

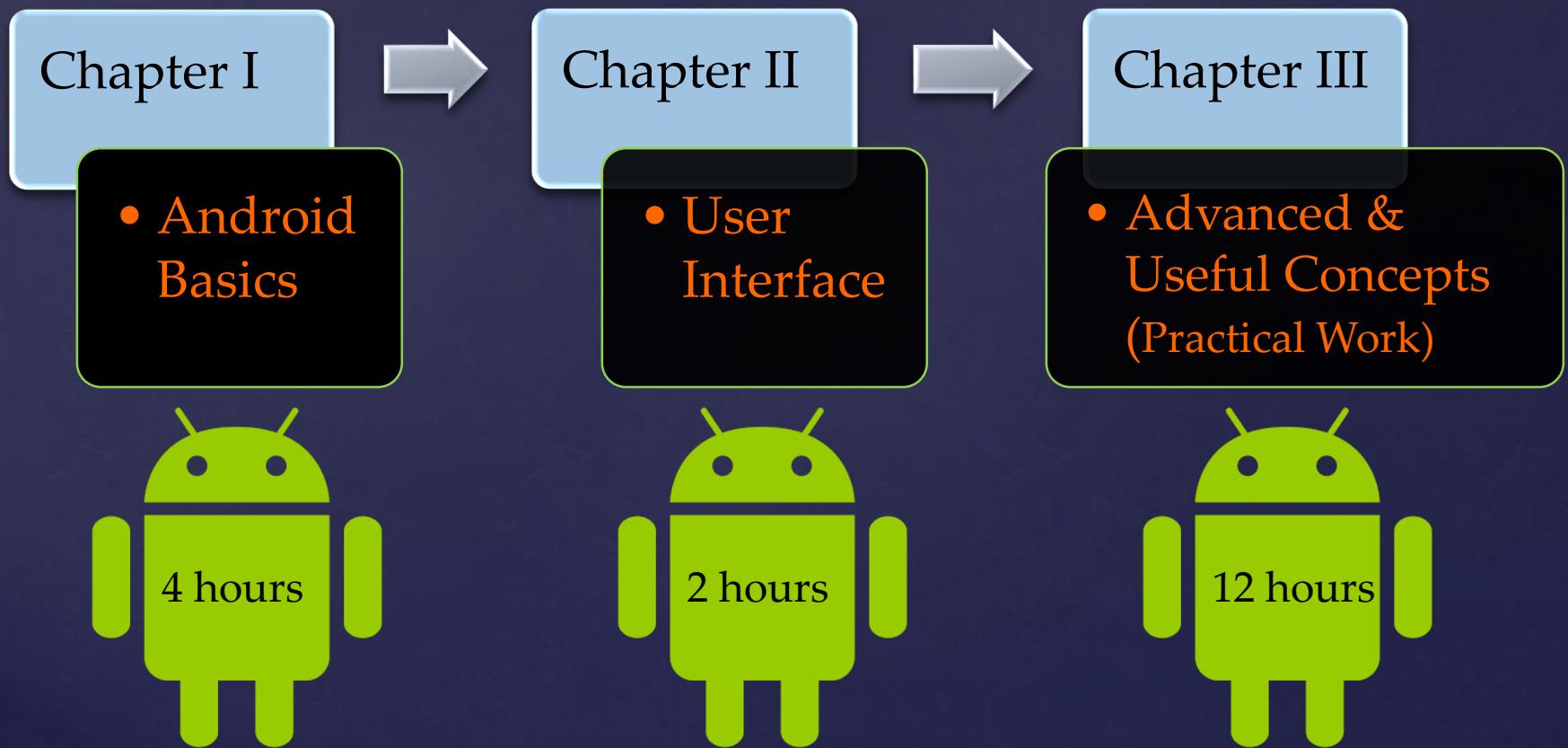
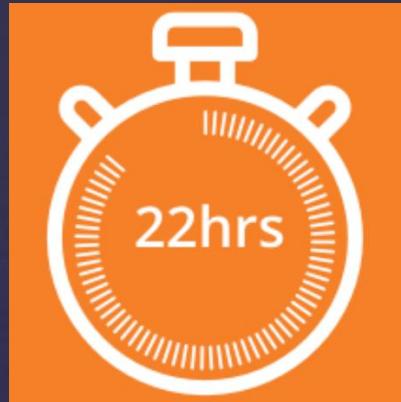
# ANDROID COURSE

By: Mr Adnane Ayman



Université Internationale  
de Casablanca

UNIVERSITÉ RECONNUE PAR L'ÉTAT





2 hours

Final Exam

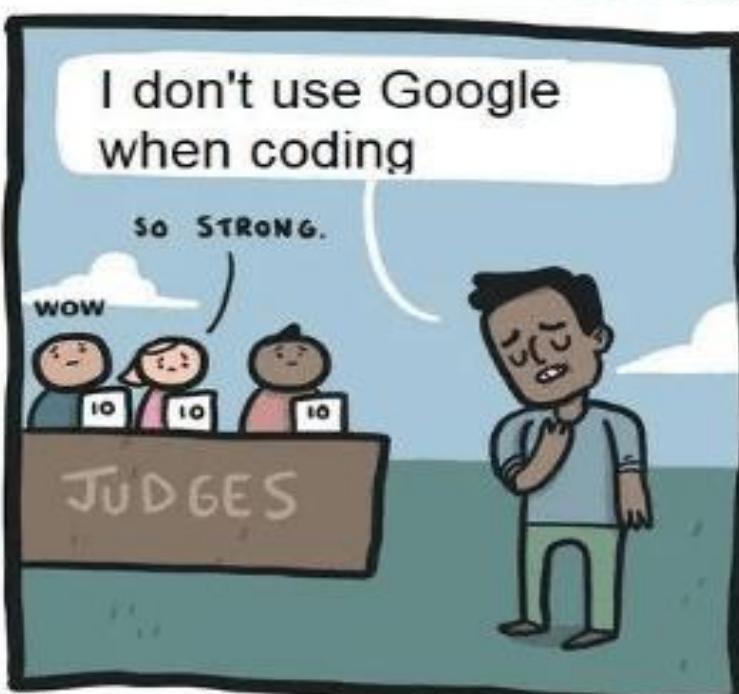


2 hours



#Android

# Facts



# Chapter I: Android Basics



1. Overview [What is android ? / why android? /features]
2. Environment Setup ( IDEs ):
3. Architecture
4. Application Components
5. Running my first application
6. Resources
7. Activities
8. Intents

# Chapter II: User Interface



1. UI layouts
2. UI Controls
3. Event handlings

# Chapter III: Advanced & Useful Concepts: Practical Work



1. Alert Dialogs
2. Auto Complete
3. Sending SMS
4. Sending Emails
5. Google maps (TP markers)
6. Progress Bars
7. Shared Preferences
8. SQLite / ORMs (Sugar)
9. Text to speech
10. Firebase(FirebaseAuthenticationManager + FirebaseDatabase  
baseManager)

# INTRODUCTION TO ANDROID AND ANDROID STUDIO

{ Mr Adnane Ayman



# What Will We Learn Today

## Android Basics:

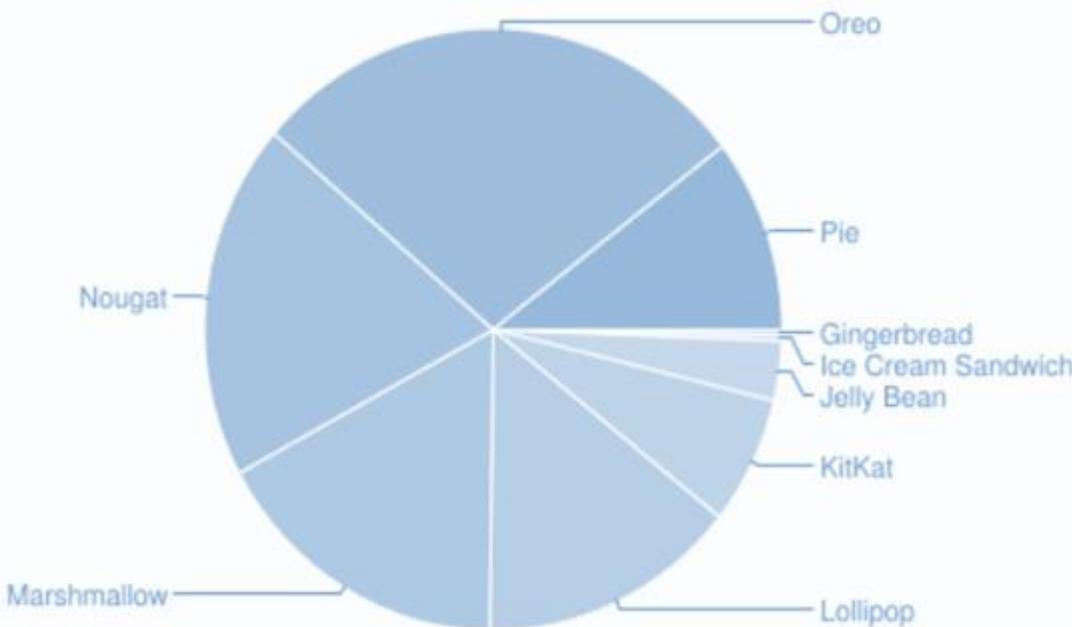
- ¶ *Overview [What is android ? / why android? /features]*
- ¶ *Environment Setup ( IDEs ):*
- ¶ *Architecture*
- ¶ *Application Components*
- ¶ *Running my first application*
- ¶ *Resources*
- ¶ *Activities*
- ¶ *Intents*

# *What is android ?*



Android is the world's most popular and dominant mobile operating system. It's based on Linux and is open-source. It runs on a wide variety of hardware, including smartphones, smart watches, cars, televisions, digital cameras, game consoles and more. It was founded by Andy Rubin and three others in October 2003 and got acquired by Google in August 2005.

