Introduction to Android

Ambient intelligence

Teodoro Montanaro

Politecnico di Torino, 2016/2017

Disclaimer

- This is only a fast introduction:
  - It is not complete (only scrapes the surface)
  - Only superficial notions are provided

It is a **guide** to **self-learning** and **self-documentation**

Summary

- Short history
- Platform
- Architecture
- Android app development:
  - Design an Android app
    - MVC
  - Tools
  - Application fundamentals
  - Application Main Components part 1: Activity
  - 1st Hands-on: simple calculator
    - App structure
  - Application Main Components part 2
    - Service
    - Content Provider
    - Broadcasts
  - App life cycle
  - Intent
  - 2nd Hands-on: Todo list

History

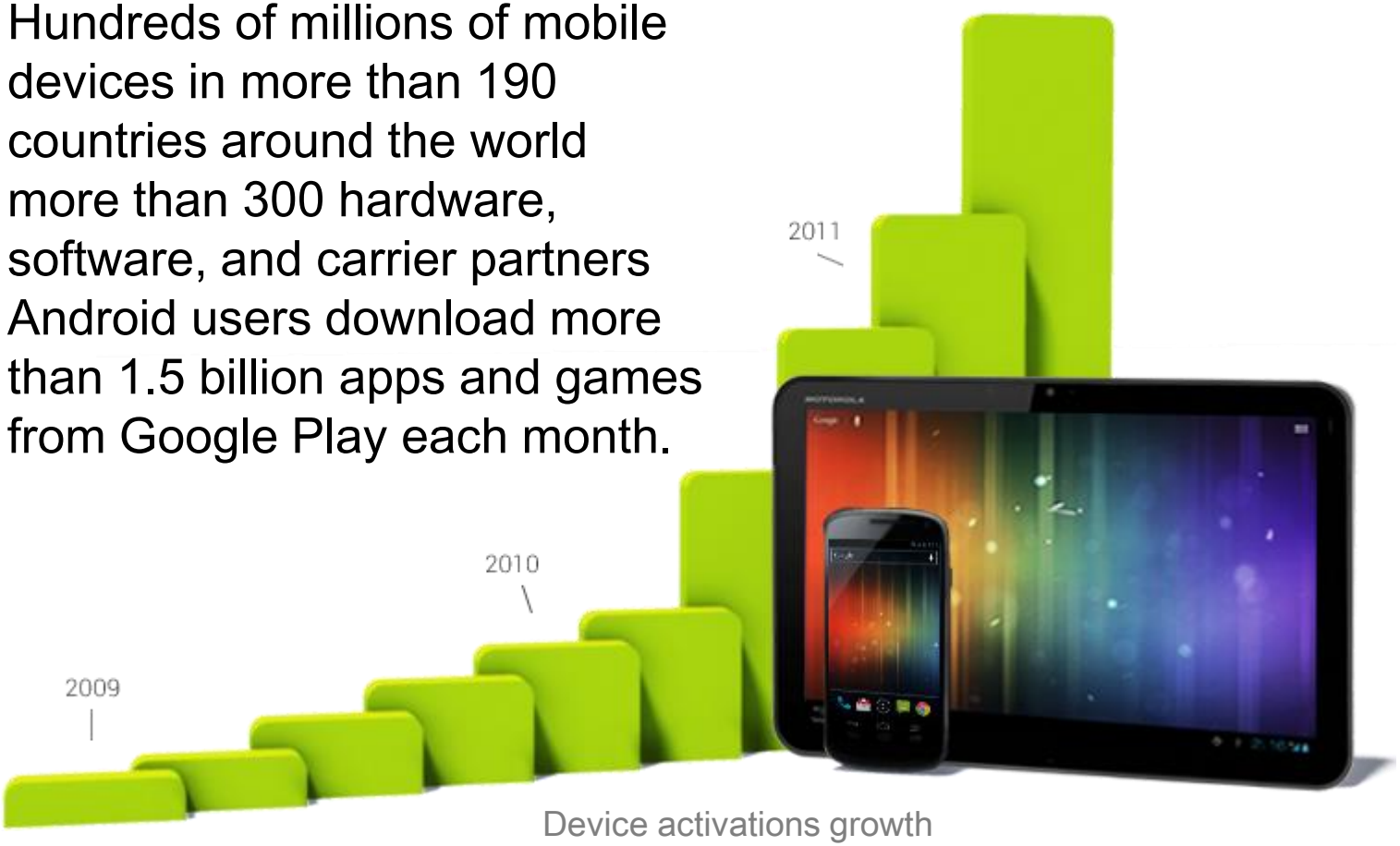
- Originally created by Andy Rubin
- Acquired by Google Inc. in 2005
- Now it is maintained by the Open Handset Alliance (OHA) (since 2007)
- Several stable releases since then

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.2%
4.2.x		17	4.6%
4.3		18	1.3%
4.4	KitKat	19	18.8%
5.0	Lollipop	21	8.7%
5.1		22	23.3%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	6.6%
7.1		25	0.5%

Last update:
May 2, 2017
(https://developer.android.com/about/dashboards/index.html)

Figures

- Hundreds of millions of mobile devices in more than 190 countries around the world
- more than 300 hardware, software, and carrier partners
- Android users download more than 1.5 billion apps and games from Google Play each month.



Device activations growth

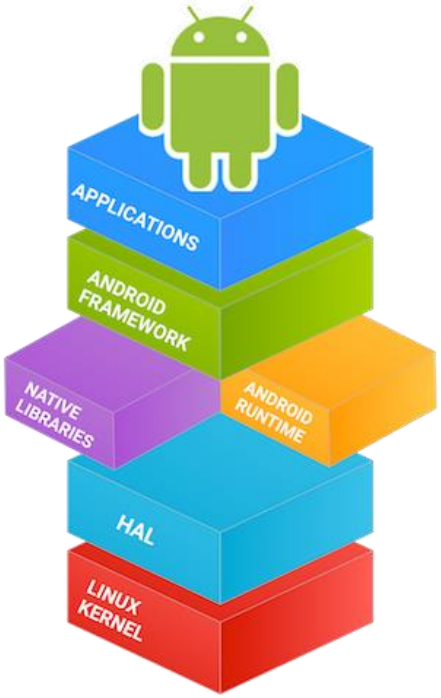
Android Platform

- Android is “an open source software stack for a wide range of mobile devices and a corresponding open source project led by Google“.[1]
- It is composed of:
  - an operating system
  - a software platform for creating apps and games
- Development Tools are free:
  - Android applications are (mostly) written in Java programming language (6 or higher)
  - Alternatively, a C++ API is available

[1] *https://source.android.com/*

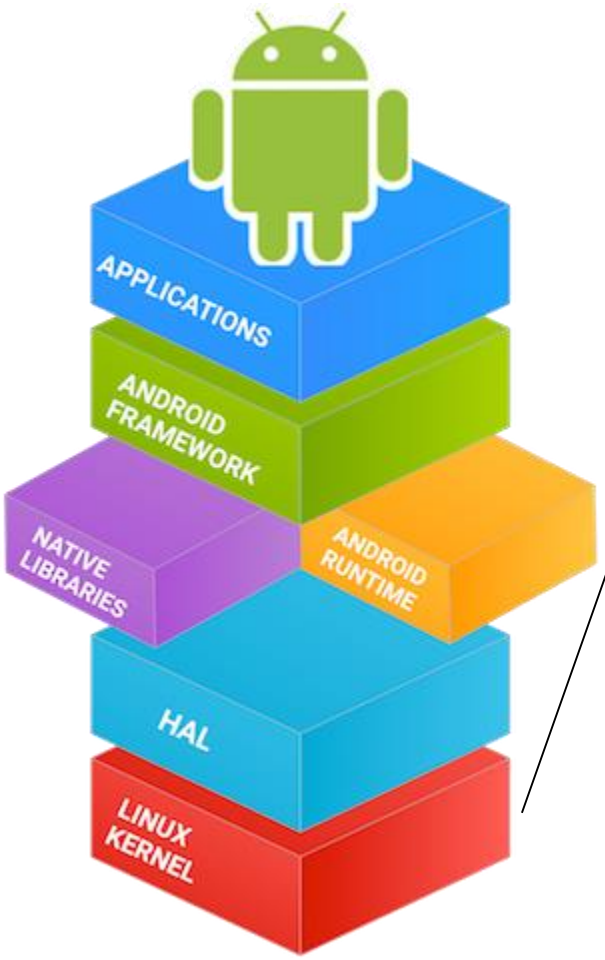
Architecture

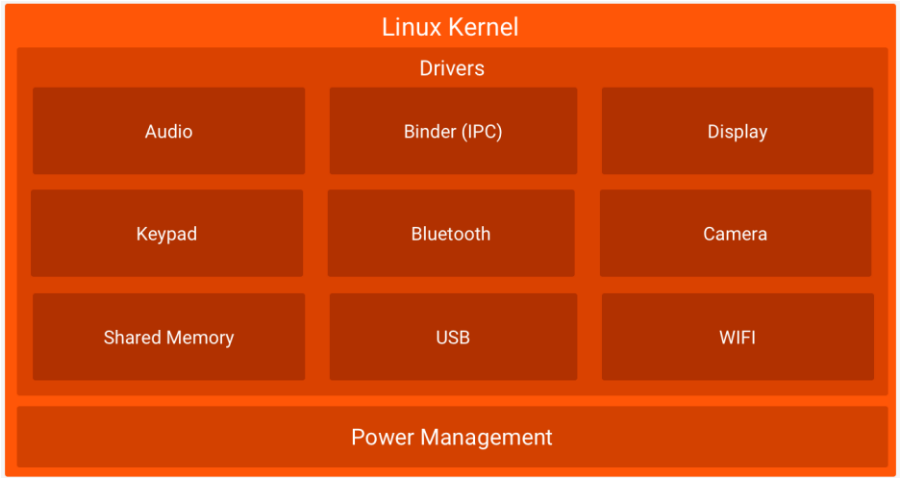
- The Android operating system is a stack of software components



