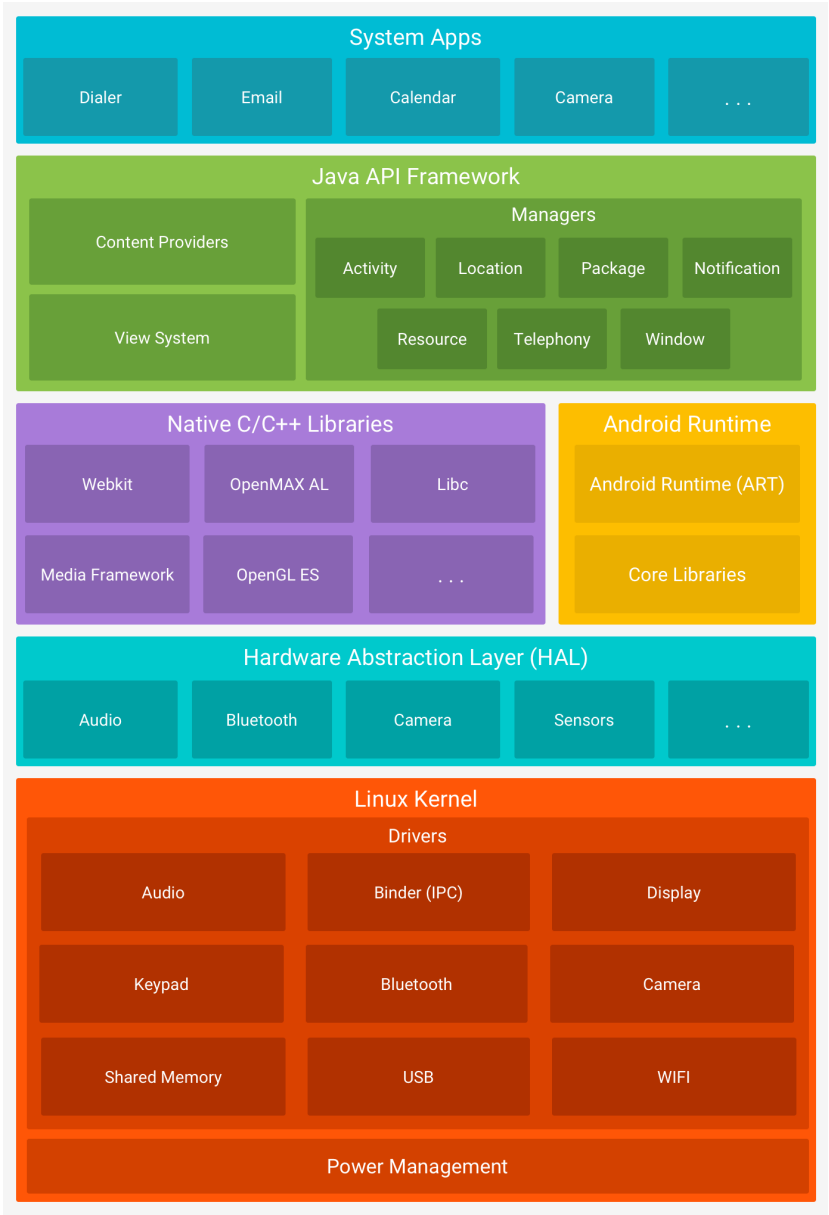
Android

System, platform and application types

Bibliography

1. Wallace Jackson, *Android Apps for Absolute Beginners*, Apress, 2017
2. Peter Späth, *Learn Kotlin for Android Development,* Apress 2019
3. Android Application Fundamentals, http://developer.android.com/guide/topics/fundamentals.html