# Distribution dashboard



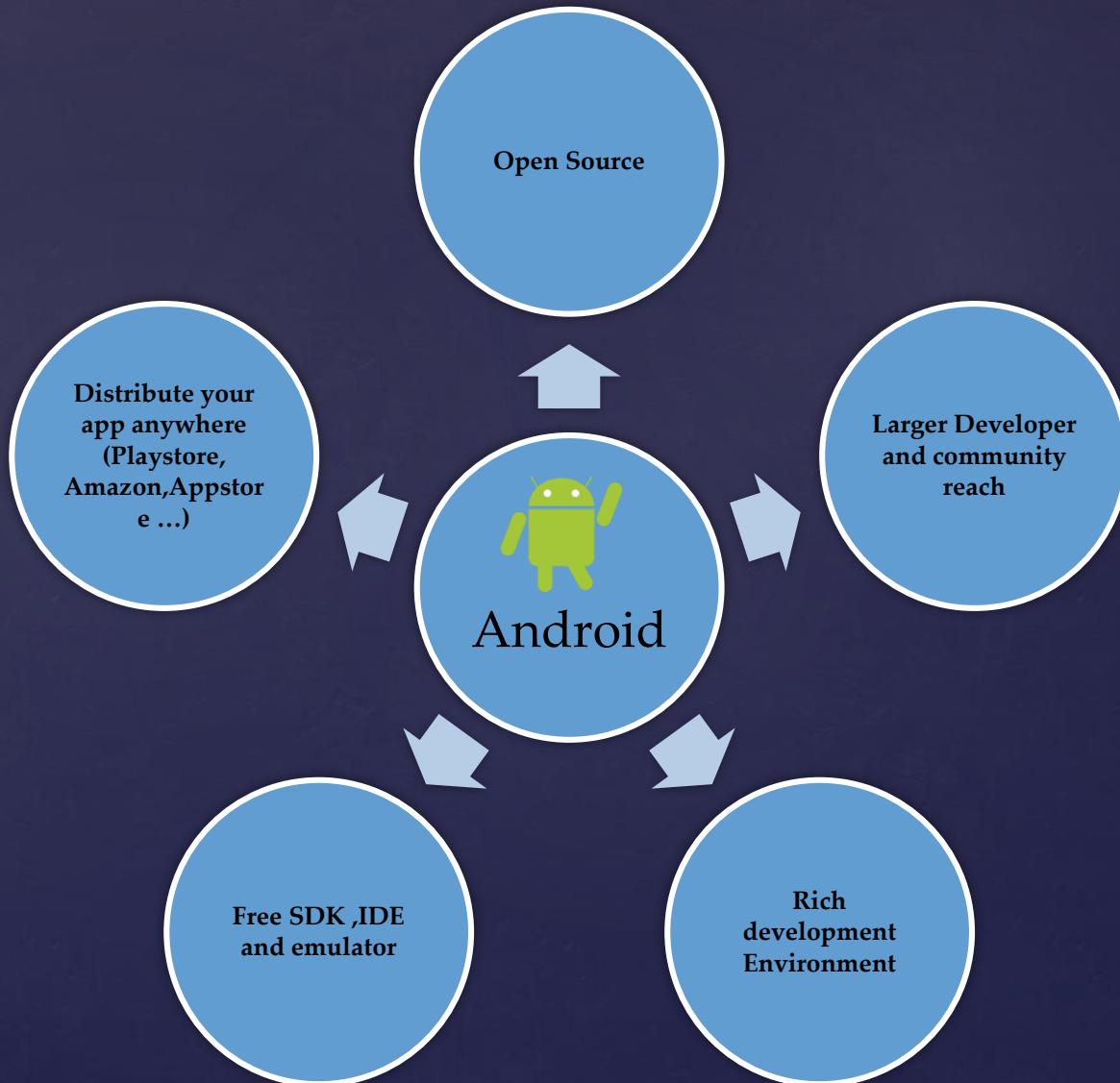
Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

# Worldwide Market Share



With Over 2.5 Billion monthly active users

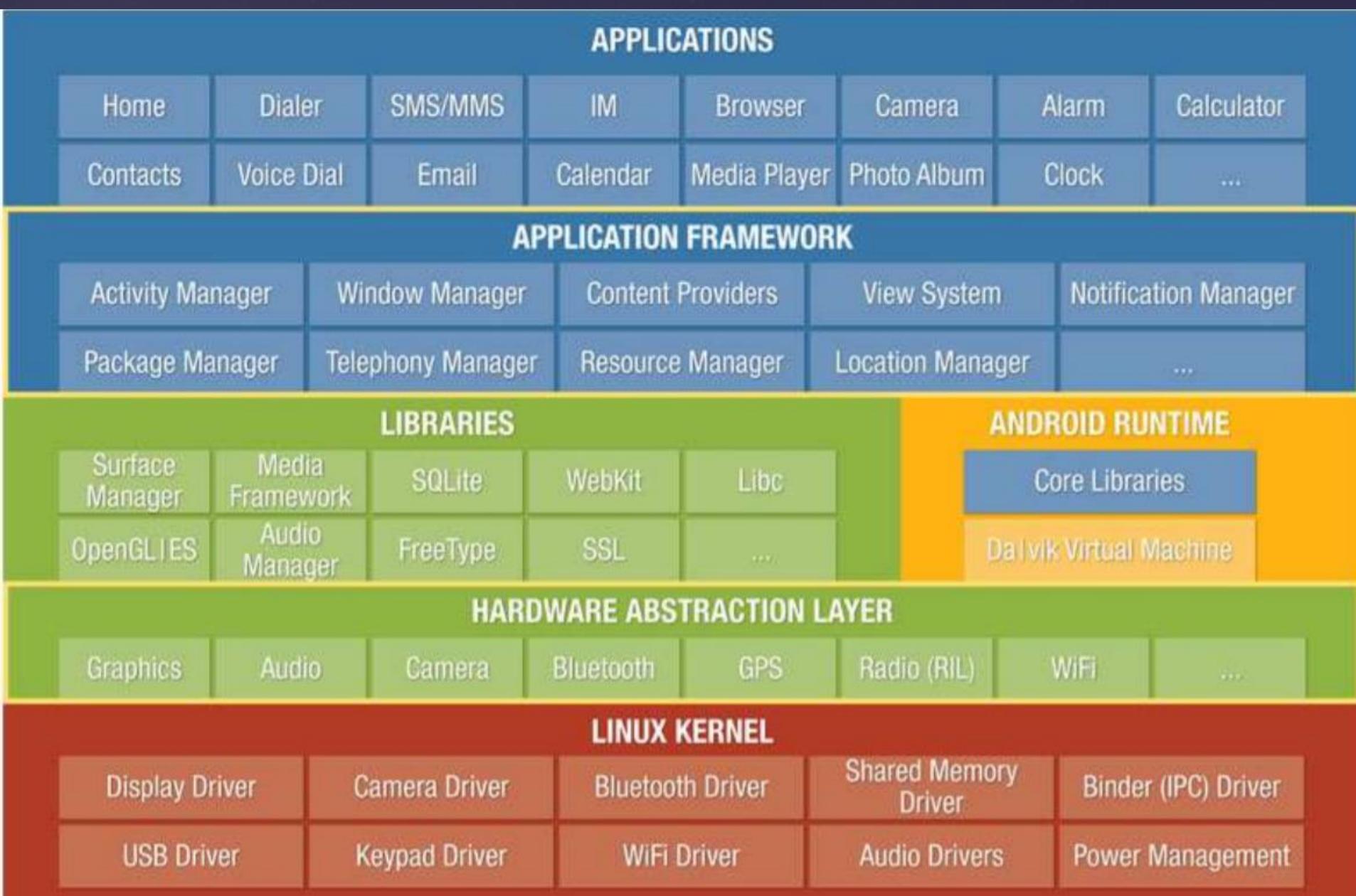
# Why Android?



# Features

Feature & Description	Feature & Description
<b>Beautiful UI</b> Android OS basic screen provides a beautiful and intuitive user interface.	<b>Multi-touch</b> Android has native support for multi-touch which was initially made available in handsets.
<b>Connectivity</b> GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.	<b>Multi-tasking</b> User can jump from one task to another and same time various application can run simultaneously.
<b>Storage</b> SQLite, a lightweight relational database, is used for data storage purposes.	<b>GCM</b> Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
<b>Media support</b> MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, SVG ...	<b>Android Beam</b> A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.
<b>Messaging</b> SMS and MMS	
<b>Web browser</b> Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.	

# Android Platform Overview



# Android Platform Overview

Android is actually a system of the Linux family, for once without the GNU tools. The OS is based on:

- ¶ A Linux kernel (and its drivers)
- ¶ a virtual machine: Dalvik Virtual Machine
- ¶ applications (browser, contact management, telephony application ...)

[Dalvik] is the name of the open-source virtual machine used on Android systems. This virtual machine is running .dex files and is not compatible with a JVM of the type Java SE or even Java ME

# *What Do I Need To Build An Android App?*



- Java Programming Language & XML
- Android SDK & SDK Tools
- Android Studio
- The desire to learn

# *Environment Setup ( IDEs ):*

You can start the development of your Android application on one of the following operating systems:

- Microsoft Windows XP or later.
- Mac OS X 10.5.8 or later with Intel chip.
- Linux, including GNU C Library 2.7 or later.

Second, all the tools needed to develop Android apps are available for free and can be downloaded from the web. Below is a list of the software you will need before you start programming your Android application.

- Java JDK5 or later
- Android Studio

# *Environment Setup ( IDEs ):*

You can download the latest version of Java JDK from Oracle's Java site

– Java SE Downloads. You will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally set PATH and JAVA\_HOME environment variables to refer to the directory that contains **java** and **javac**, typically **java\_install\_dir/bin** and **java\_install\_dir** respectively.

## Android IDEs

There are so many sophisticated Technologies are available to develop android applications, the familiar technologies, which are predominantly using tools as follows

- Android Studio
- Eclipse IDE(Deprecated)