Source: *https://developer.android.com/guide/platform*

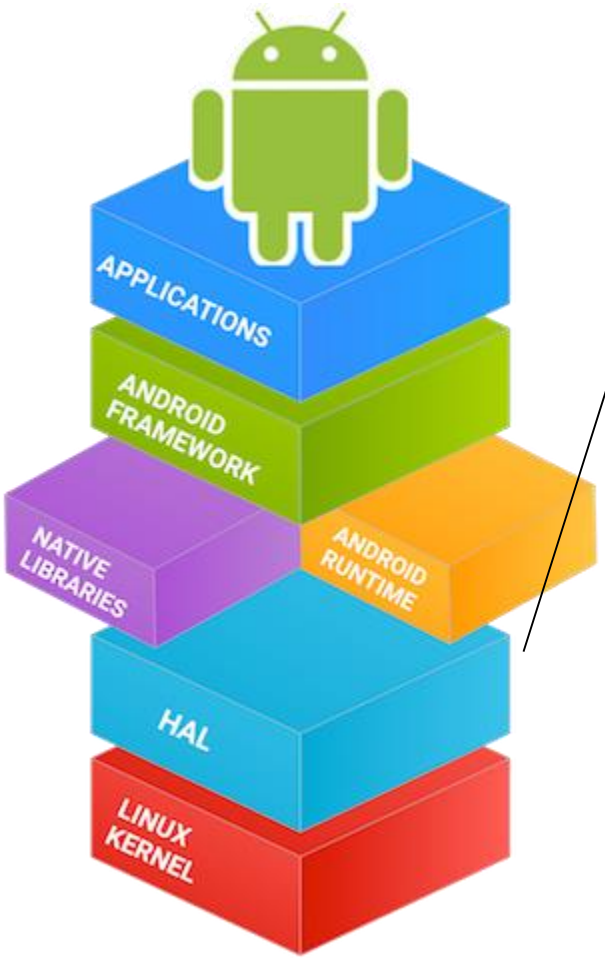
Architecture

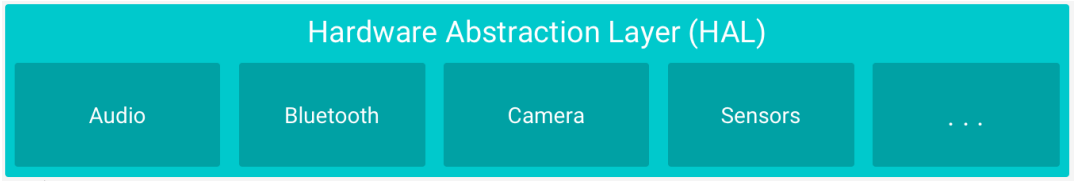




- Android is based on the Linux Kernel:
  - takes advantage of the Linux Kernel key security features
  - allows device manufacturers to develop hardware drivers for a well-known kernel

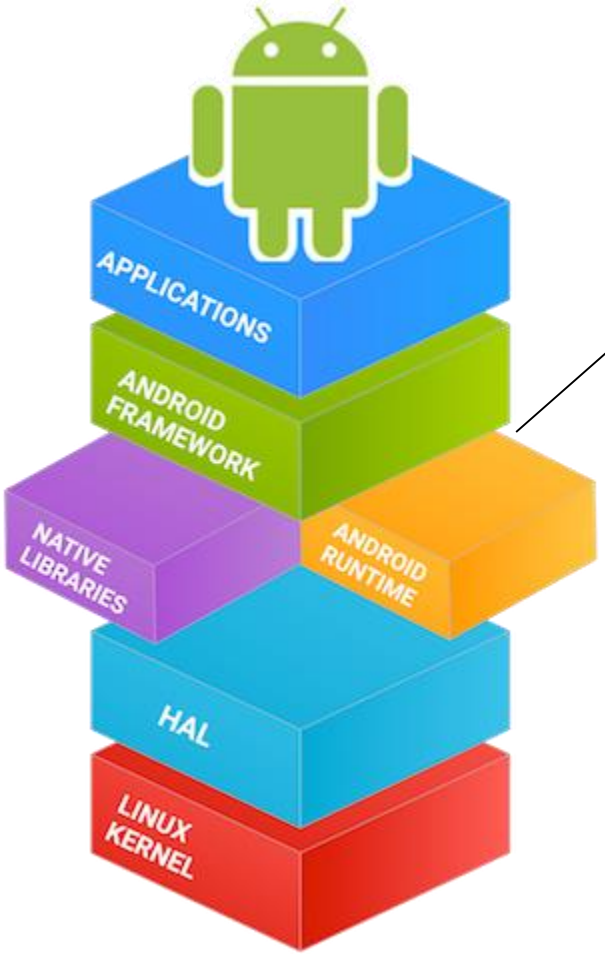
Architecture

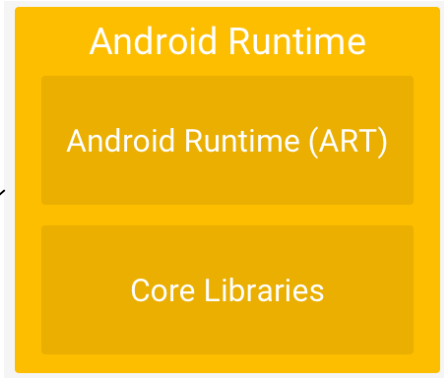




- Provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.
- It consists of multiple library modules that implement interfaces for specific type of hardware components (e.g., camera, Bluetooth …)

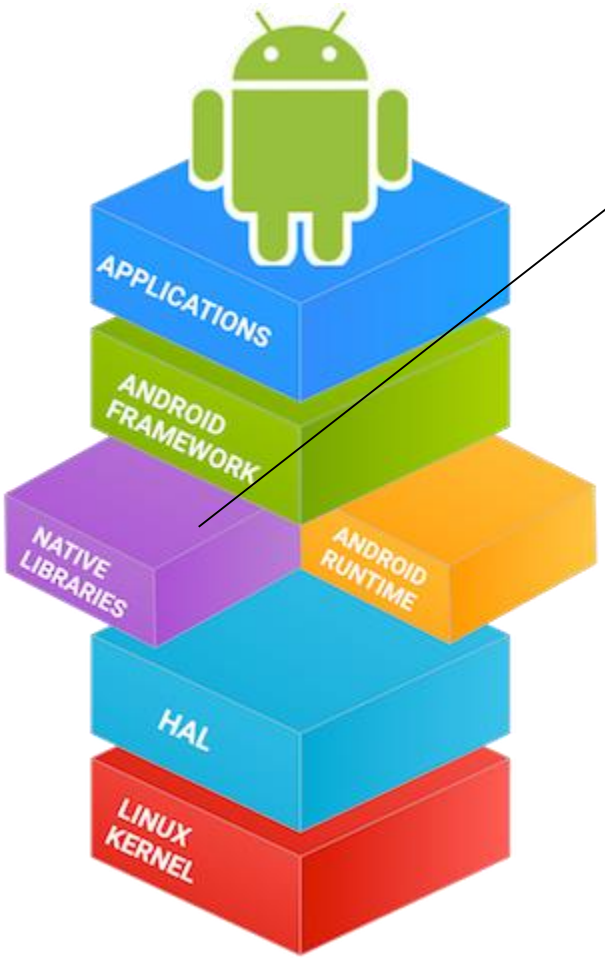
Architecture

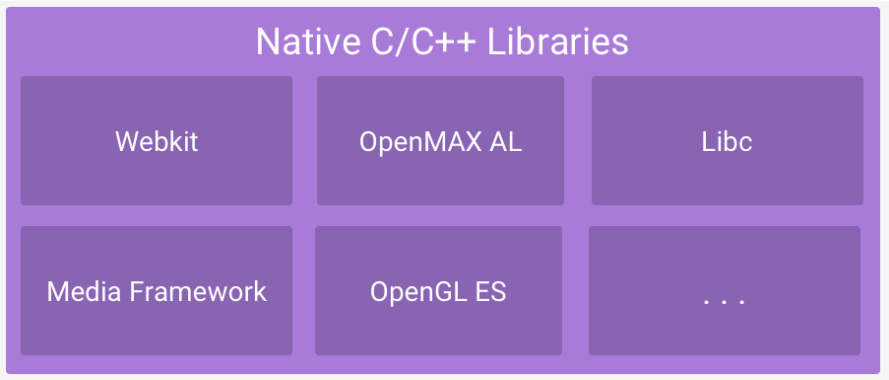




- ART is an application runtime environment (prior to Android 5.0, Dalvik used instead of ART)
- It is written to run multiple virtual machines for each running application
- Each app runs in its own process within its own instance of the Android Runtime (ART)