Android Schematics



Android Components

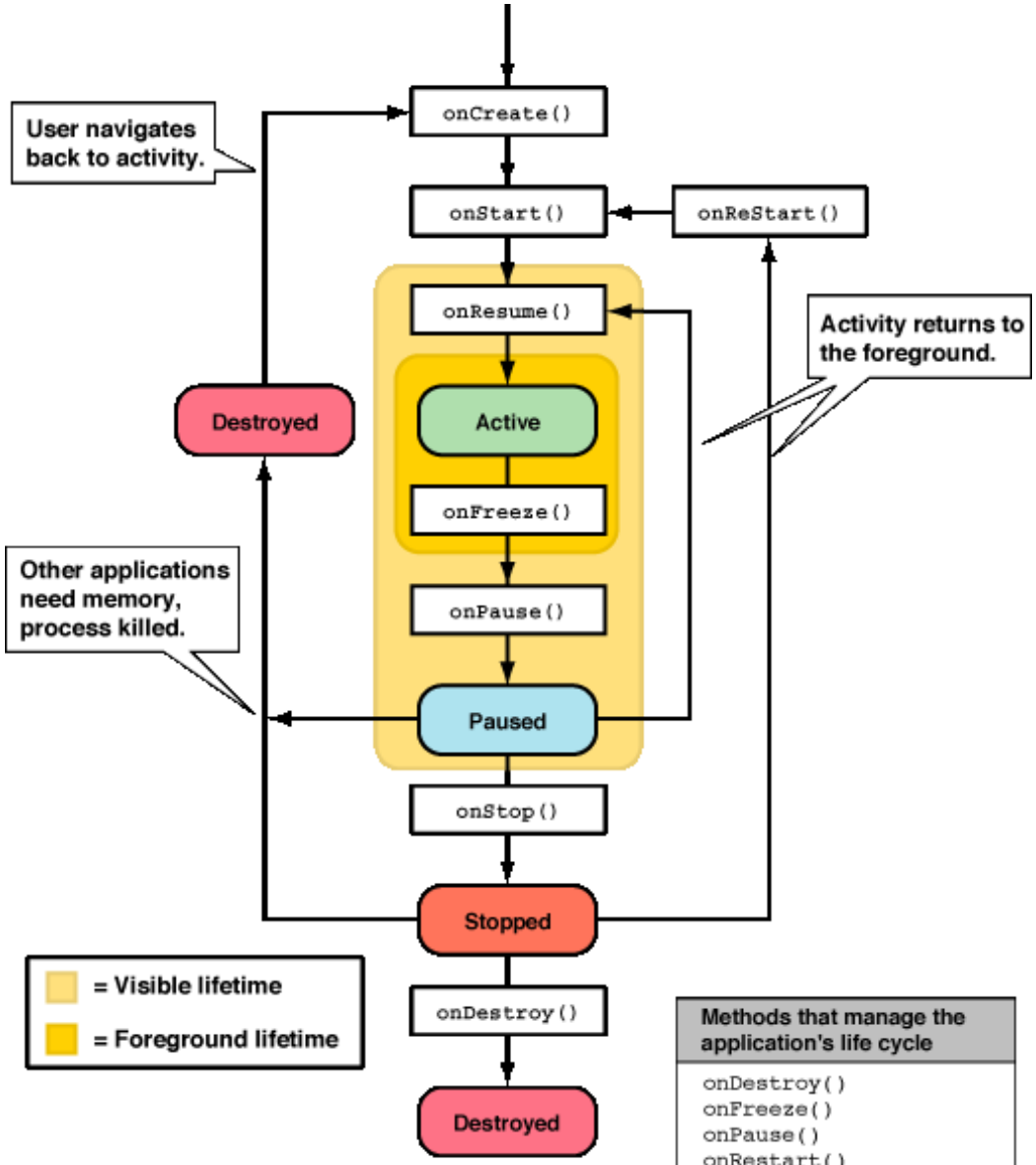
- Activities
- Services
- Content Providers
- Broadcast Receivers

Activity

- **Window**
- Implementation: extends class Activity
- *Lives longer than the process*
  - Serializing
  - **Partially controlled by the programmer**
- Not for a lot of processing