We are going to freeze to death if we don't do something



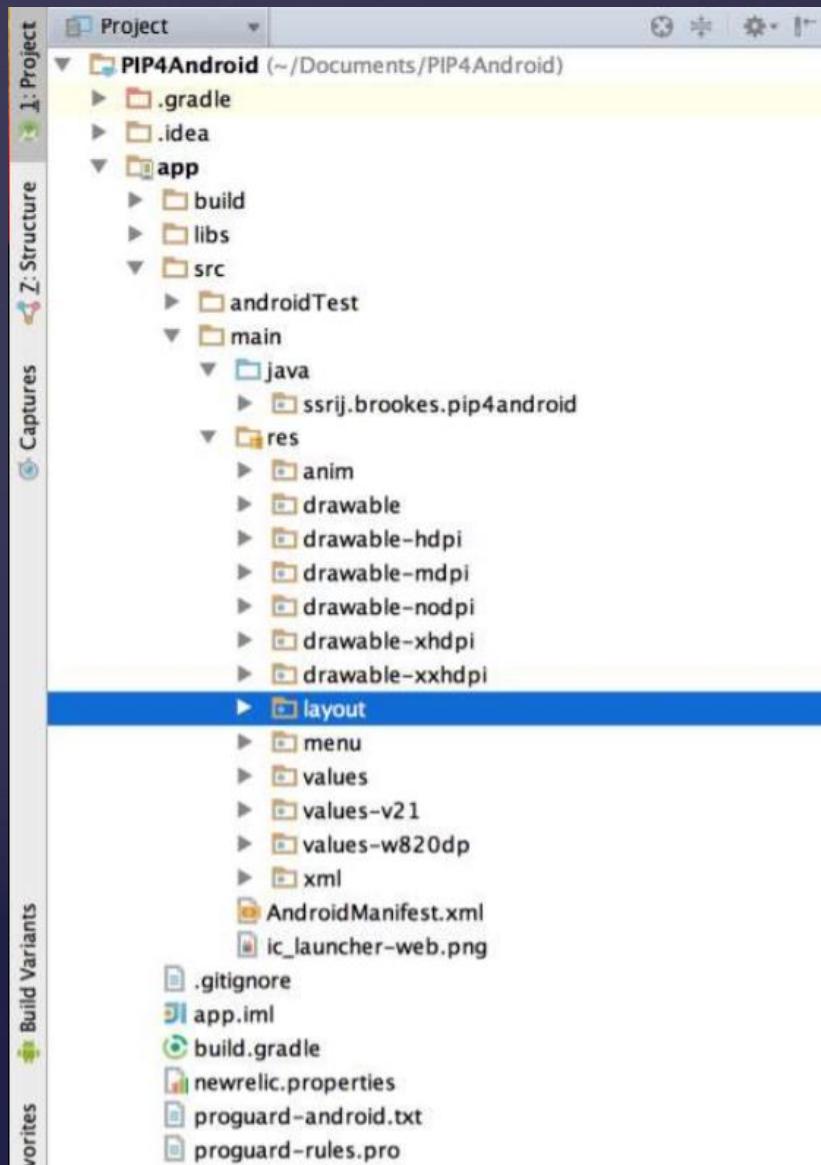
We only have one option left



Run Android Studio



# Architecture

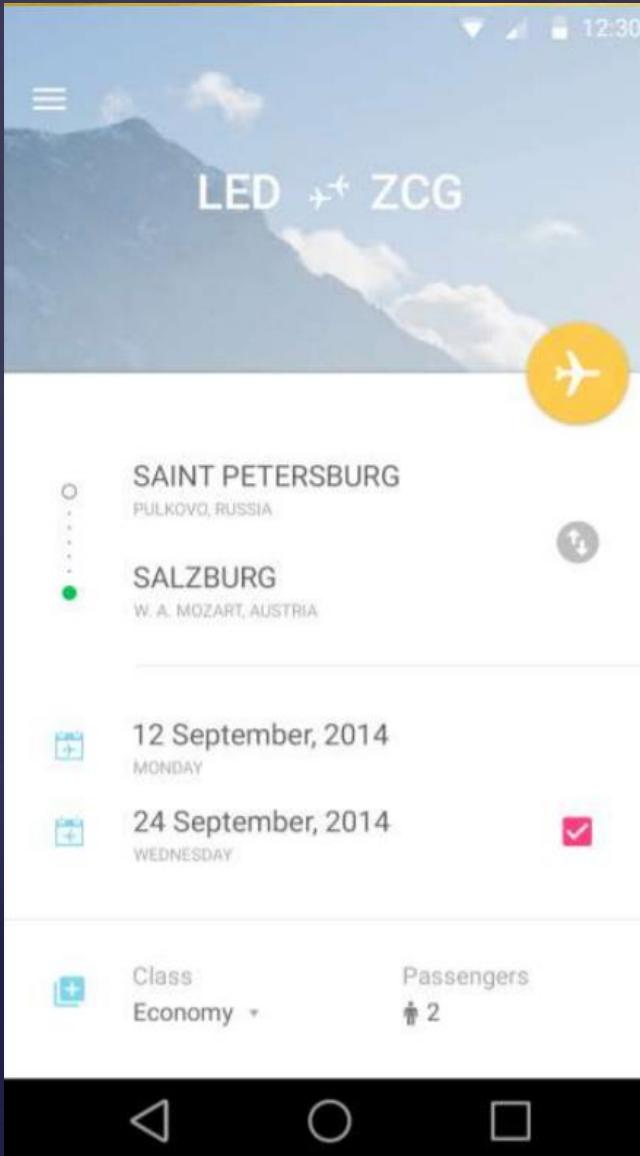


- Java: Java class files containing app logic
- Res: Different resource files
  - Anim: Animation resource files
  - Drawable: Images
  - Drawable-Xdpi: Images depending on screen density
- Layout: App layout files
- Menu: Layout menu files
- Values: Value files (strings, colors, arrays, etc)
- Values-vX: Value files depending on API level
- Values-Xdp: Value files depending on screen density
- XML: XML files
- AndroidManifest.xml: App metadata file
- build.gradle: Build related settings

# *Application Components*

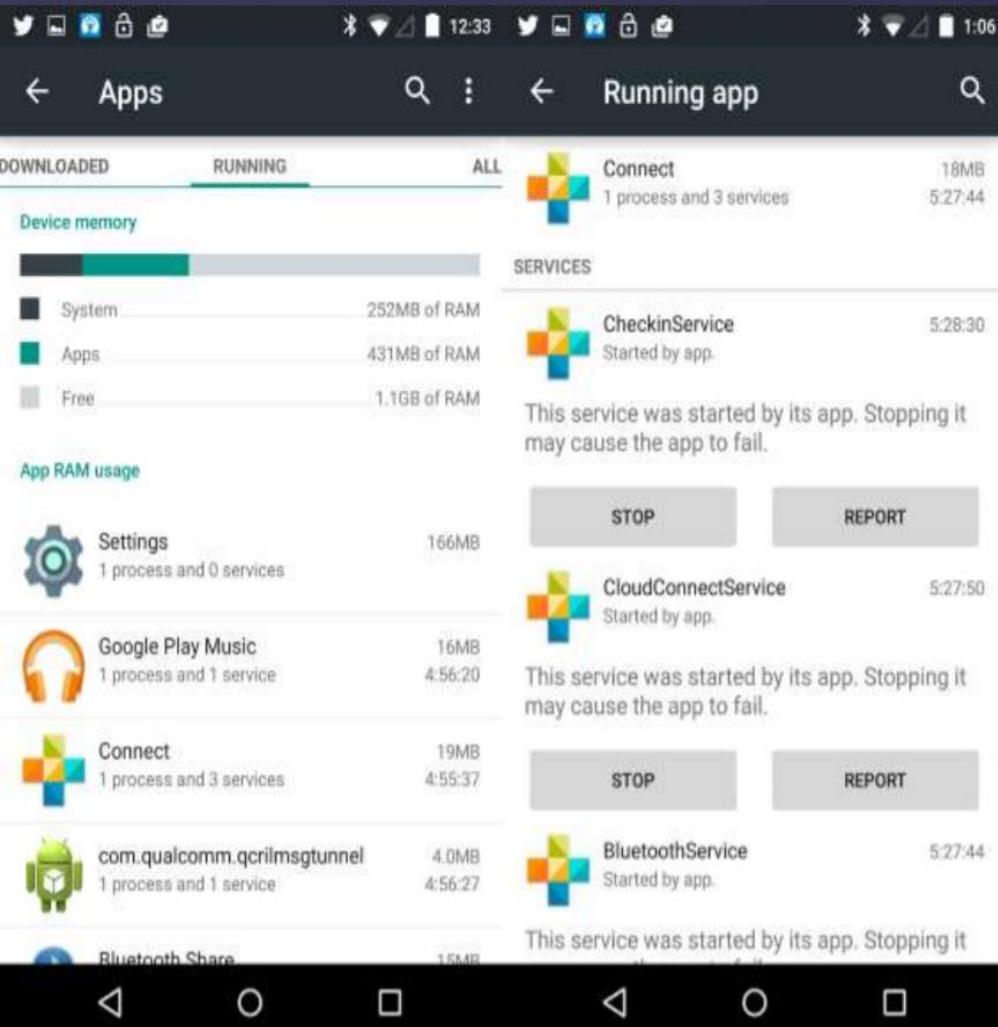
Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact. There are following four main components that can be used within an Android application :

# Activities



- They dictate the UI and handle the user interaction to the smart phone screen
- Every app has at least one activity
- Activities can be full-screen, floating or embedded inside another activity

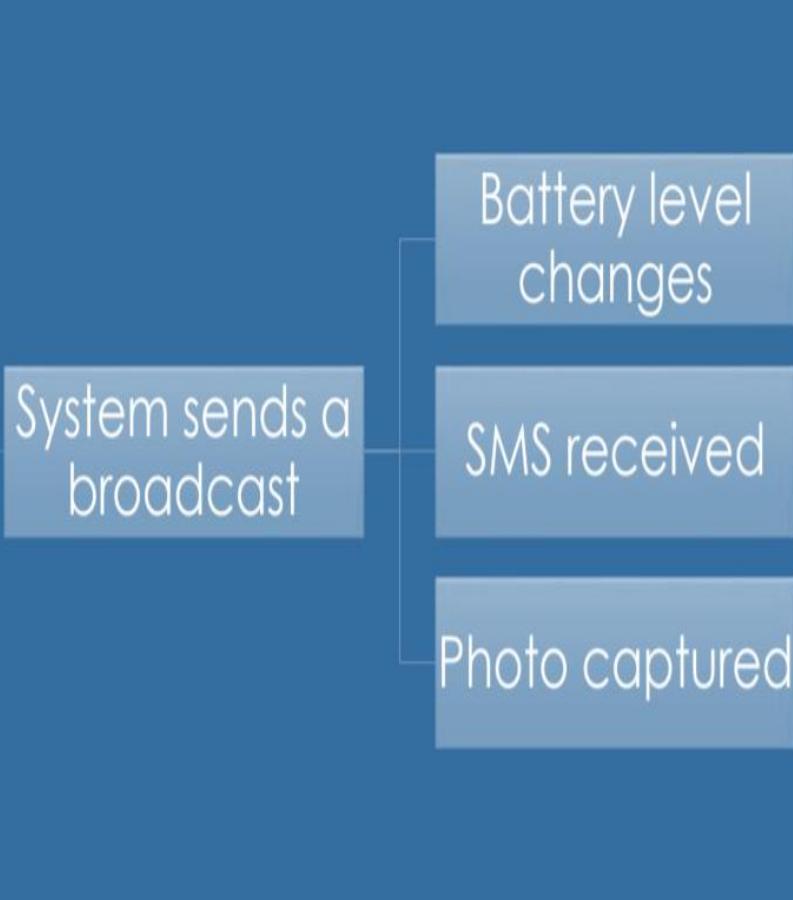
# Services



- ¶ A service is a long running operation in the background
- ¶ There are two types of services in Android – Bounded (which runs as long as components which bind to it run) and Unbounded (which runs indefinitely)
- ¶ Services run on the main thread of the application by default