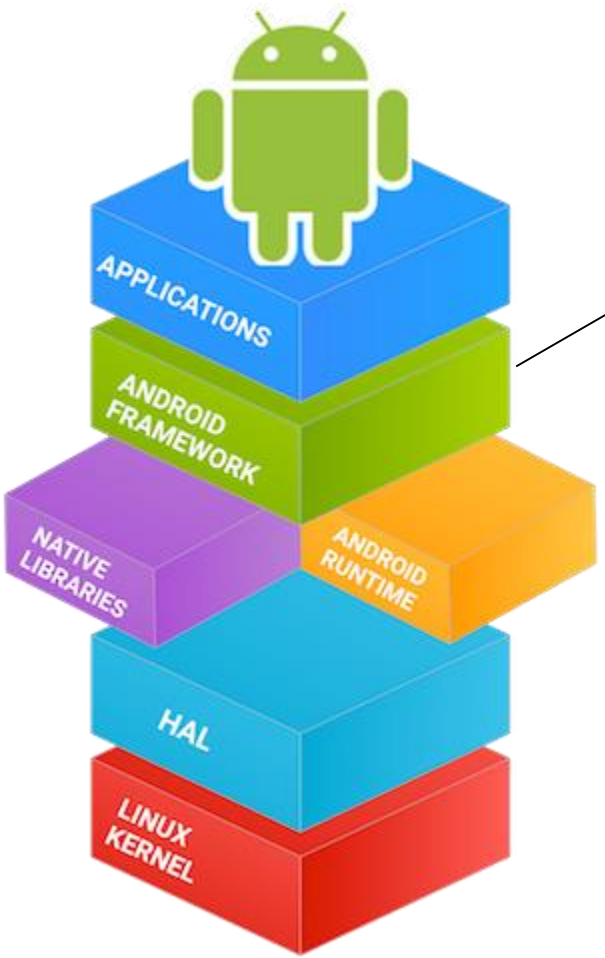
Architecture

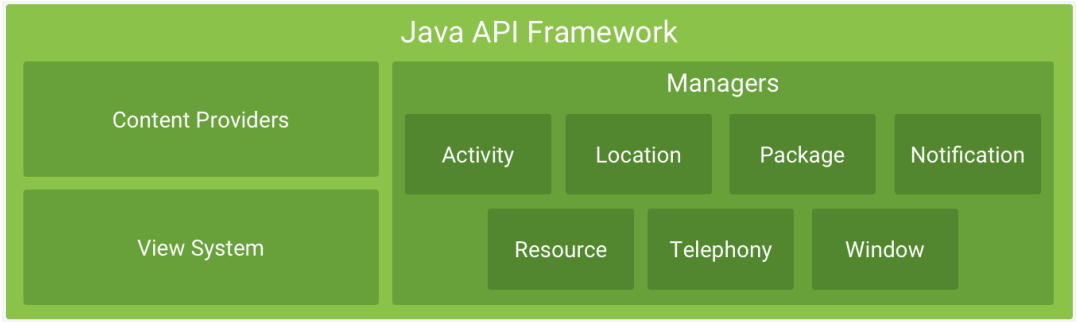




- Many core Android system components and services, (e.g., ART and HAL), are built from native code that require native libraries written in C and C++
- If you want to develop your app using C or C++, you can use the Android NDK

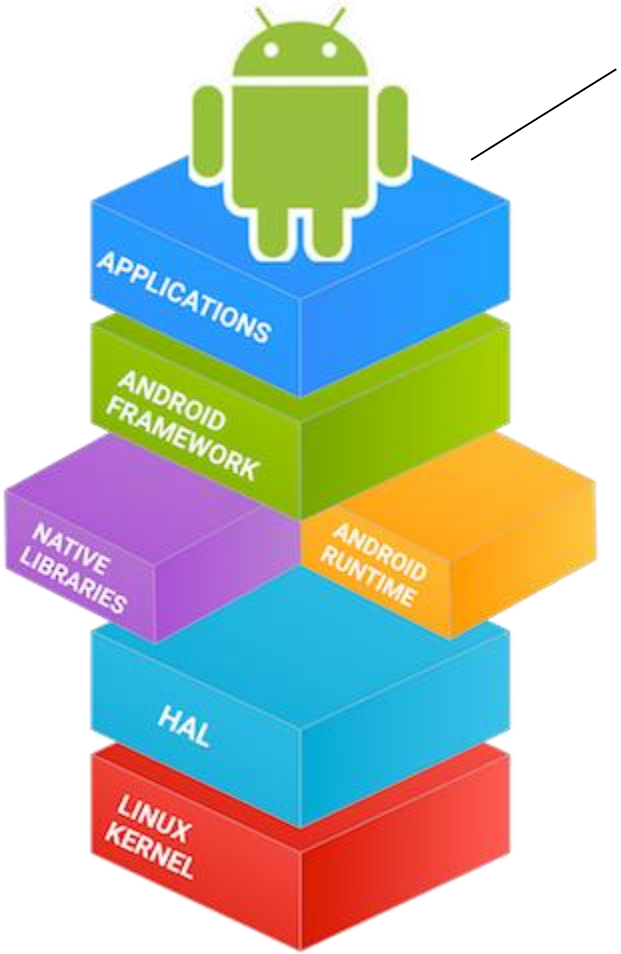
Architecture

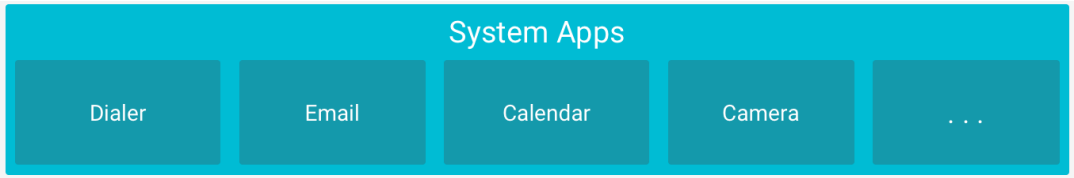




- The entire feature-set of the Android OS is available through Java APIs
- These APIs form the building blocks needed to create Android apps

Architecture





- Android comes with a set of core apps
- Android doesn't make any distinction between native and third-party applications

Android app development

DESIGN AN ANDROID APP

Design an Android app

- In Android 5 (lollipop) Google introduced **Material Design**:

  A new design metaphor inspired by paper and ink that provides a reassuring sense of tactility.

Source: *https://material.io/guidelines/material-design/introduction.html*

Design an Android app

- Material Design is based on **3 main principles**

  - **Material is the metaphor**:

    Build your app's components as real objects! Material is an object and real objects are more fun than buttons and menus.

  - **Bold, graphic, intentional**:

    The foundational elements of print-based design – typography, grids, space, scale, color, and use of imagery – create hierarchy, **meaning**, and focus.

  - **Motion provides meaning**

    Motion respects and reinforces the user as the prime mover; it is meaningful and appropriate, serving to focus attention and maintain continuity.

Design an Android app: **MVC**

- Best practice in developing an Android application: use the Model View Controller (MVC) pattern

Design an Android app: **MVC**

- Best practice in developing an Android application: use the Model View Controller (MVC) pattern
- MVC is composed by 3 components:

Design an Android app: **MVC**

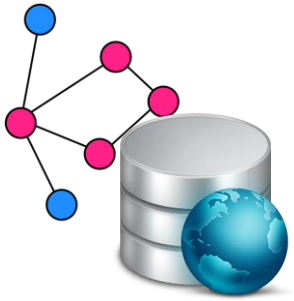
- Best practice in developing an Android application: use the Model View Controller (MVC) pattern
- MVC is composed by 3 components:

View

Design an Android app: **MVC**

- Best practice in developing an Android application: use the Model View Controller (MVC) pattern
- MVC is composed by 3 components:

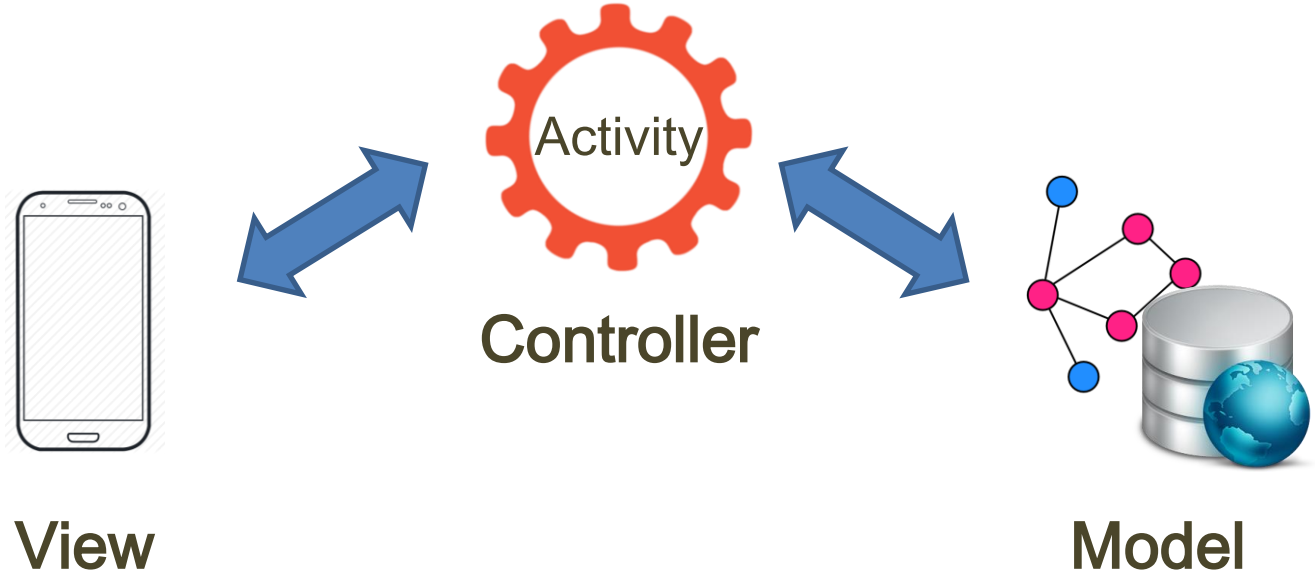
View



Model

Design an Android app: **MVC**

- Best practice in developing an Android application: use the Model View Controller (MVC) pattern
- MVC is composed by 3 components:



Android app development

TOOLS

Android app development: **Tools**

- To develop an Android app we need:
  - **Java SDK** (6 or higher)
  - **Android SDK** (included in Android Studio)
  - **Android Integrated Development Environment**
    - **Android Studio** (official IDE, used in this hands-on)
    - Eclipse ADT
  - Android SDK emulator (included in Android Studio)
    - Can in general be quite slow
  - Third-party emulators
    - Based on hardware virtualization
    - Typically faster
    - E.g., GenyMotion

Android app development

APPLICATION FUNDAMENTALS

Application fundamentals

- Android apps are written in the Java programming language.
- Each process is run in its own virtual machine (VM), so an app's code runs in isolation from other apps.
- Once installed on a device, each Android app lives in its own security sandbox:
  - The Android operating system is a multi-user Linux system in which each app is a different user.
  - By default, the system assigns each app a unique Linux user ID. The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.

