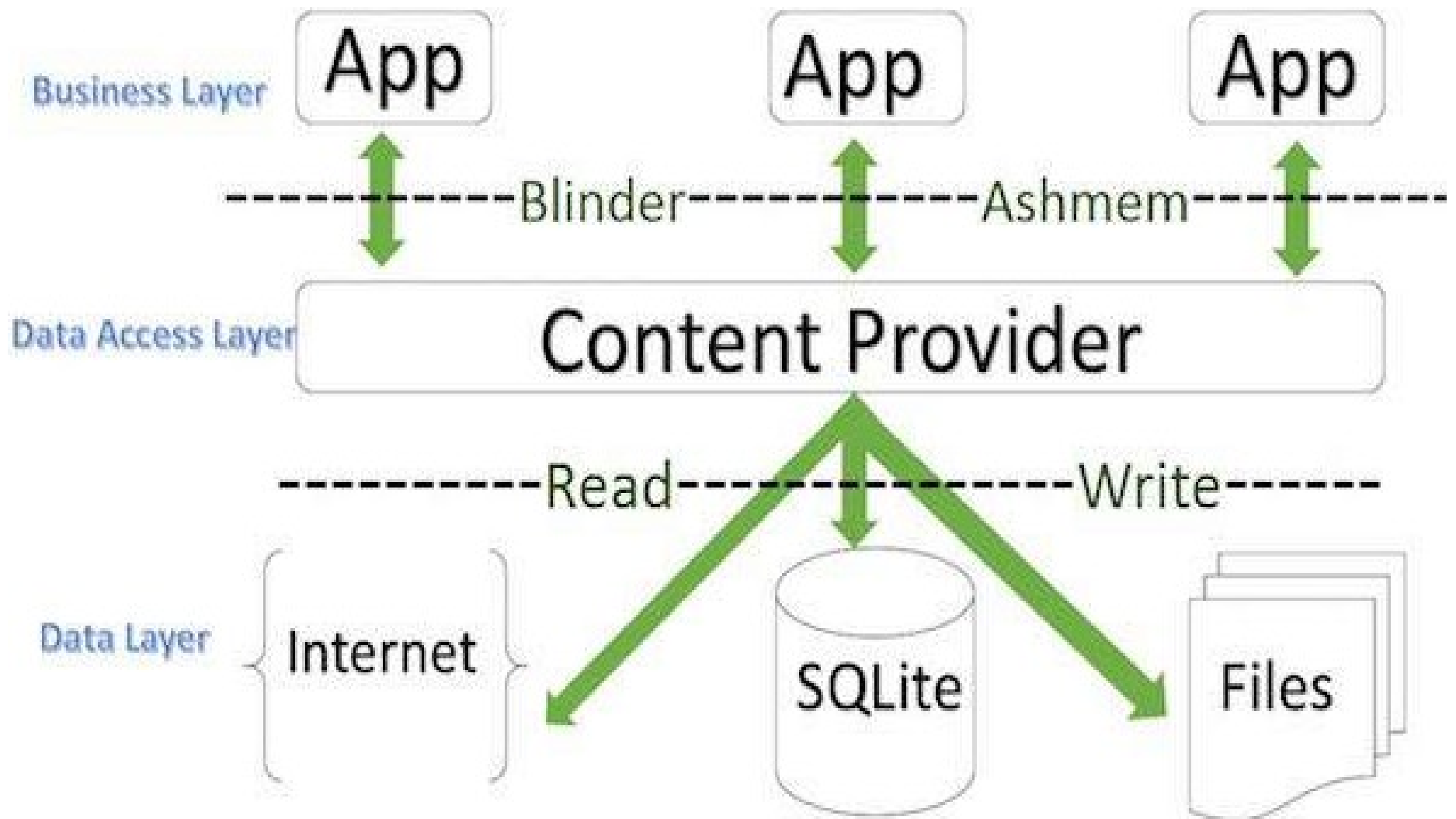
Whenever an application needs to store large amount of data then using **sqlite is more** preferable than other repository system like SharedPreferences or saving data in files.

- Android has built in SQLite database implementation. It is available locally over the device(mobile & tablet) and contain data in text format.
- It carry light weight data and suitable with many languages. So, it doesn't required any administration or setup procedure of the database

- Important Note – The database created is saved in a directory:

data/data/APP\_Name/databases/DATABASE\_NAME.

# •Content Providers



# •Content Providers

- Binder
- Binder is an Android-specific interprocess communication mechanism, and remote procedure call system similar to Dbus.
- ashmem

ashmem - Android shared memory

implementation is in mm/ashmem.c

# •Content Providers

- A content provider manages access to a central repository of data.
- Typically you work with content providers in one of two scenarios;
  - 1) You may want to implement code to access an existing content provider in another application,
  - 2) You may want to create a new content provider in your application to share data with other applications.

# Advance Components of Android

- Web App
- JSON parsing
- Google Map
- GPS
- Sensors
- Bluetooth / Wi-Fi connectivity



# Multimedia, Animation & Graphics

- Playing Audio / Video
- Rotate Animation
- Fade In / Fade Out Animation
- Zoom Animation
- Scale Animation
- 2 D & 3 D Graphics