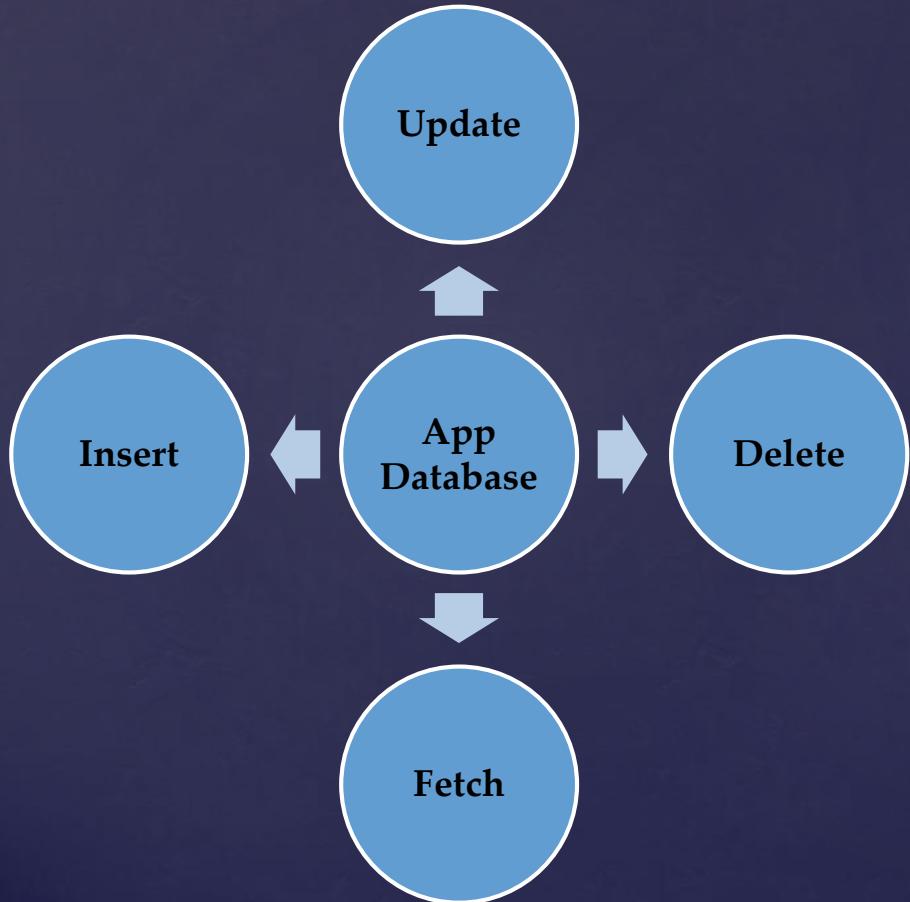
# *BroadcastReceiver*

App/Service



- ¶ A broadcast is a system or app event that can be “broadcasted” so other apps/services can listen for it
- ¶ Broadcasts are handled by a BroadcastReceiver, which is a component that allows you to listen for broadcasts
- ¶ A BroadcastReceiver can be implemented in AndroidManifest.xml, or dynamically by calling registerReceiver(), or both

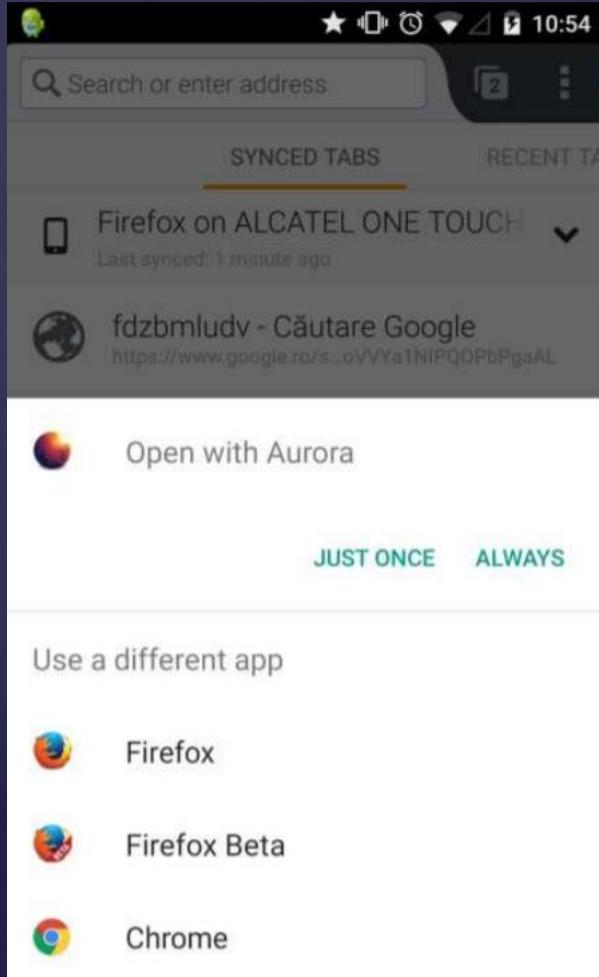
# *Content Providers/Resolvers*



- A content provider allows you to store data in your app in a structured way, similar to a relational database like SQL, for the purpose of providing it to other apps. Example usage: Contacts app, SMS app, etc
- A content resolver allows you to get data from a content provider or manipulate its data (modify, delete, update, etc)
- You cannot request to read data from a content provider at runtime, it has to be declared in `AndroidManifest.xml`

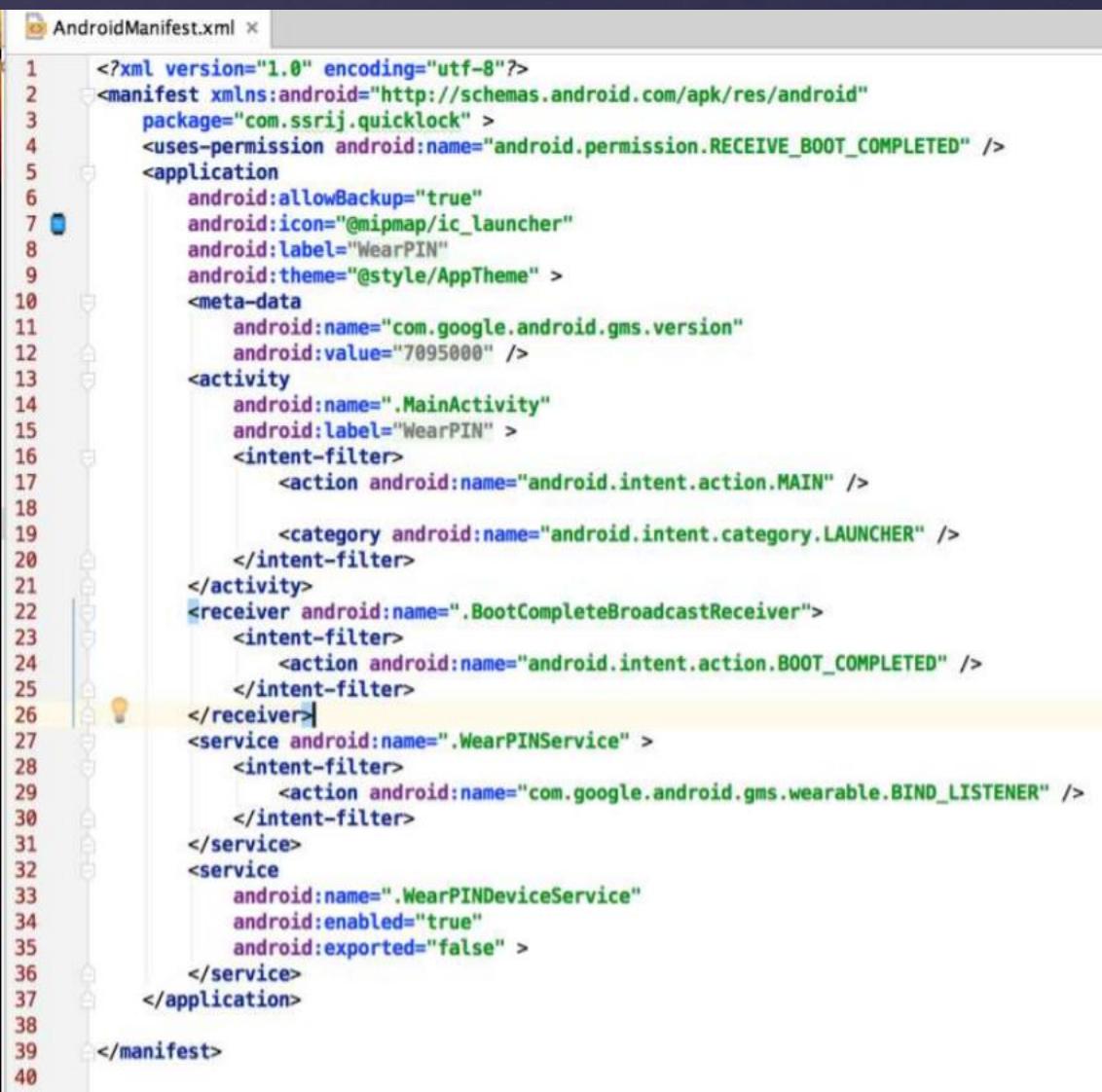
# *Additional Components*

## *Intents*



- An intent is an abstract description of an operation to be performed. Think of it as an “intention” to do something
- Intents can be used to start activities, services or send a broadcast
- Intents are of two types – Explicit (when you know what exactly you want to do), and Implicit (when you’re not sure what you want to do)