Application Main Components

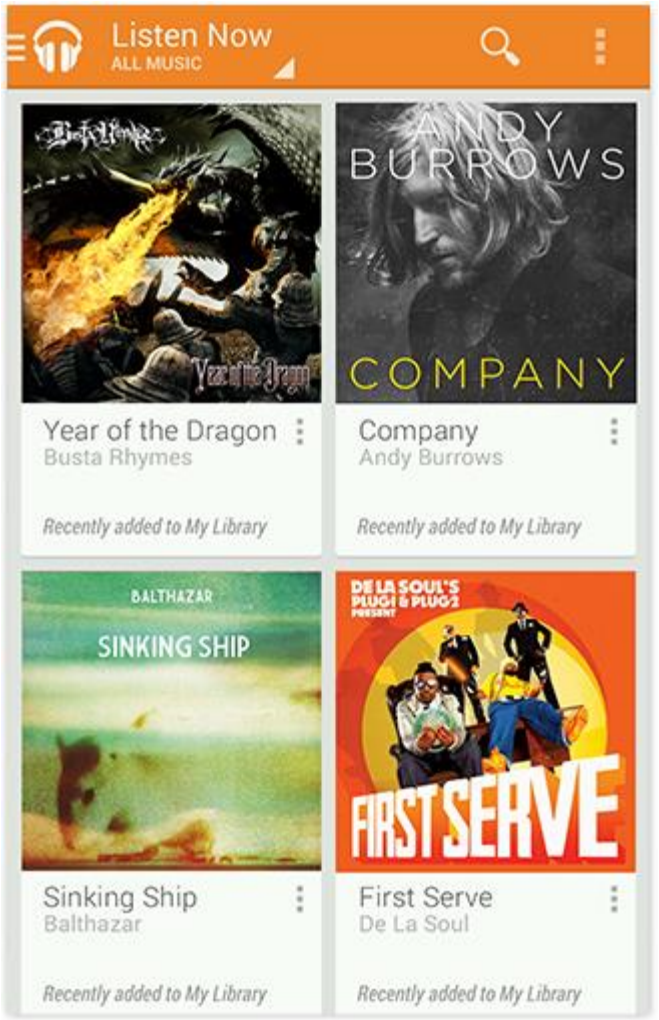
- Each application consists of one or more of the following components:
  - Activities
  - Services
  - Content providers
  - Broadcast receivers

- Each of them takes care of a specific interaction inside the Operating System
  - It is activated according to a specific life cycle

Source: *https://developer.android.com/guide/components*

Activity

- An activity represents a single screen with a user interface.
- An application is composed by one or more activities.
  - For example, an email app might have the following activities
    - one that shows a list of new emails;
    - one to compose an email;
    - one for reading emails.

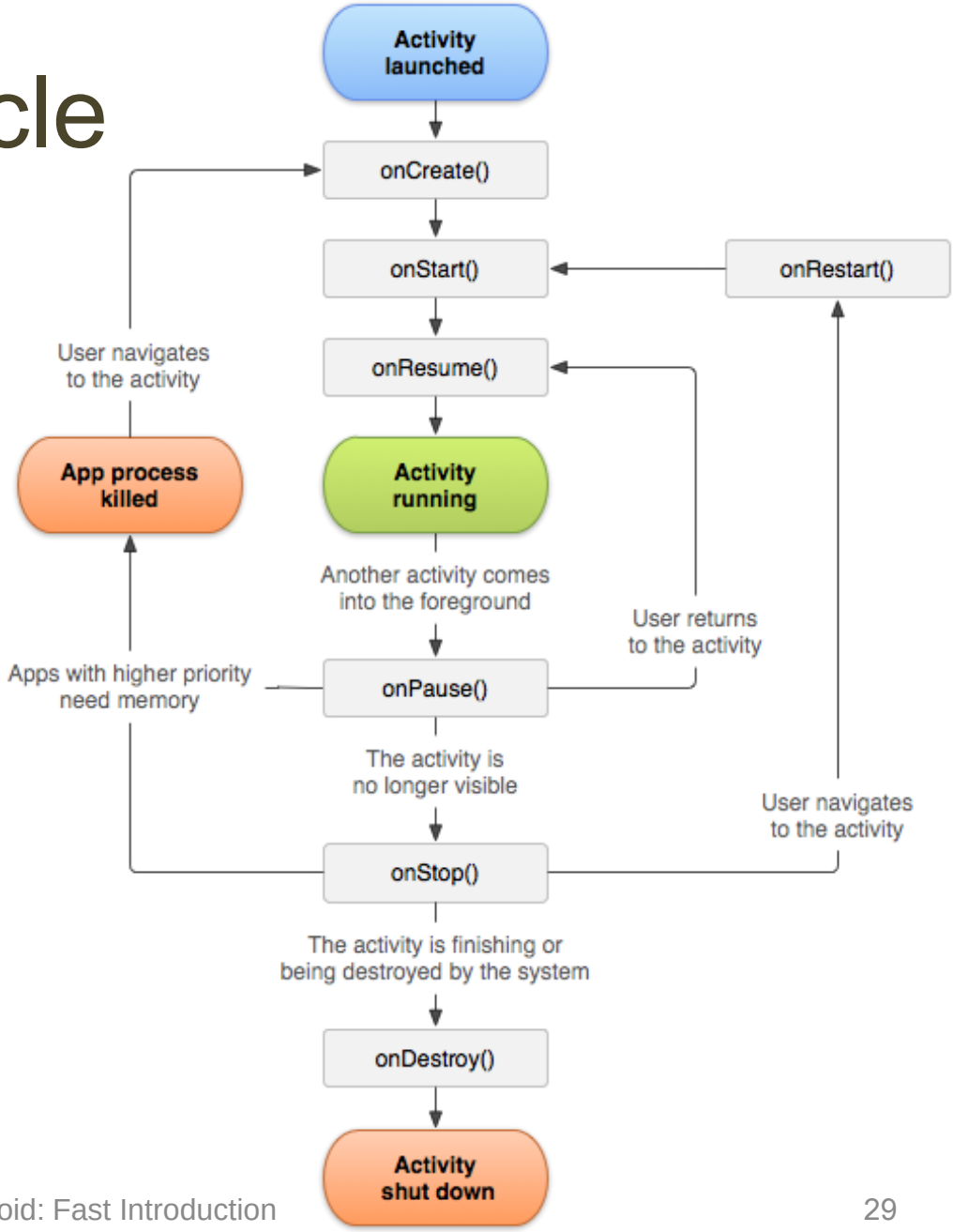


Activity

- A “main” activity is mandatory in each application
  - it is presented to the user when she launches the application for the first time.
- Each activity can start another activity
  - to perform different actions.
- Each time a new activity starts:
  - the previous activity is stopped,
  - the system preserves the activity in a stack (the "back stack").
- When a new activity starts:
  - it is pushed onto the back stack and takes user focus

Activity - Lifecycle

- As a user navigates through, out of, and back to an app, the Activity instances in this app transit through **different states**