To read: developer.android.com

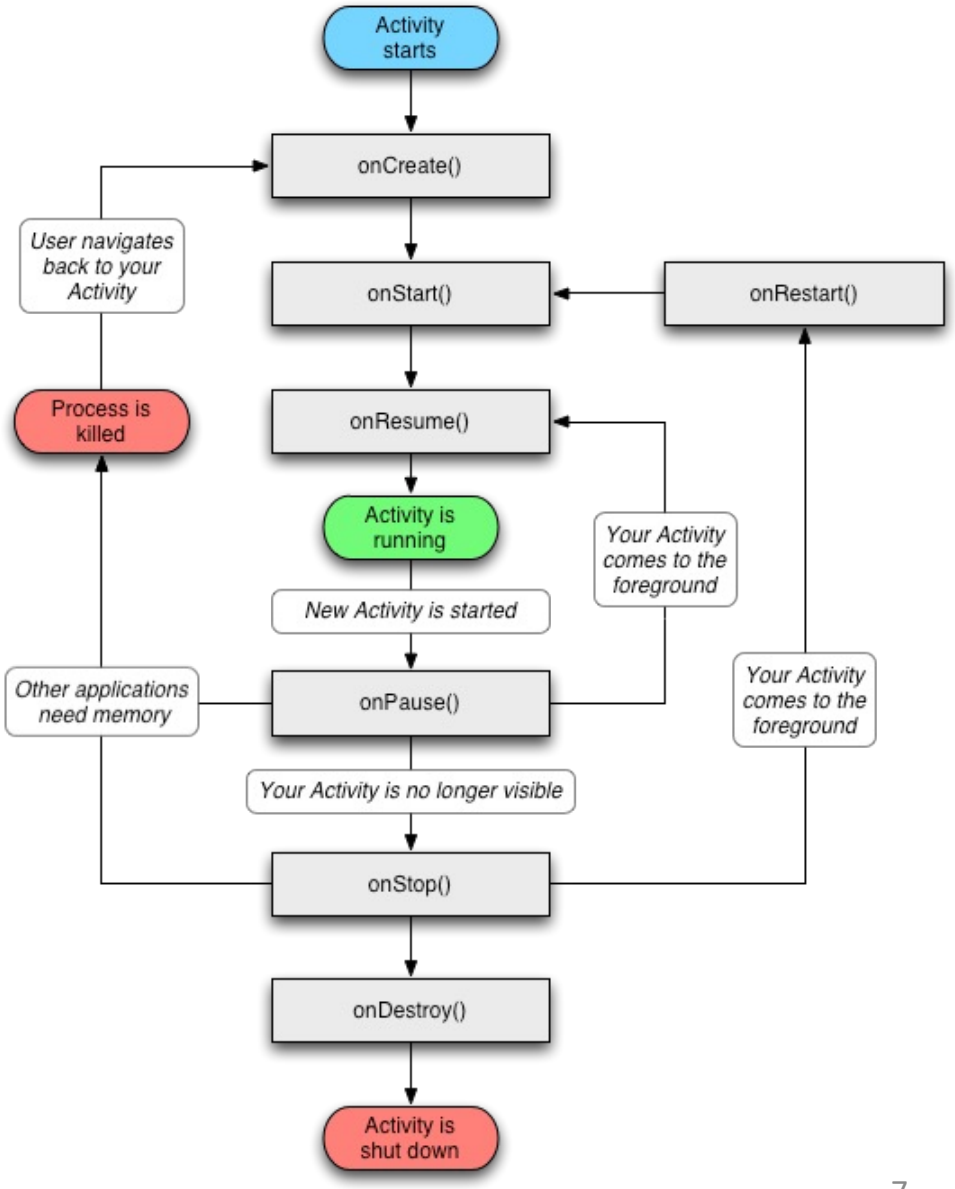
Activity lifecycle



To read: developer.android.com

Important Functions

- Class **Activity**
  - **void** *onCreate* (…);
  - **void** *onStart* (…);
  - **void** *onRestart* (…);
  - **void** *onResume* (…);
  - **void** *onPause* (…);
  - **void** *onStop* (…);
  - **void** *onDestroy* (…);
- **Must call parent functions**
  - **super**.*onCreate* (…);
  - **…**



Storing and loading of the status

- Store
  - **void** *onSaveInstanceState* (**Bundle** state)
- Load
  - **void** *onLoadInstanceState* (**Bundle** state)
  - **void** *onCreate* (**Bundle** savedInstance)

```
lateinit var textView: TextView

// Some transient state for the activity instance.
var gameState: String? = null

override fun onCreate(savedInstanceState: Bundle?) {
    // Call the superclass onCreate to complete the creation of
    // the activity, like the view hierarchy.
    super.onCreate(savedInstanceState)

    // Recover the instance state.
    gameState = savedInstanceState?.getString(GAME_STATE_KEY)

    // Set the user interface layout for this activity.
    // The layout is defined in the project res/layout/main_activity.xml file.
    setContentView(R.layout.main_activity)

    // Initialize member TextView so it is available later.
    textView = findViewById(R.id.text_view)
}

// This callback is called only when there is a saved instance previously saved using
// onSaveInstanceState(). Some state is restored in onCreate(). Other state can optionally
// be restored here, possibly usable after onStart() has completed.
// The savedInstanceState Bundle is same as the one used in onCreate().
override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
    textView.text = savedInstanceState?.getString(TEXT_VIEW_KEY)
}

// Invoked when the activity might be temporarily destroyed; save the instance state here.
override fun onSaveInstanceState(outState: Bundle?) {
    outState?.run {
        putString(GAME_STATE_KEY, gameState)
        putString(TEXT_VIEW_KEY, textView.text.toString())
    }
    // Call superclass to save any view hierarchy.
    super.onSaveInstanceState(outState)
}
```