# AndroidManifest.XML

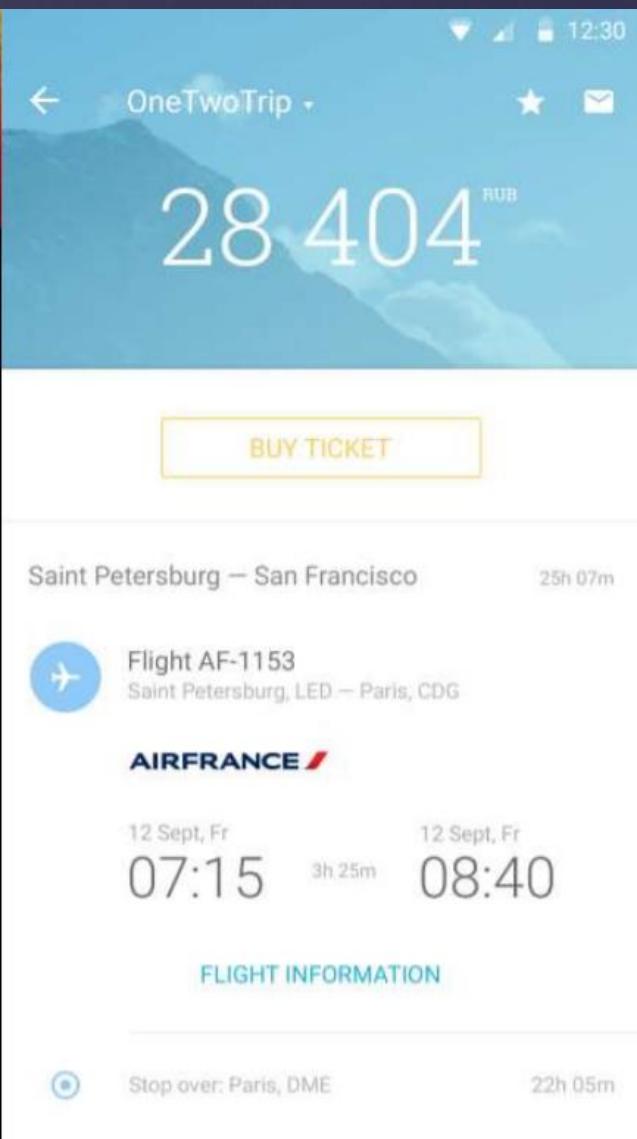


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   package="com.ssrij.quicklock" >
4     <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
5     <application
6       android:allowBackup="true"
7       android:icon="@mipmap/ic_launcher"
8       android:label="WearPIN"
9       android:theme="@style/AppTheme" >
10       <meta-data
11         android:name="com.google.android.gms.version"
12         android:value="7095000" />
13       <activity
14         android:name=".MainActivity"
15         android:label="WearPIN" >
16         <intent-filter>
17           <action android:name="android.intent.action.MAIN" />
18
19           <category android:name="android.intent.category.LAUNCHER" />
20         </intent-filter>
21       </activity>
22       <receiver android:name=".BootCompleteBroadcastReceiver">
23         <intent-filter>
24           <action android:name="android.intent.action.BOOT_COMPLETED" />
25         </intent-filter>
26       </receiver>
27       <service android:name=".WearPINService" >
28         <intent-filter>
29           <action android:name="com.google.android.gms.wearable.BIND_LISTENER" />
30         </intent-filter>
31       </service>
32       <service
33         android:name=".WearPINDeviceService"
34         android:enabled="true"
35         android:exported="false" >
36       </service>
37     </application>
38
39 </manifest>
```

Is the file which describes the fundamental characteristics of the app and defines each of its components.

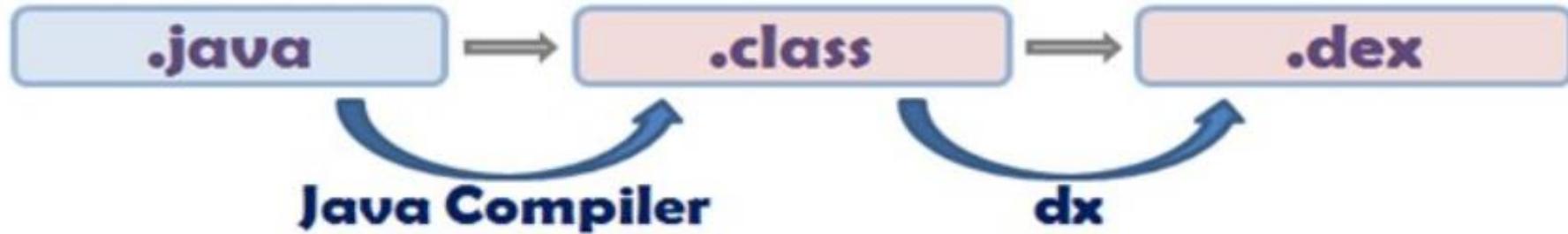
# Layouts

- A layout defines the visual structure for a user interface, such as the UI for an activity or app widget
- Layouts can be defined both in XML or programmatically using View and ViewGroup objects
- There are 5 different types of Layouts in Android: LinearLayout, RelativeLayout, FrameLayout, TableLayout and AbsoluteLayout



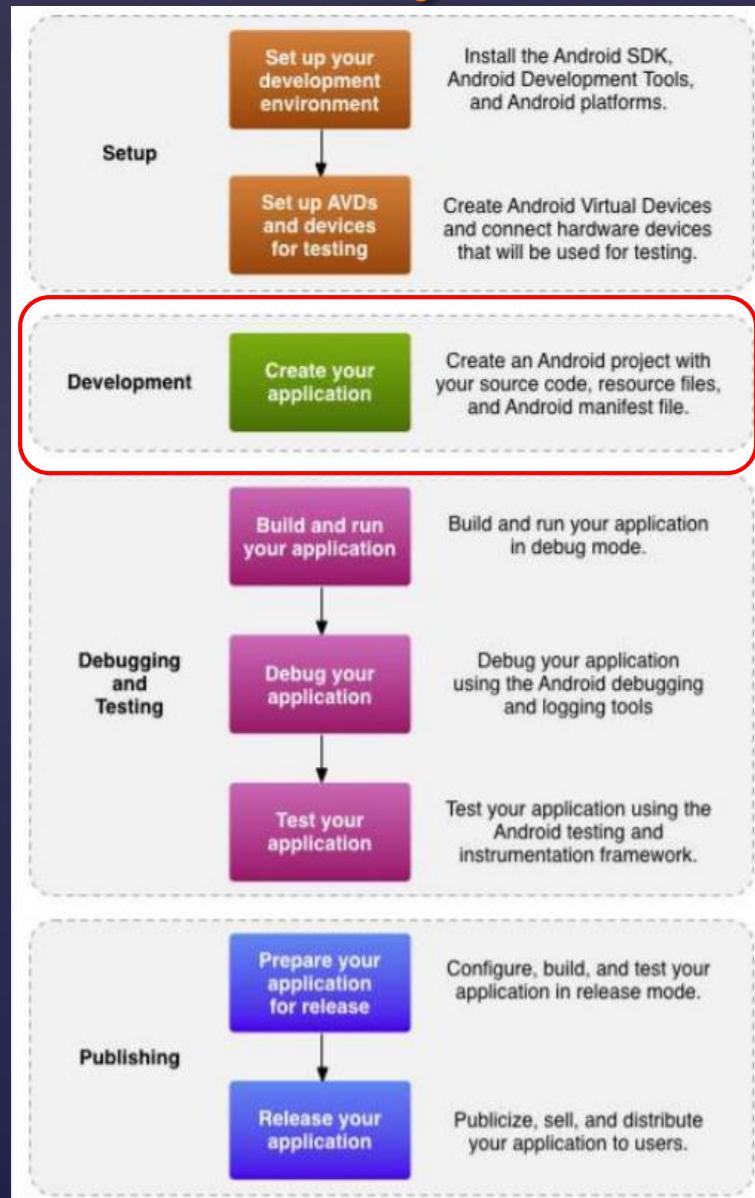
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
    <EditText  
        android:id="@+id/name"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" />  
    <Button  
        android:layout_width="96dp"  
        android:layout_height="wrap_content"  
        android:layout_below="@+id/name"  
        android:layout_alignParentRight="true"  
        android:text="@string/done" />  
</RelativeLayout>
```

# *Life cycle of Android application*



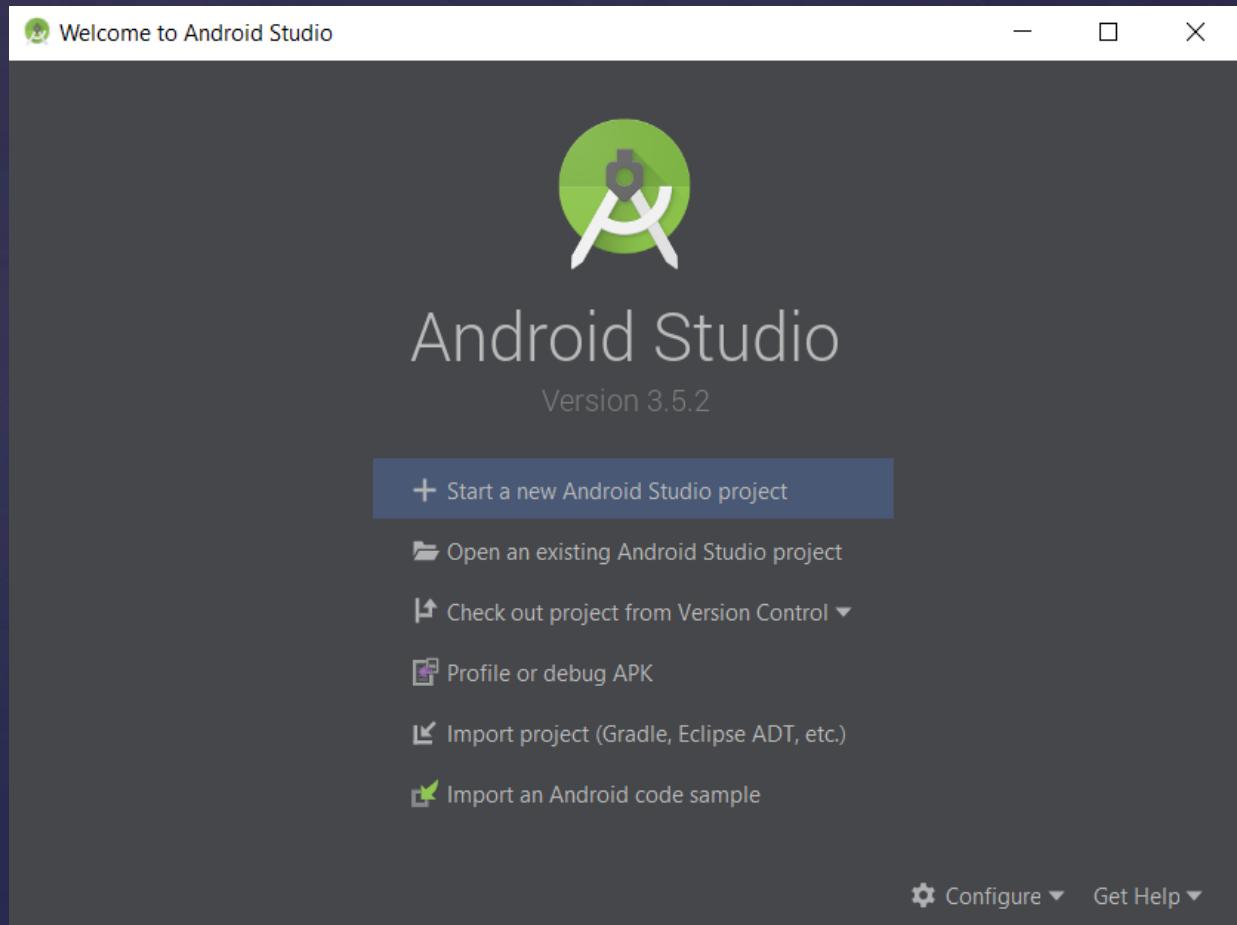
- Android application starts its life as Java Source code.
- Compiled by Javac to byte code (.class files).
- Byte code is input to Android SDK.
  - The dx tool available in the SDK converts Java bytecode to DVM bytecode at build time
  - The dx format is a highly efficient and compact bytecode
- Android SDK converts it to .dex files, which run on Dalvik Vm.