Activity - Handling lifecycle

```java
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

Activity - Handling lifecycle

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

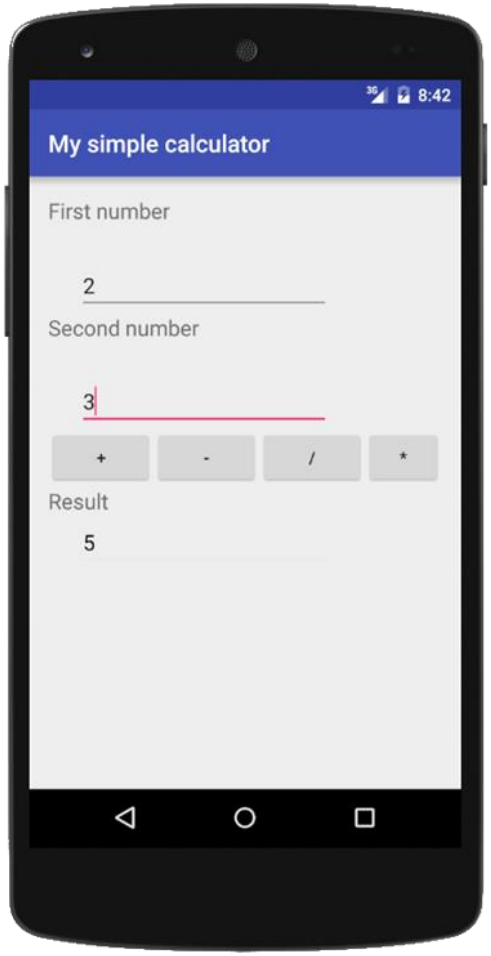
Activity - onCreate()

- The method onCreate(Bundle b) is called in two different scenarios
  - When the activity is run for the first time
    - The Bundle parameter is null
  - When the activity is launched again after being terminated (due to lack of resources or for other reasons)
    - The Bundle parameter contains status information
- This is where all normal static set up should be done: create views, bind data to lists, etc.
- It is always followed by onStart().

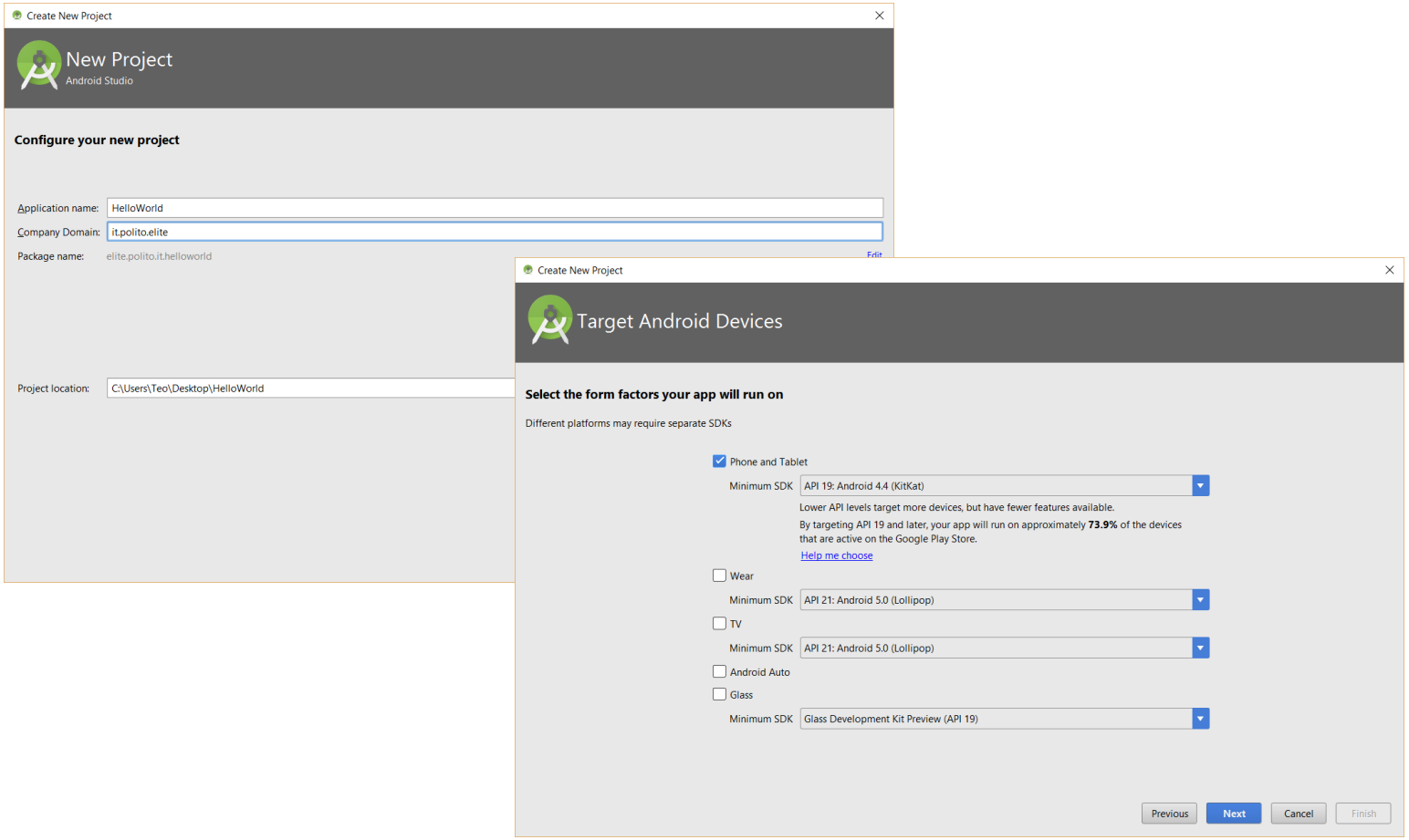
Activity - onStart()

- Called when the activity is becoming visible to the user.
- Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.