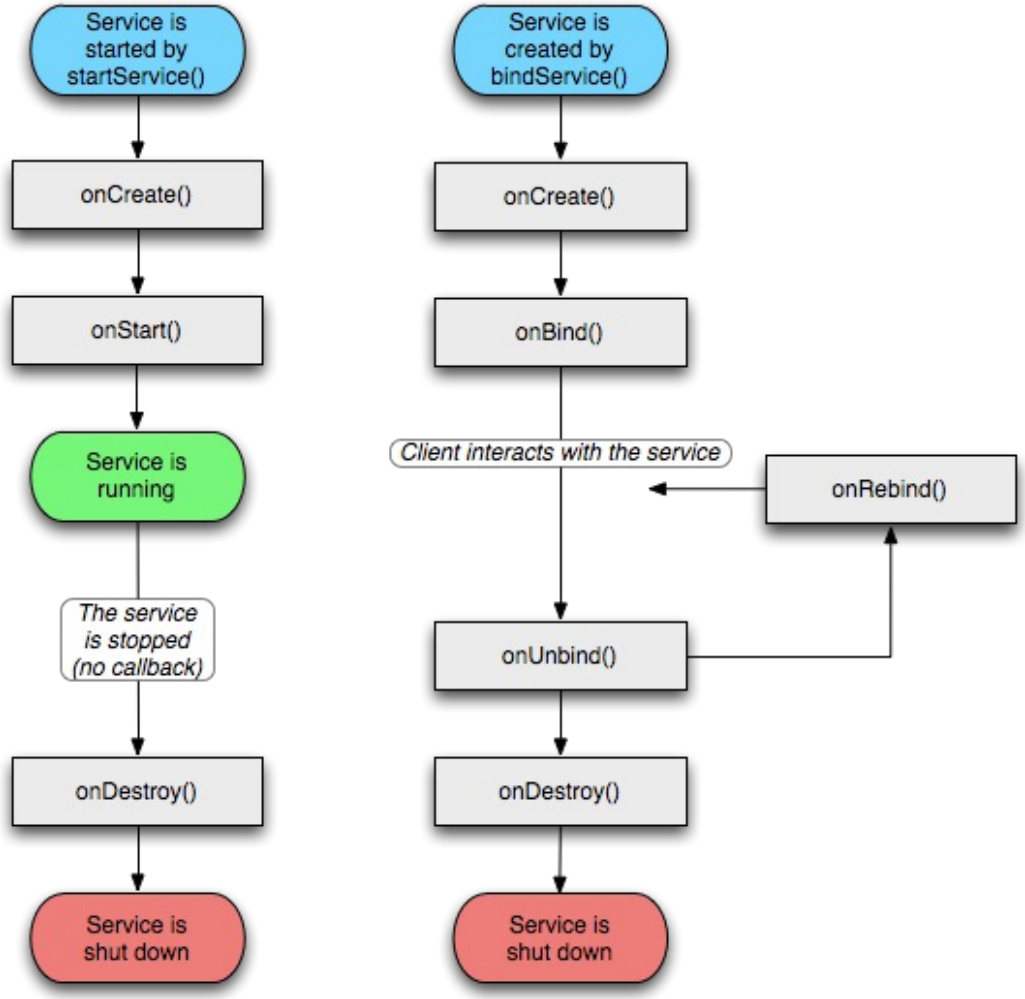
Services

- Specially for processing
- Runs in *background*
- process
  - Low priority
  - More stable (in time)

To read: developer.android.com

Service implementation

- Extends class **Service**
  - Simple Service
    - **void** *onCreate* ();
    - **void** *onStart* (**Intent** intent, **int** startID);
    - **void** *onDestroy* ();
  - Using AIDL
    - **void** *onBind* (**Intent** intent);
    - **void** *onUnbind*();



Content Providers / Broadcast Receivers

- Content providers (link)
  - Offer information
  - Link with SQLite
  - Based on URLs
- Broadcast receivers (link)
  - Observers
  - Public events
    - SCREEN_ON
    - SCREEN_OFF
    - BATTER_STATUS_CHANGED