# Workflow



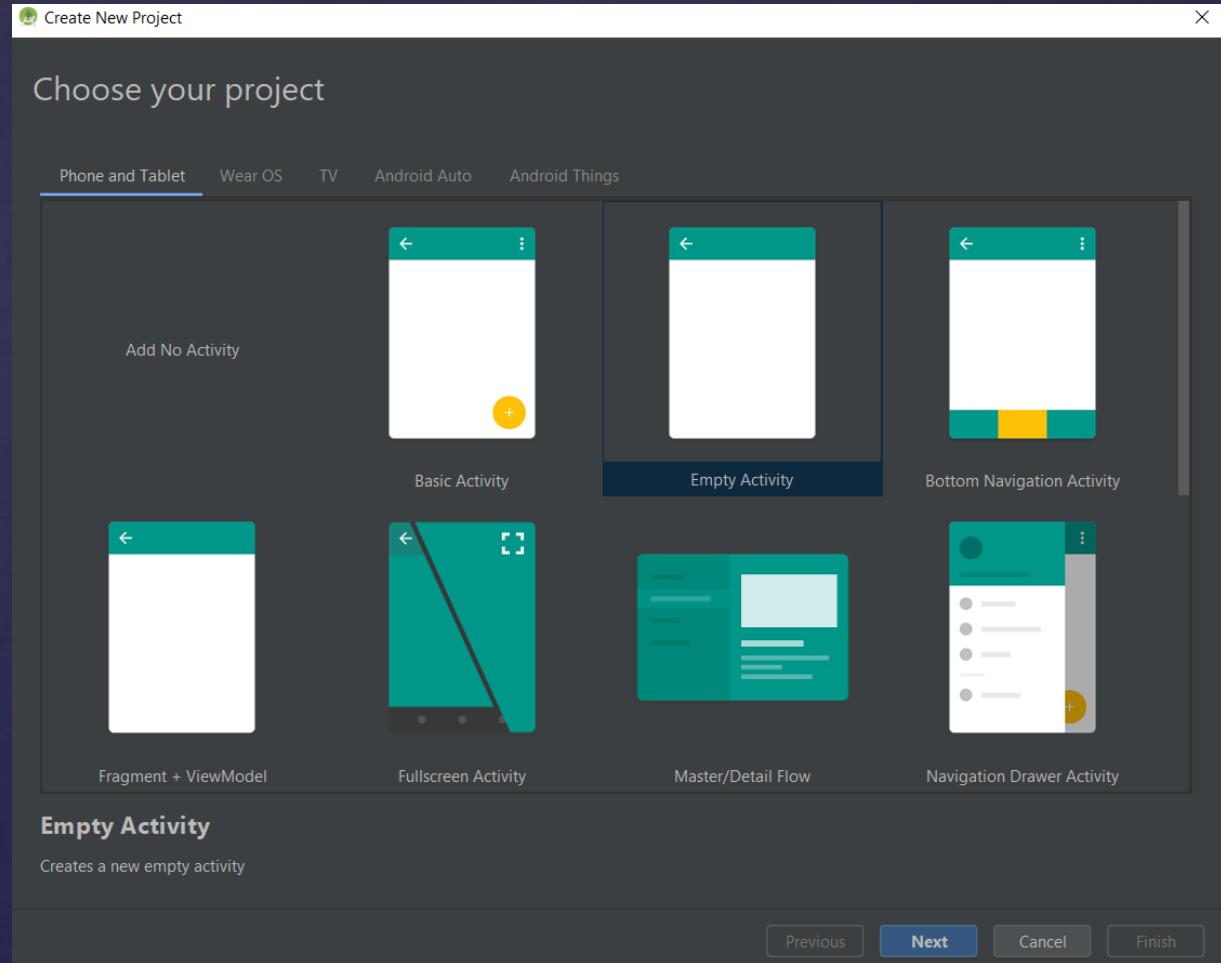
# *Running my first application*

The first step is to create a simple Android Application using Android studio. When you click on Android studio icon, it will show screen as shown here:



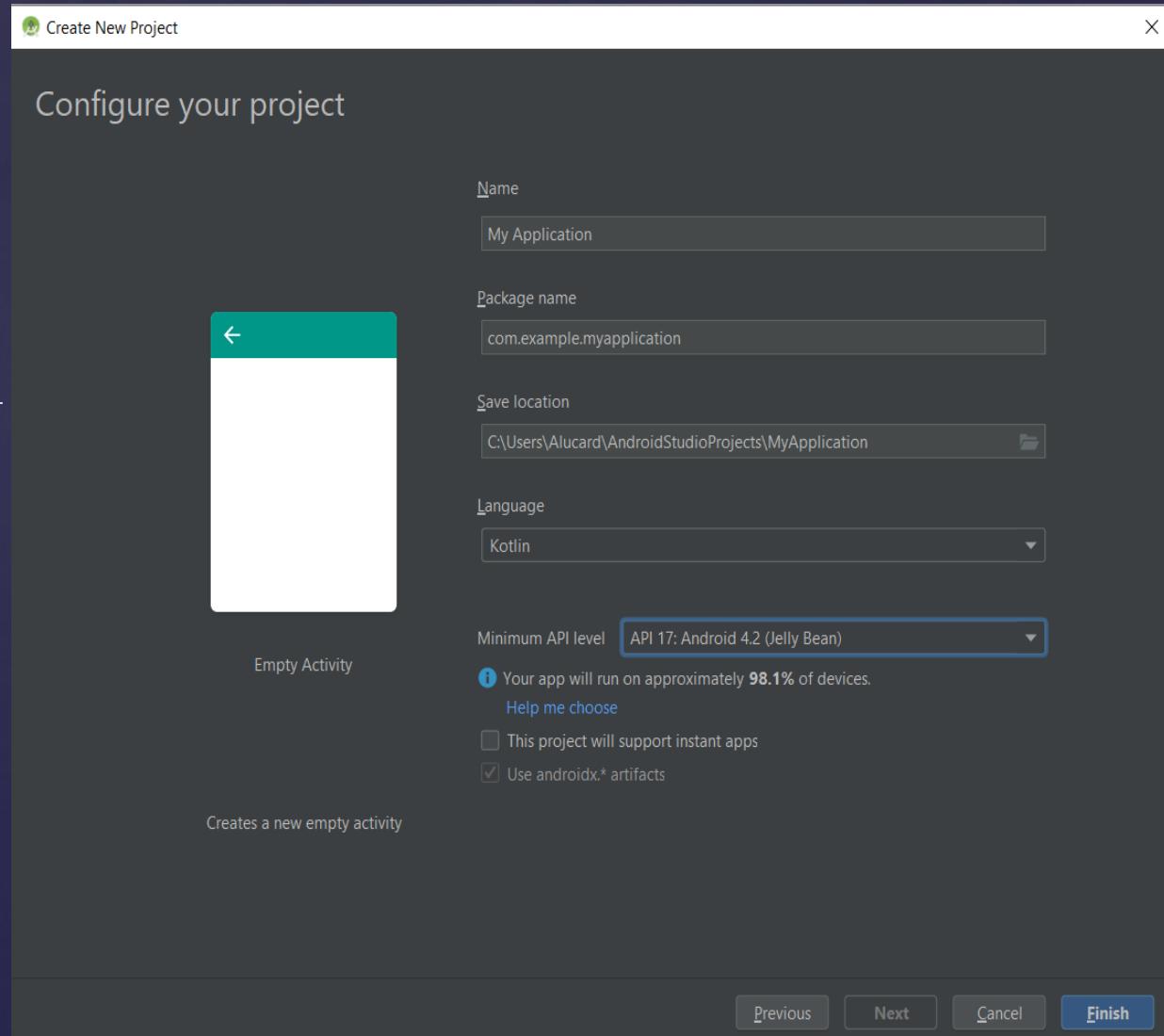
# *Running my first application*

The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.