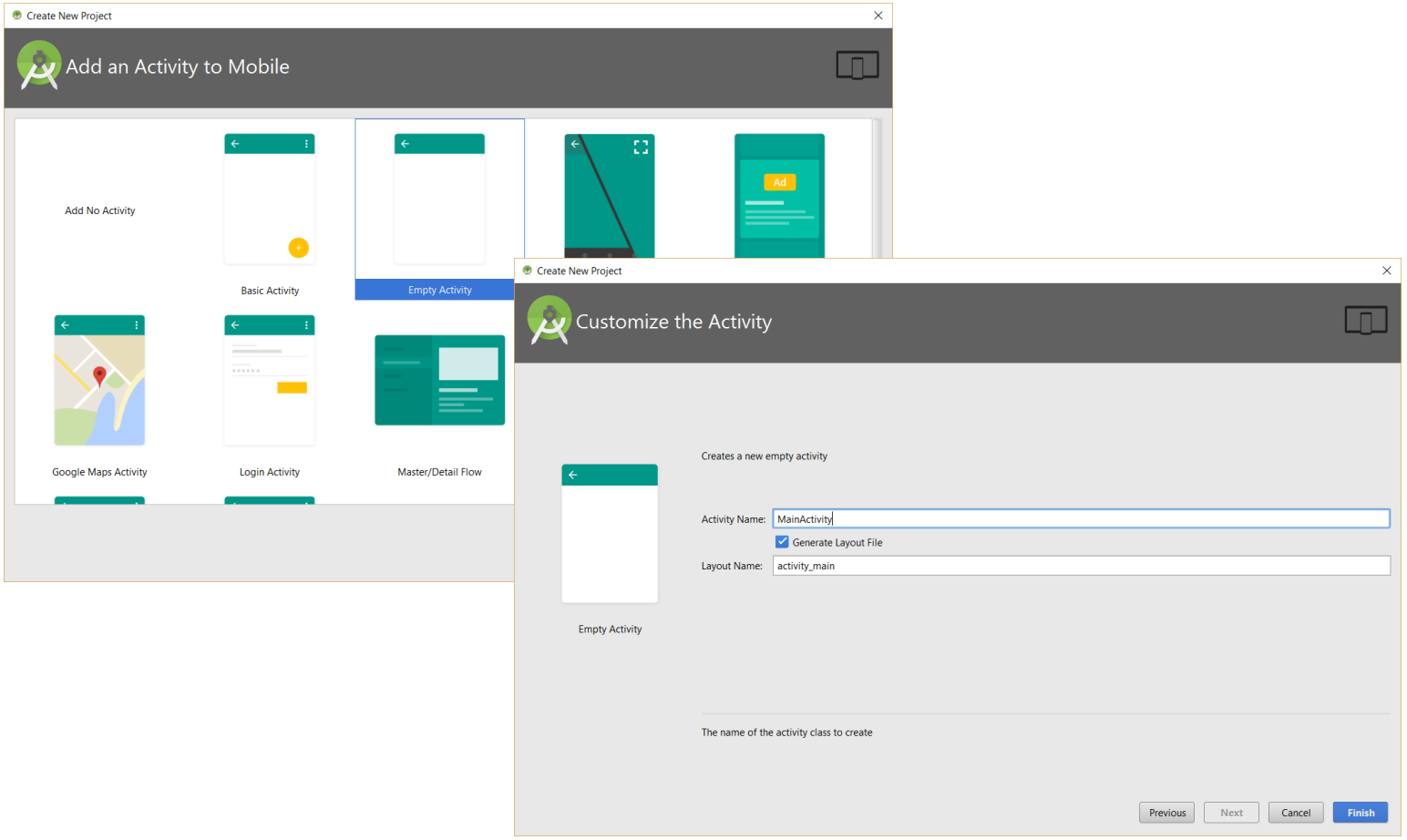
Hands on: simple calculator



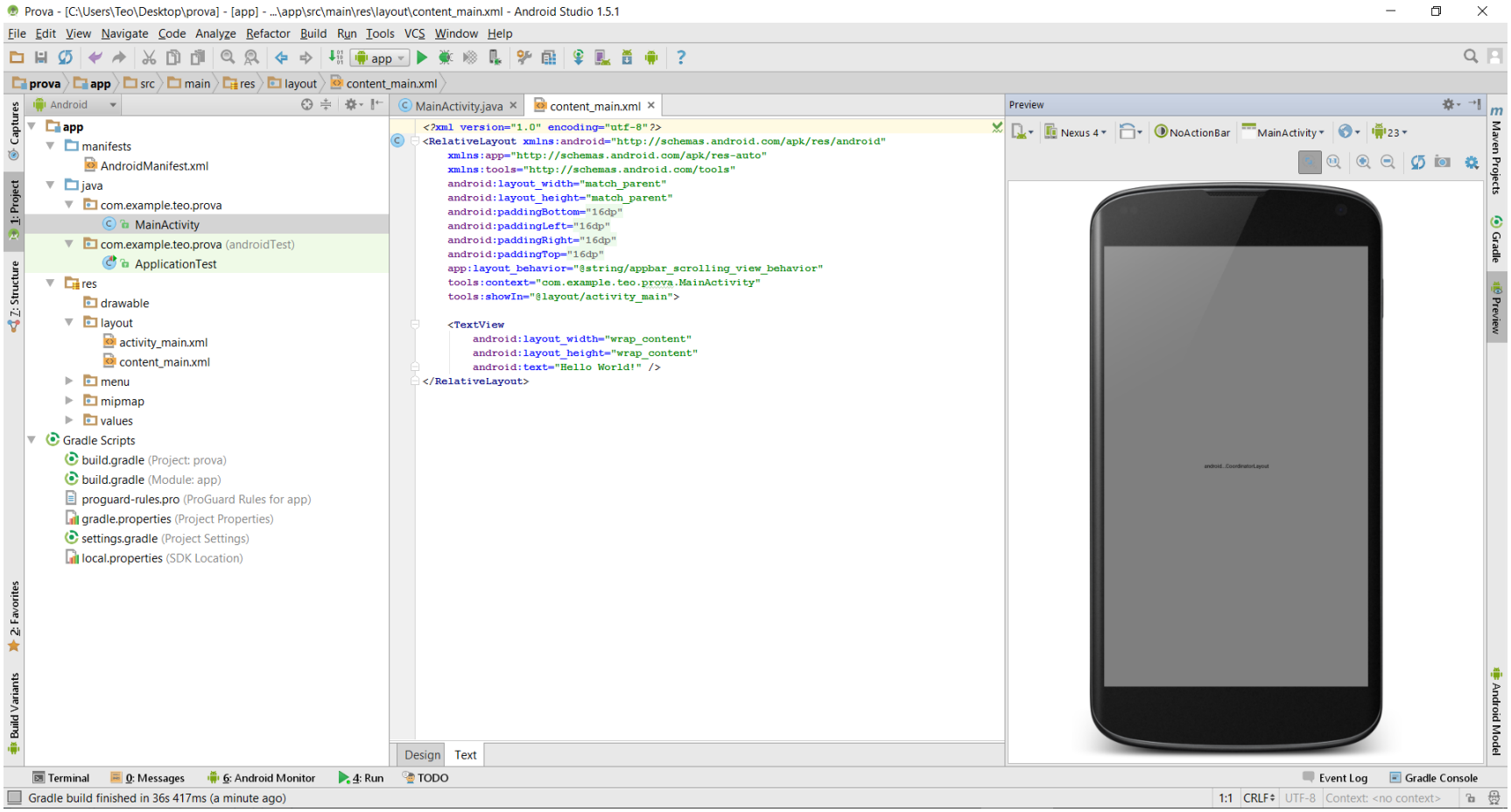
Android Studio: Create a new app



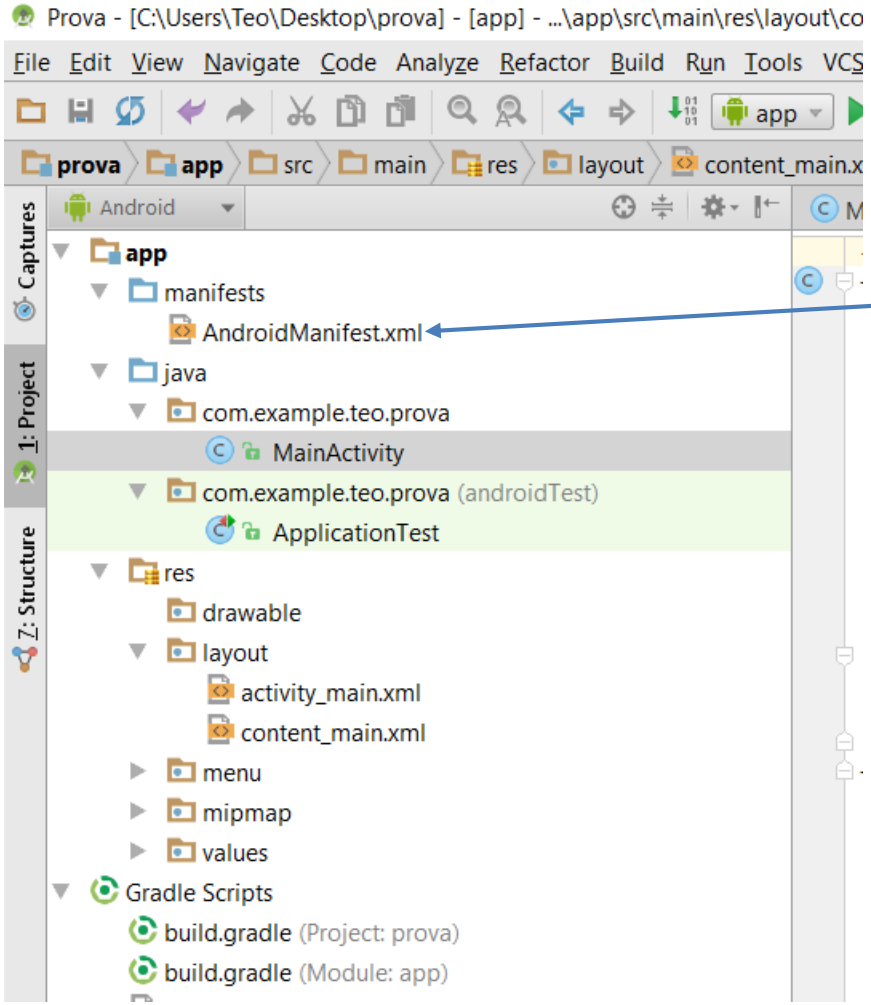
Android Studio: Create a new app



App structure

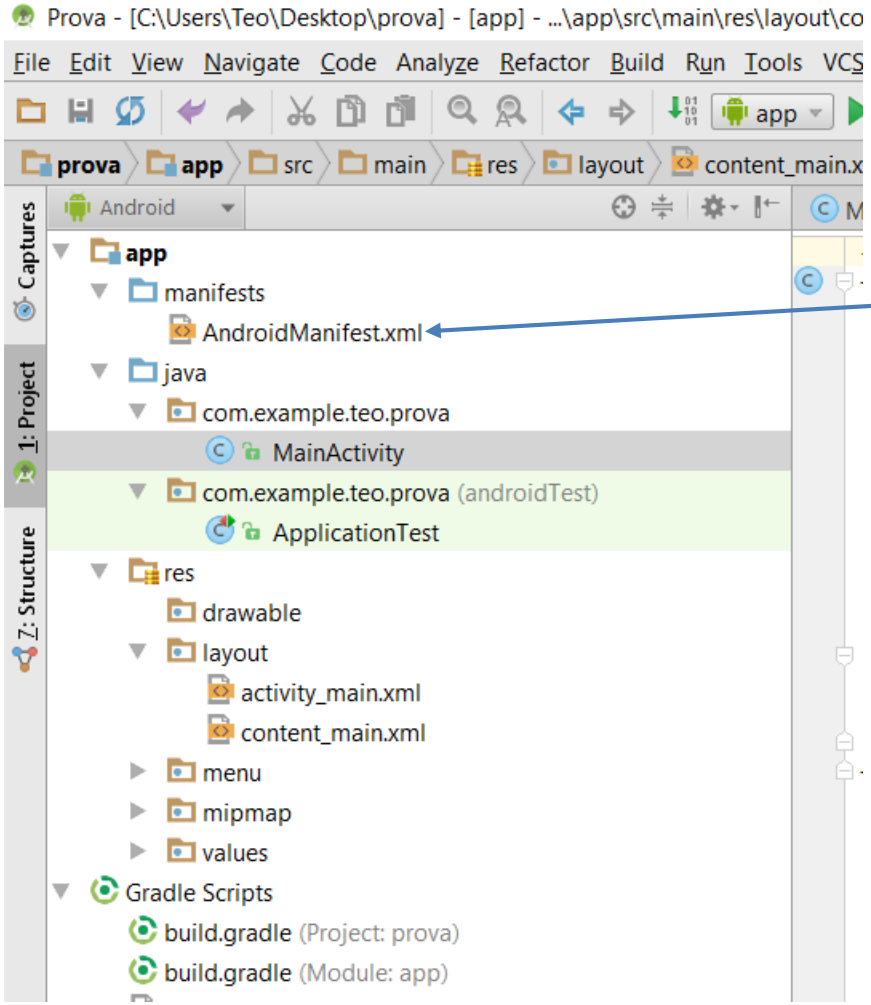


App structure



- The manifest file describes the fundamental characteristics of the app and defines each of its components.

App structure



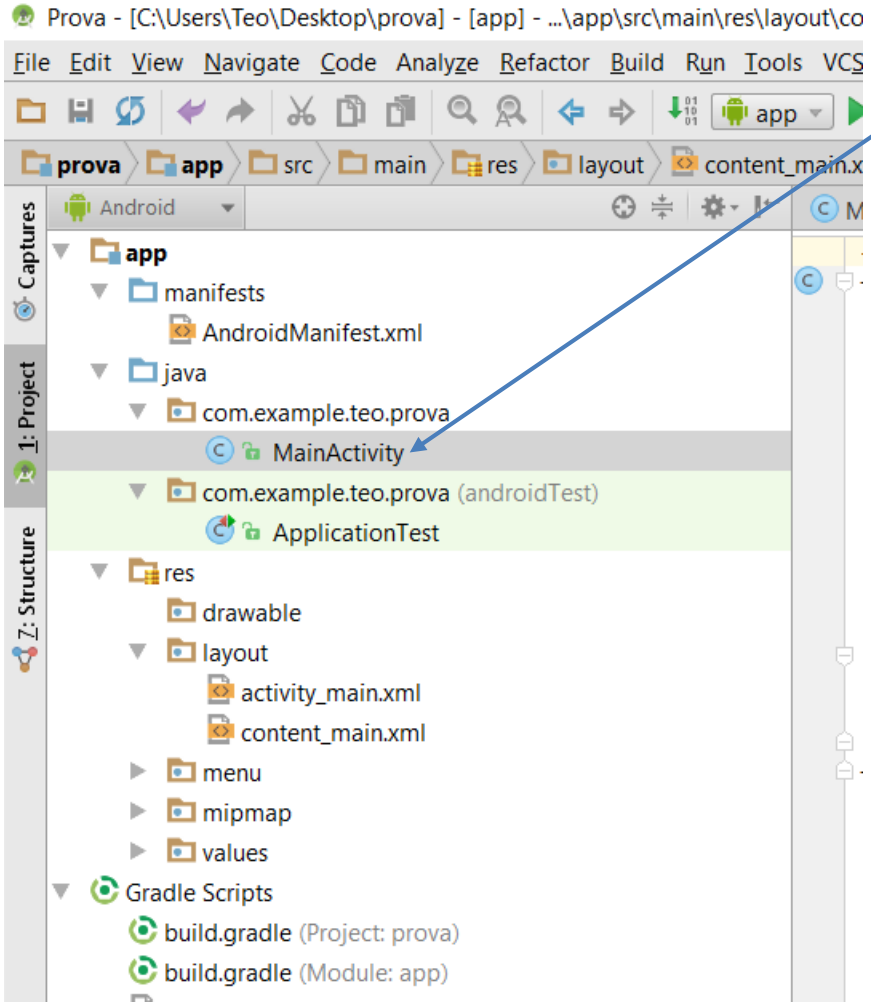
- Among other things, the manifest contains the following elements:
  - all the single components that compose the application
  - the requested permissions
  - the app configurations information

App structure

Example of a Manifest file

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Prova"
        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>
```