Context

- Context (link)
  - any application component
    - Activities
    - Services
    - *Content Providers*
  - Generated by *Dalvik/ART*
    - at startup
    - provided as a parameter
      - *Broadcast Receivers*

App Manifest

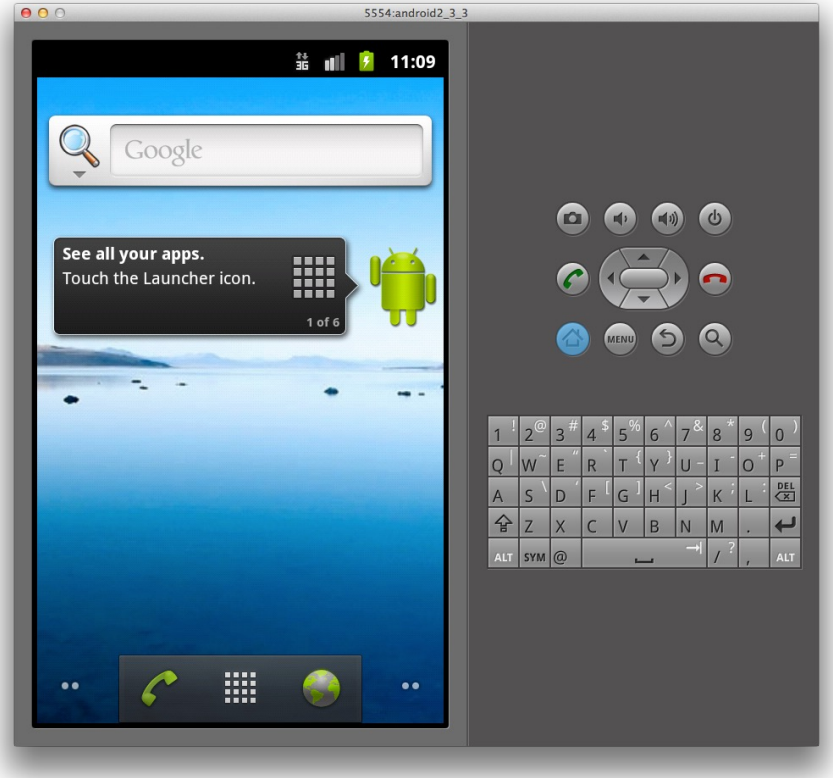
- Is mandatory
- Contains:
  - Apps components
    - Components properties
  - Permissions
  - Hardware/software requirements

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

<!--    PERMISSION USAGE DECLARATION-->
    <uses-permission android:name="android.permission.TURN_SCREEN_ON" />

<!--    SOFTWARE REQUIREMENTS DECLARATION-->
    <uses-sdk
        android:minSdkVersion="31"
        android:maxSdkVersion="33"/>

    <application
        android:allowBackup="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">

<!--        ACTIVITY DELCARATION-->
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.MyApplication">
            <intent-filter...>
        </activity>

<!--        SERVICE DECLARATION-->
        <service android:name=".MyService" />

<!--        BROADCAST RECEIVER DECLARATION-->
        <receiver android:name=".MyReceiver"/>