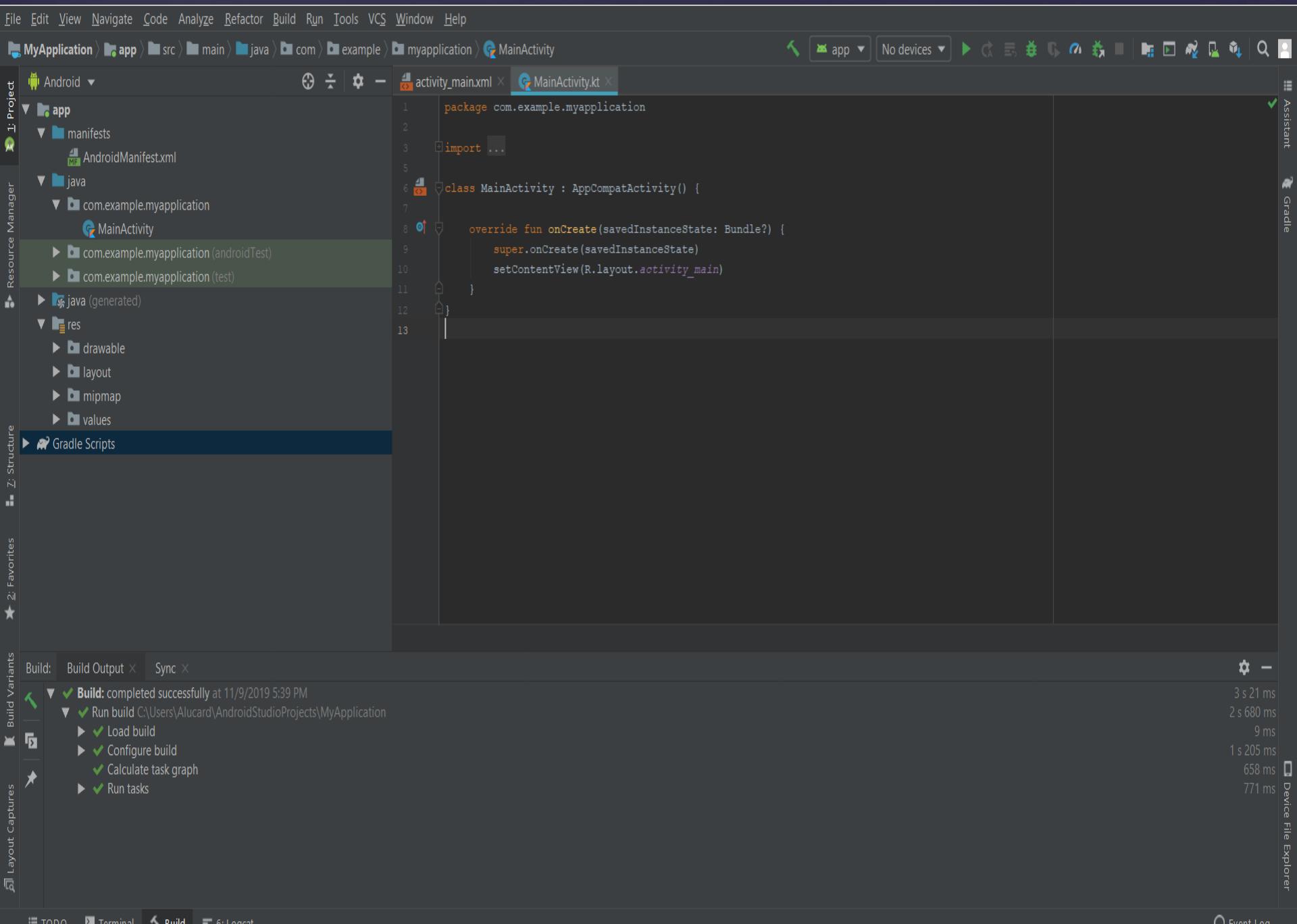


# *Running my first application*

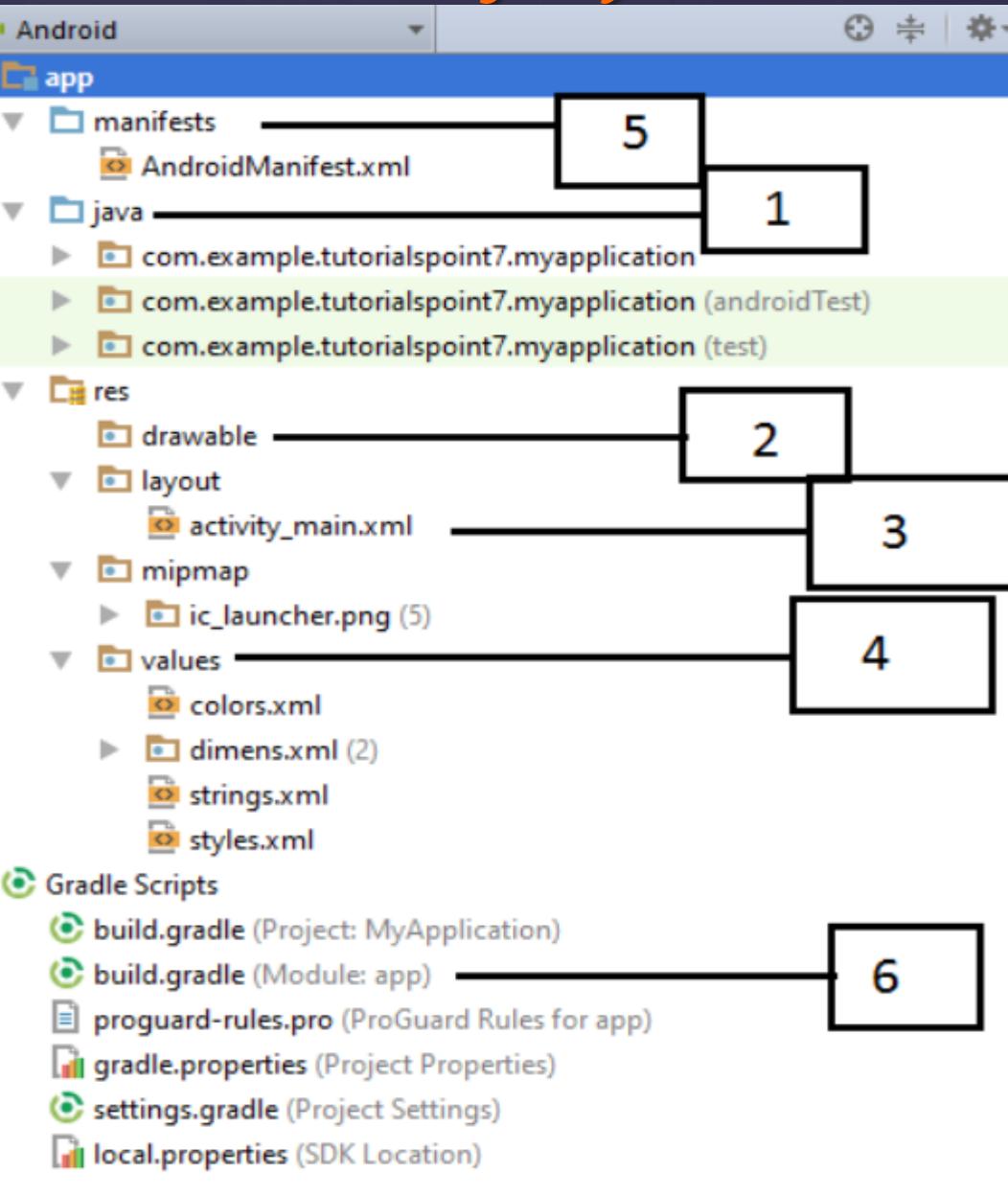
A new installation frame should ask Application name, package information and location of the project. You need to specify Minimum SDK, and declare as API:17 Android 4.2(Jelly Bean)



# At the final stage it going to be open development tool to write the application code.



# Anatomy of Android Application



1. **Java** => Contains the `.java` source files for your project.
2. **res/drawable** => A directory for drawable objects
3. **res/layout** => A directory for files that define your app's UI
4. **res/values** => A directory for other various XML files that contain a collection of strings and colours definitions...
5. **AndroidManifest.xml**
6. **Build.gradle** => contains `compileSdkVersion`, `buildToolsVersion`, `applicationId`, `minSdkVersion`, `targetSdkVersion`, `versionCode` and `versionName`

# The Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the application wizard for *Hello World!* application

```
package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Here, *R.layout.activity\_main* refers to the *activity\_main.xml* file located in the *res/layout* folder. The *onCreate()* method is one of many methods that are figured when an activity is loaded.

# The Manifest File

Following is the list of tags which you will use in your manifest file to specify different Android application components :

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        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>
```

# *The Strings File*

The **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file.

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

# The Layout File

The **activity\_main.xml** is a layout file available in *res/layout* directory, that is referenced by your application when building its interface

The screenshot shows the Android Studio interface with the layout editor open. On the left is the XML code editor displaying the `activity_main.xml` file. The code defines a ConstraintLayout containing a single TextView with the text "Hello World!". On the right is the layout preview window showing the resulting UI.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Activity Main

Hello World!

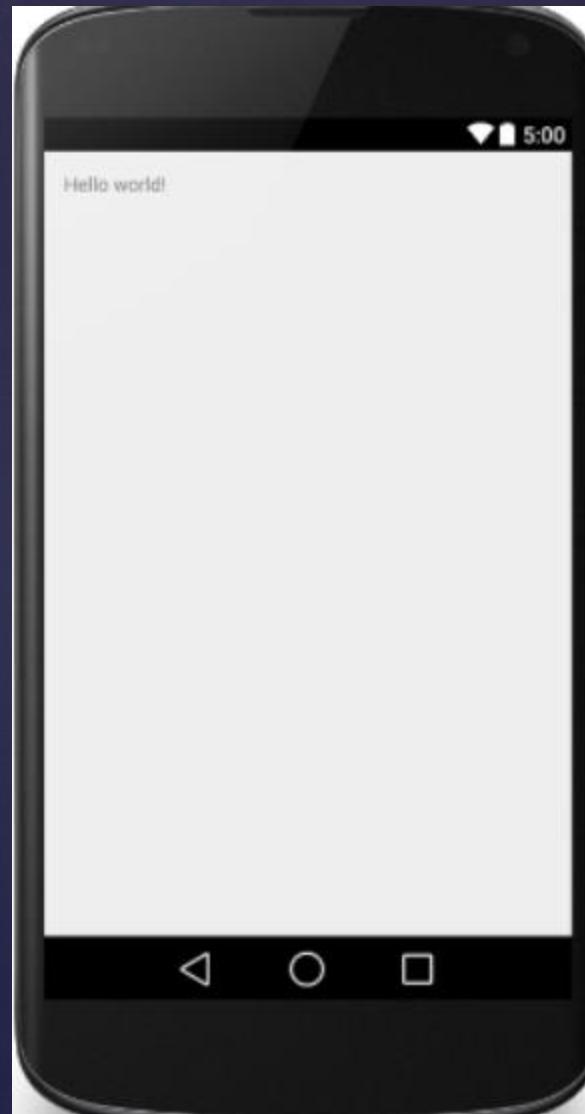
Pixel 29

Palette

33%

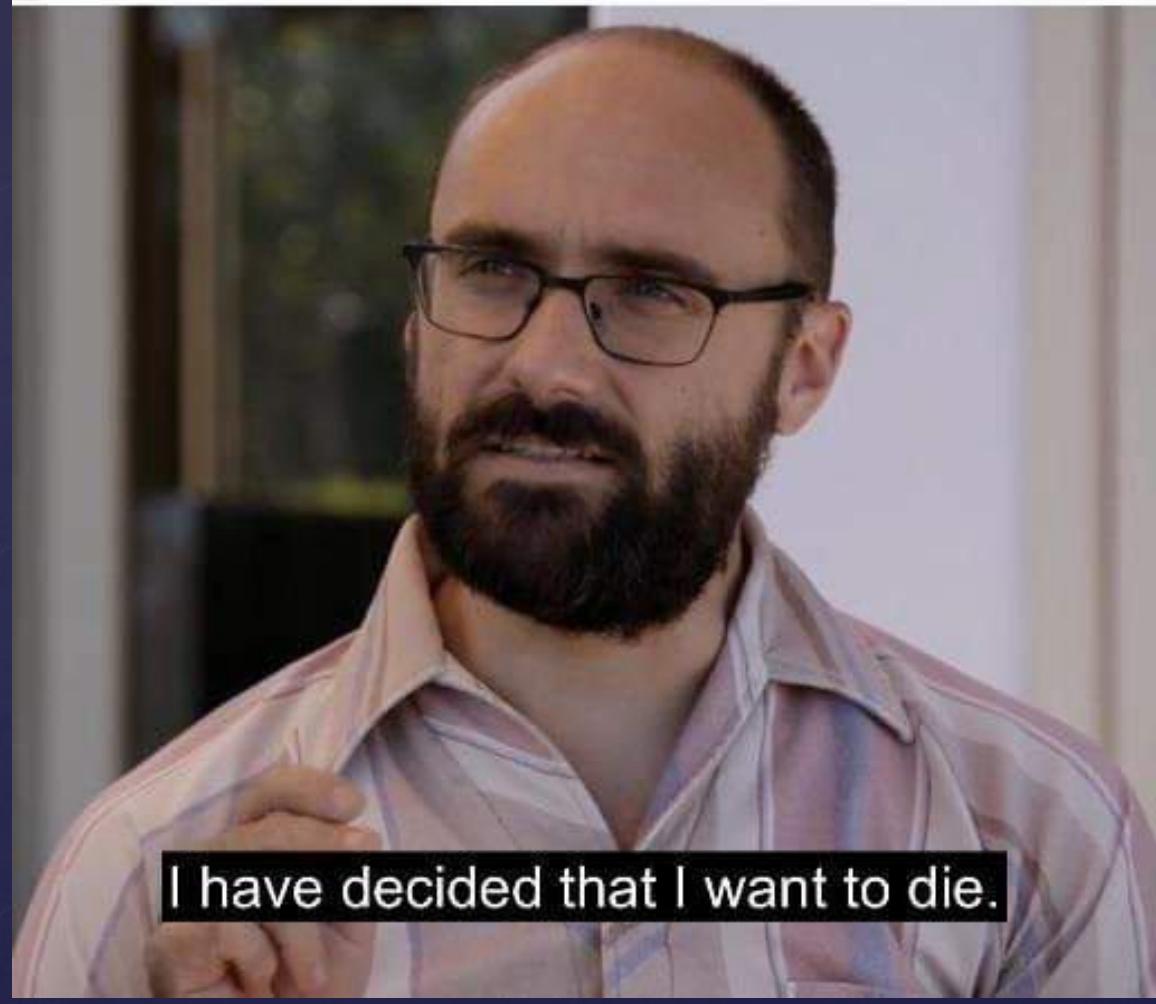
# *Running the application*

Congratulations!!! you have developed your first Android Application



Friendly  
reminder

**Me :** just wrote 200  
unsaved lines of code  
**My computer :**



I have decided that I want to die.