App structure



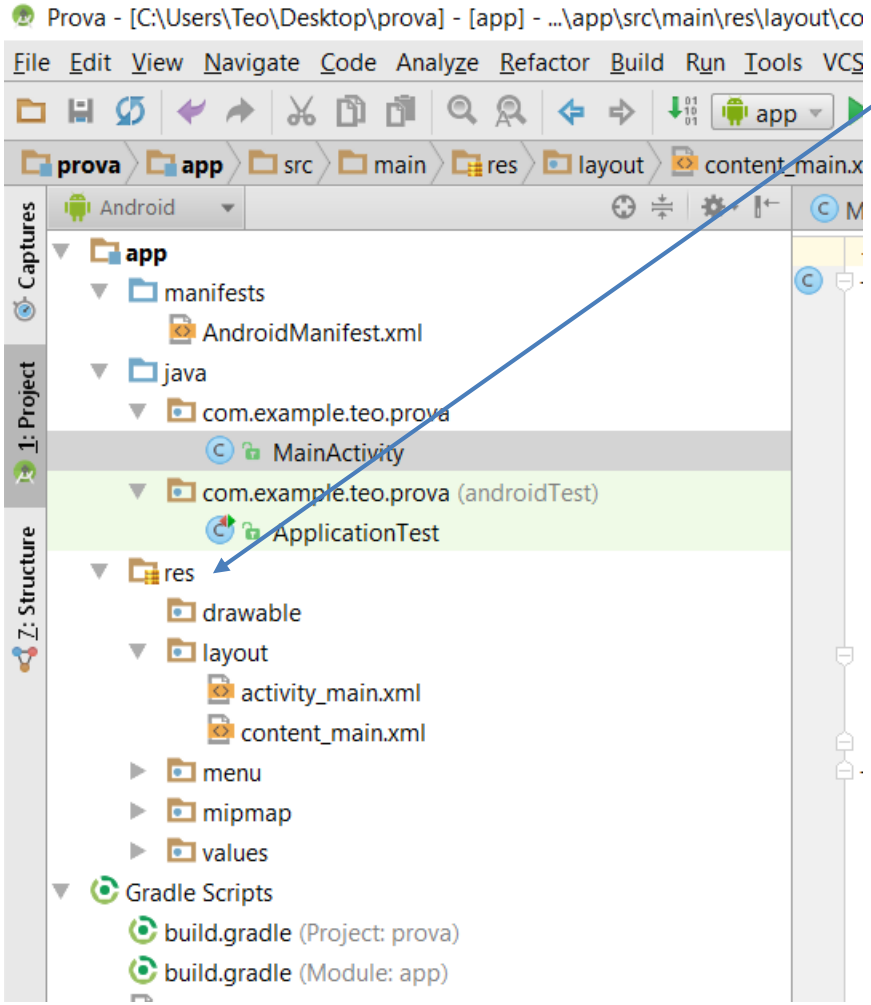
- The MainActivity.java file contains the class definition for the main activity.
- When the app is built and run, the Activity class starts the activity and runs the contained code
- It implements the **Controller** of the MVC pattern