    </application>
</manifest>
```

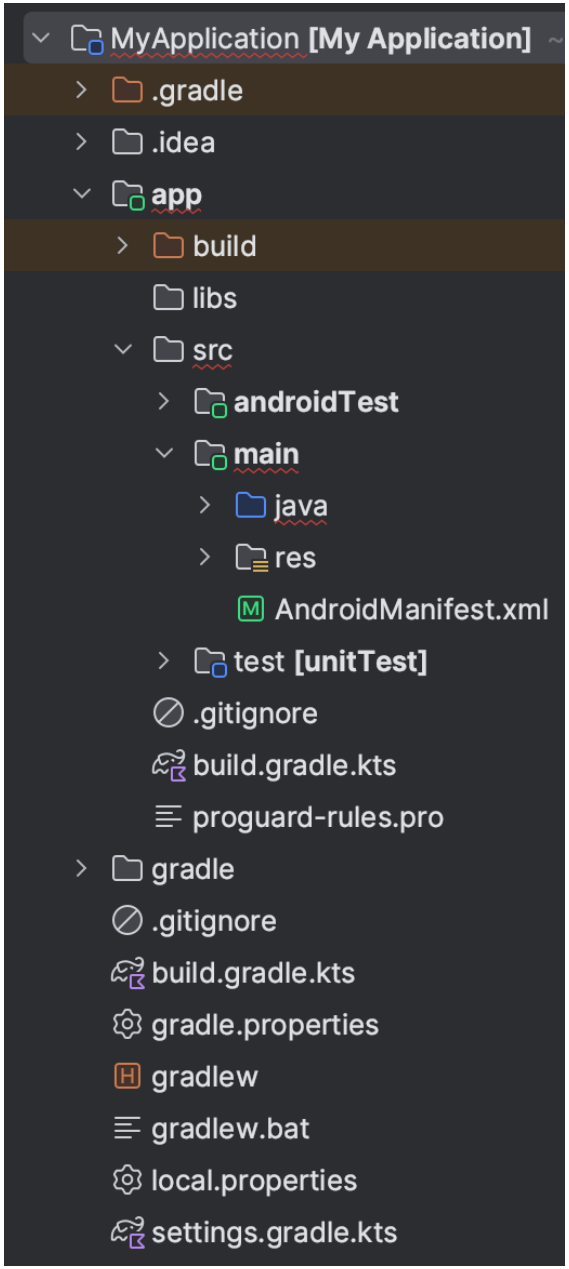
Emulator vs. Real Phone

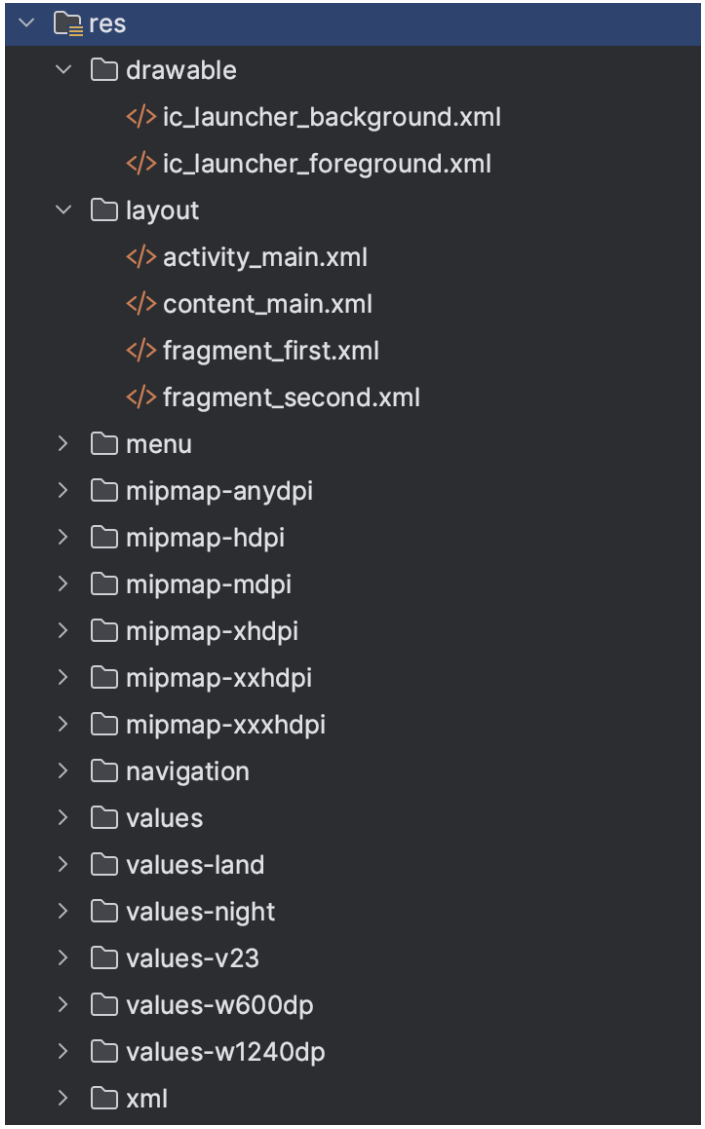
PDM
sayHello();

- Emulator
  - Real
    - Boots Linux
  - Several versions
  - **Runs separately**
- Real Phone
  - **USB Debugging**
    - Applications/Development
  - If you have an extra phone, use that one



Application template





Conclusions

- Android Applications are a set of components
  - Activities
    - Activities lifecycle
  - Services
  - Content Providers
  - Broadcast Receivers
- Context
- Run apps on
  - Emulator
  - Phone

Keywords

- Emulator
- Lifecycle
- Manifest
- Process
- Callback

- Application State
  - Running
  - Paused
  - Stopped
  - Destroyed
  - Event
    - method call

Questions

PDM
sayHello();