# Accessing Resources in Code

When your Android application is compiled, a **R** class gets generated, which contains resource IDs for all the resources available in your **res/** directory. You can use R class to access that resource using sub-directory and resource name or directly resource ID

## Example 1 :

To access *res/drawable/myimage.png* and set an ImageView you will use following code:

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
imageView.setImageResource(R.drawable.myimage);
```

## Example 2:

Consider next example where *res/values/strings.xml* has following definition

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string  name="hello">Hello, World!</string>
</resources>
```

Now you can set the text on a TextView object with ID msg using a resource ID as follows :

```
TextView msgTextView = (TextView) findViewById(R.id.msg);
msgTextView.setText(R.string.hello);
```

# Accessing Resources in Code

Example 3 :

Consider the following resource XML *res/values/strings.xml* file that includes a color resource and a string resource:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="opaque_red">#f00</color>
    <string name="hello">Hello!</string>
</resources>
```

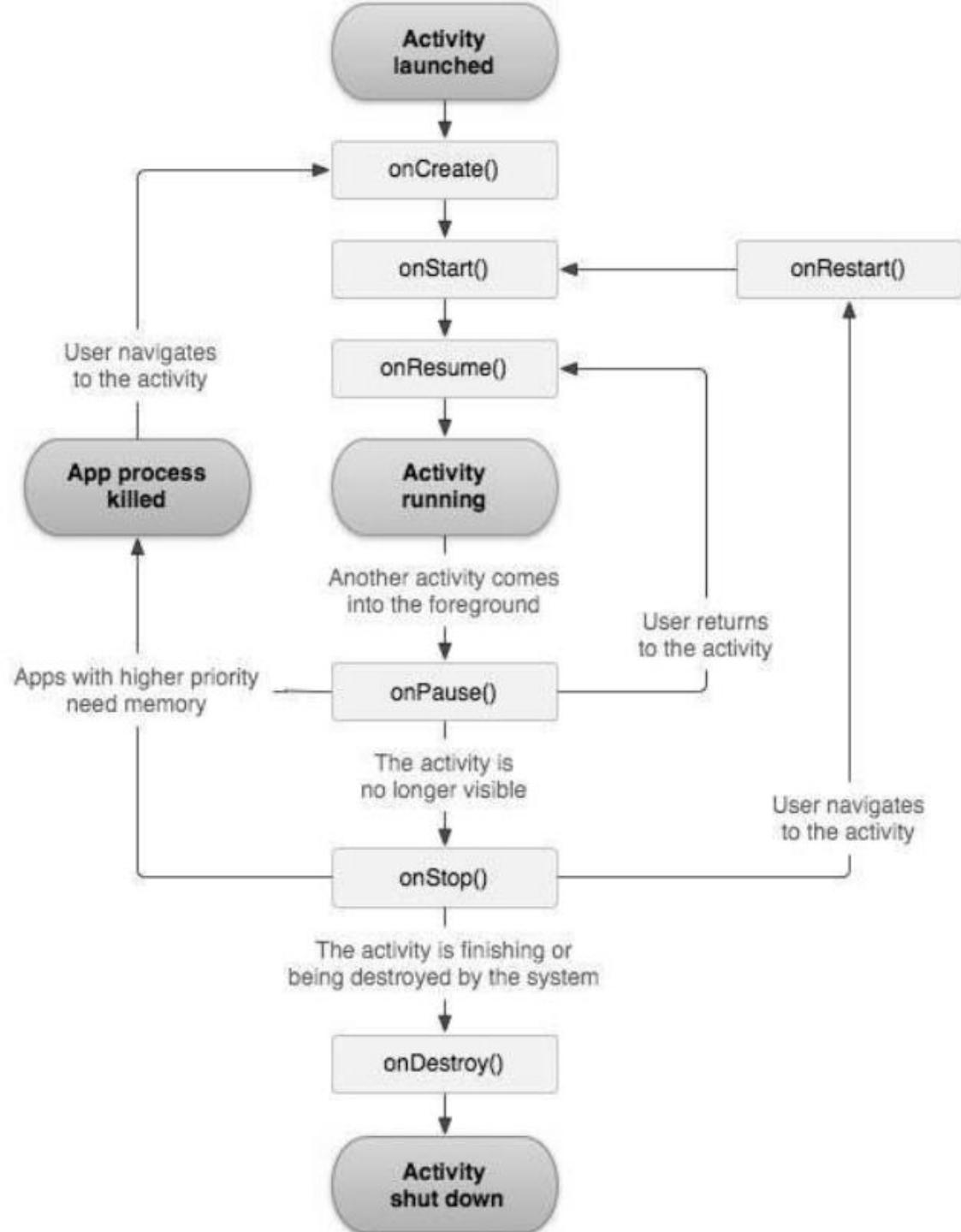
Now you can use these resources in the following layout file to set the text color and text string as follows :

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textColor="@color/opaque_red"
    android:text="@string/hello" />
```

# *The Application life cycle*



## Click here



# Layouts

Linear Layout



Relative Layout



Web View



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.

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).

Displays web pages.

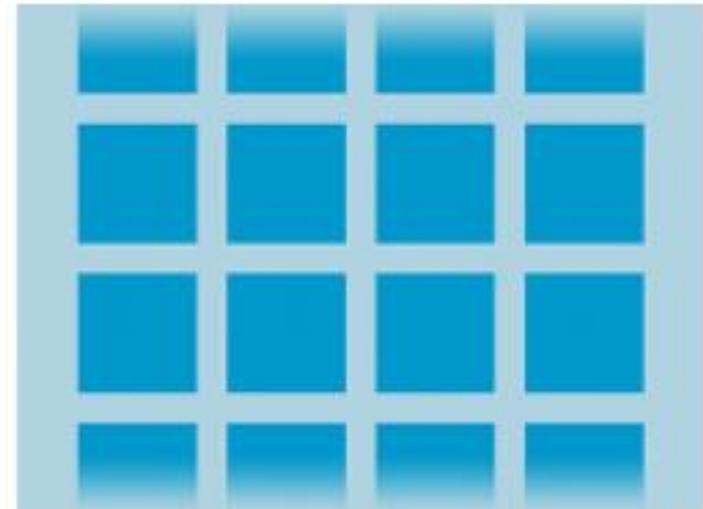
# Layouts

- When the content for your layout is dynamic or not pre-determined, you can use a layout that subclasses `AdapterView` to populate the layout with views at runtime.

List View



Grid View

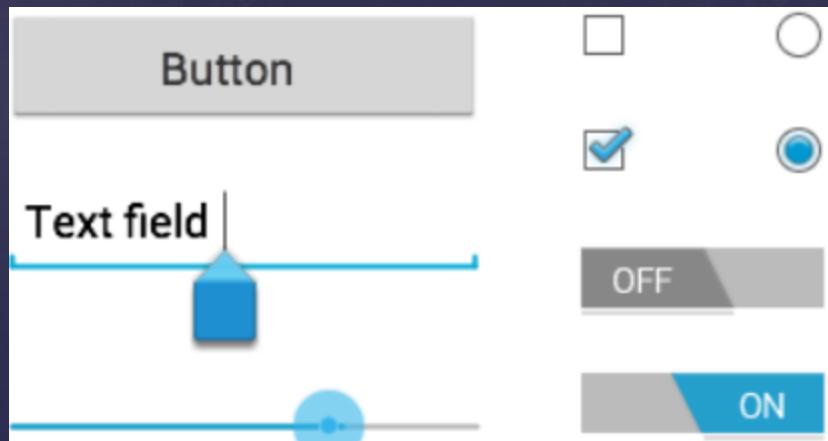


Displays a scrolling single column list.

Displays a scrolling grid of columns and rows.

# UI Controls

& Input controls are the interactive components in your app's user interface. Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, check box, zoom buttons, toggle buttons, and many more.



test.txt

# UI Controls

- Input controls are the interactive components in your app's user interface. Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, check box, zoom buttons, toggle buttons, and many more.

Example 1: Button with id = myButton

& Step 1:

Public class MainActivity extends AppCompatActivity implements View.OnClickListener

& Step 2:

Private Button button;

(inside on create)=>

```
button = (Button) findViewById(R.id.myButton);
```

```
button.setOnClickListener(this);
```

# UI Controls

Example 1: Button with id = myButton

¤ Step 3:

```
@override  
Public void onClick(View view) {  
if(view == myButton){  
//traitement  
}  
}
```

# UI Controls

Example 2: Button style

<https://angrytools.com/android/button/>

- ¤ Step 1: paste the xml generated code in layout.xml
- ¤ Step 2: paste the buttonShape.xml generated code in the drawable folder

Example 3 : changing case of textView

TP ; help (setText , toLower , toUpper)

# Intents

An Android **Intent** is an abstract description of an operation to be performed.

Example 1 :

```
// Explicit Intent by specifying its class name
```

```
Intent i = new Intent(FirstActivity.this, SecondActivity.class);
```

```
// Starts TargetActivity
```

```
startActivity(i);
```

These intents (explicit) designate the target component by its name and they are typically used for application-internal messages - such as an activity starting a subordinate service or launching a sister activity.



# Intents

## Example 2 : Sending emails

```
Intent email = new Intent(Intent.ACTION_SEND, Uri.parse("mailto:"));
email.putExtra(Intent.EXTRA_EMAIL, recipients);
email.putExtra(Intent.EXTRA_SUBJECT, subject.getText().toString());
email.putExtra(Intent.EXTRA_TEXT, body.getText().toString());
startActivity(Intent.createChooser(email, "Choose an email client from..."));
```

## Example 3 : Web Search

```
String q = "Memes";
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH );
intent.putExtra(SearchManager.QUERY, q);
startActivity(intent);
```

# Tests

Example 1 : Robolectric  
<http://robolectric.org/>



Example 2 : Espresso  
<https://developer.android.com/training/testing/espresso>

# Do You Have Questions?



Is coding An  
instrument?