App structure

Example: MainActivity.java

```
package elite.polito.it.prova;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

App structure



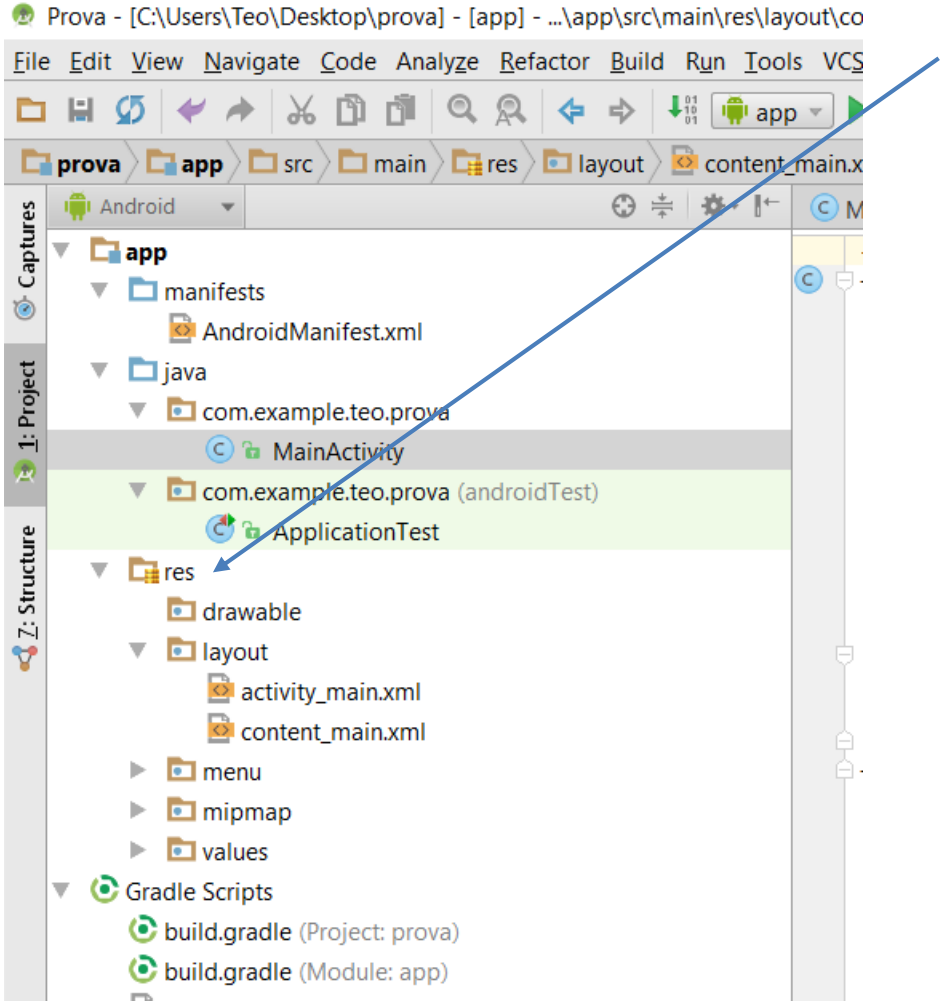
- The res folder contains the resources for the application (it represents the **View** of the MVC pattern):
  - drawable-<density>/

    Directories for drawable resources, other than launcher icons, designed for various densities (e.g., drawable-hdpi)
  - layout/

    Directory for files that define app's user interface like activity_main.xml
  - menu/

    Directory for files that define app's menu items.

App structure



- …
- mipmap/

  Launcher icons reside in the mipmap/ folder. This folder contains the ic_launcher.png image that appears when you run the default app.
- values/

  Directory for other XML files that contain a collection of resources, such as string (e.g., strings definition for different languages) and color definitions.

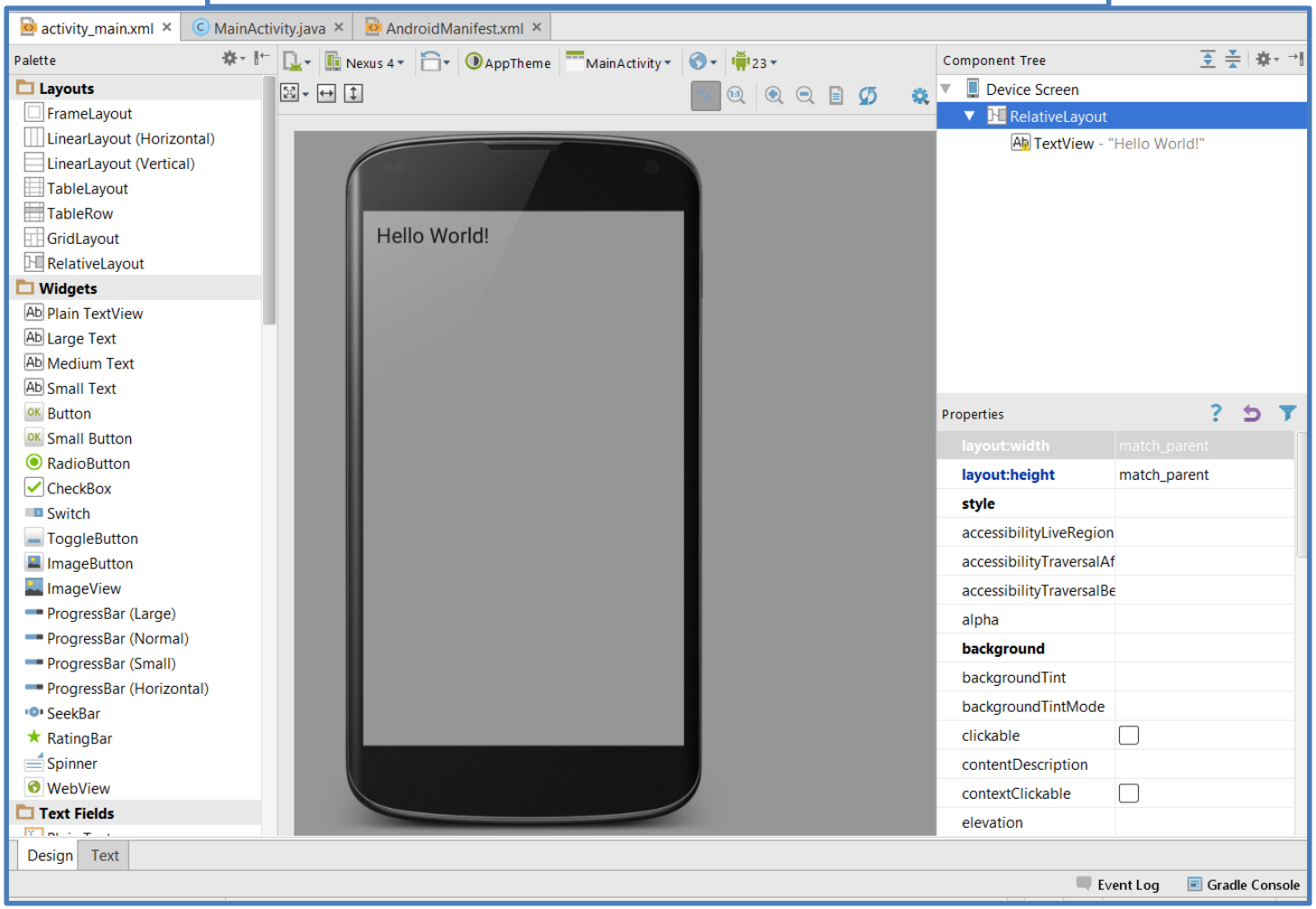
App structure: layout example

Example of activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="elite.polito.it.prova.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

App structure: layout example

Example of activity_main.xml – Design mode



Layout alternatives

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

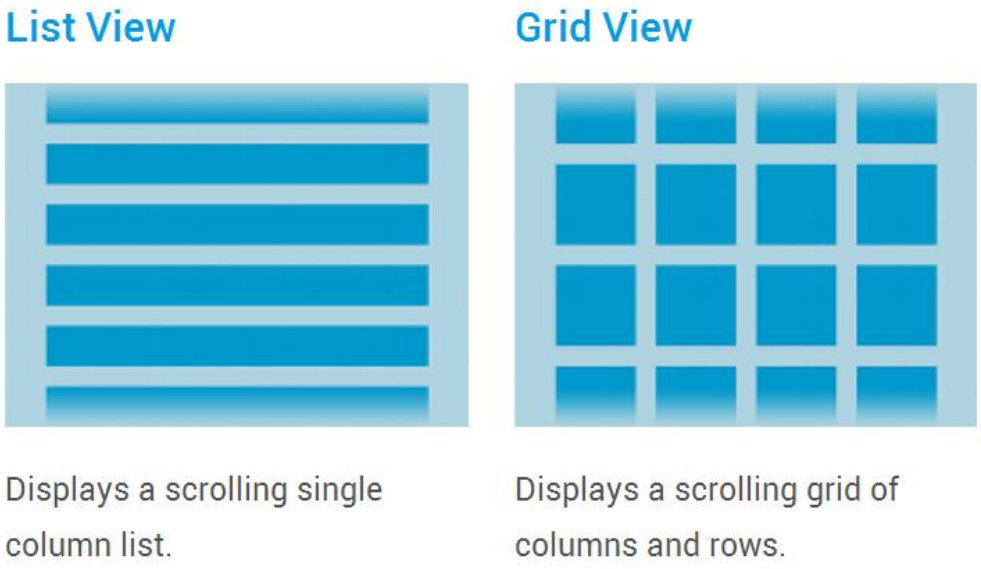
Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Web View

Displays web pages.

Layout alternatives

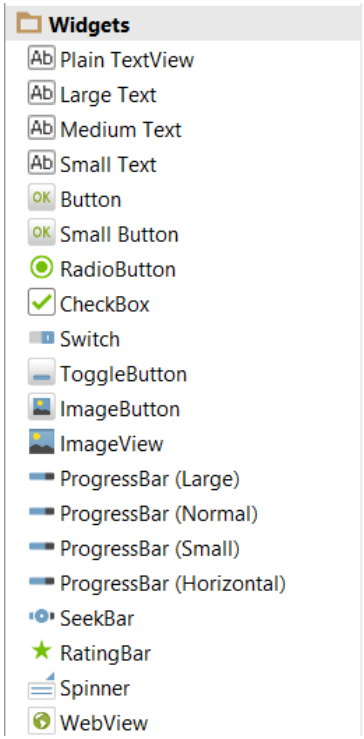
- When the content for your layout is dynamic or not pre-determined, you can use one of these 2 layouts.

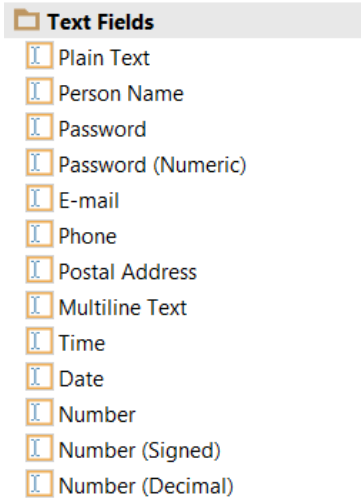


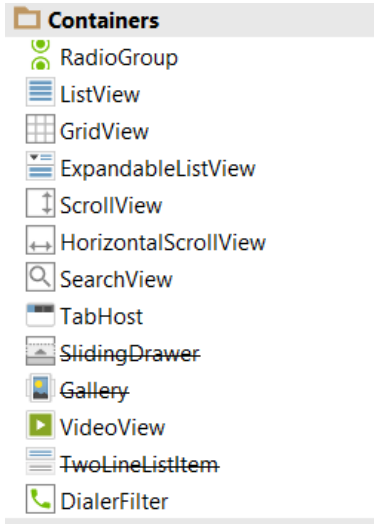
More details at http://developer.android.com/guide/topics/ui/declaring-layout.html

Widgets

- Every widget (used to create interactive UI components (buttons, text fields, etc.)) is a subclass of the View class



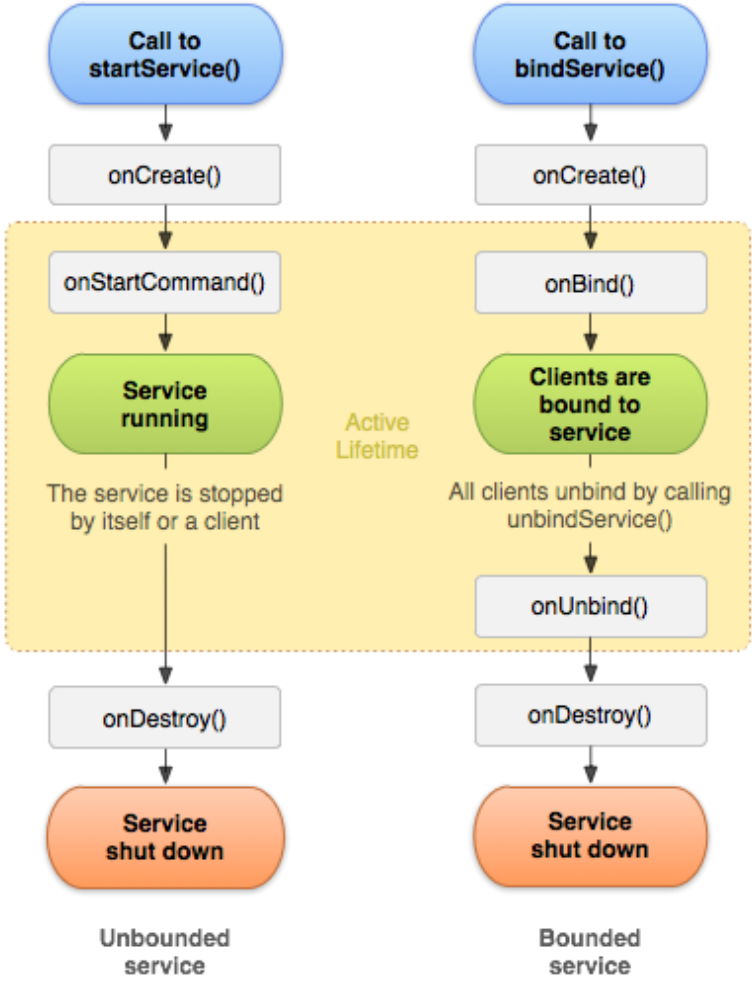




Service

- A service is a component that:
  - runs in the background to perform long-running operations or to perform work for remote processes
  - does not provide a user interface (e.g., play music in the background while the user is in a different app).
  - is a subclass of android.app.Service
- Any application component can use the service (even from a separate application)
  - by starting it with an “Intent” (we will see it later).

Service lifecycle



Content provider

- A content provider manages a shared set of app data
  - Data can be stored in the file system, in an SQLite database, on the web, or in any other persistent storage location the app can access, …
- It implements a set of standard methods that allow other applications to fetch and to store data handled by the current application
  - Other applications do not call its method directly, but they interact via a content

Broadcasts

- Android apps can send or receive **broadcast messages** from the Android system and other Android apps
- These broadcasts are sent when an event of interest occurs

- A Broadcast Receiver is a component which “waits” for messages
- Broadcast Receivers don't display a user interface
- Intended to do a very minimal amount of work (e.g., initiate a service to perform some work)

Other important stuff: **app life-cycle**

- A Linux process is created for an application when some of its code needs to be run.
- In an ideal case
  - all Android processes started by the user remain in memory
    - Faster restart
- But
  - the available memory on an Android device is limited
- Therefore
  - the Android system is allowed to terminate running processes or recycling Android components (lifecycle will be seen after app components explanation)

Other important stuff: **app life-cycle**

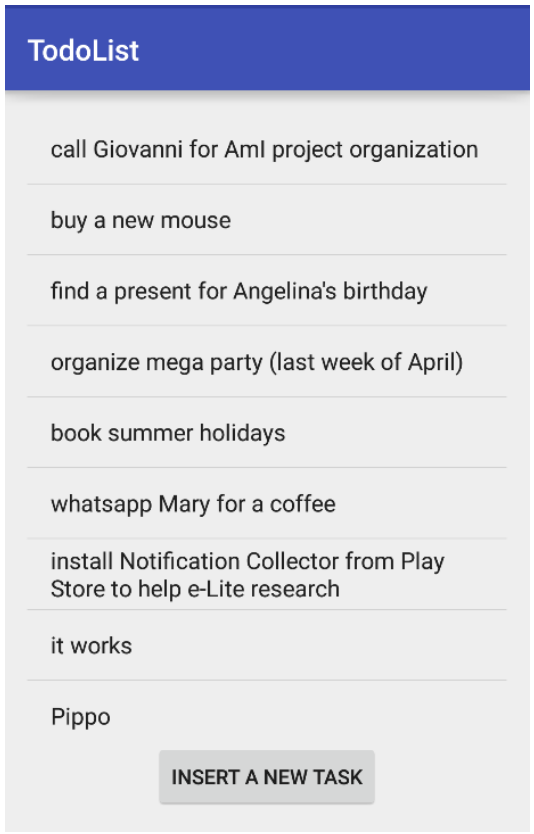
Android: Processes and Application Life Cycle

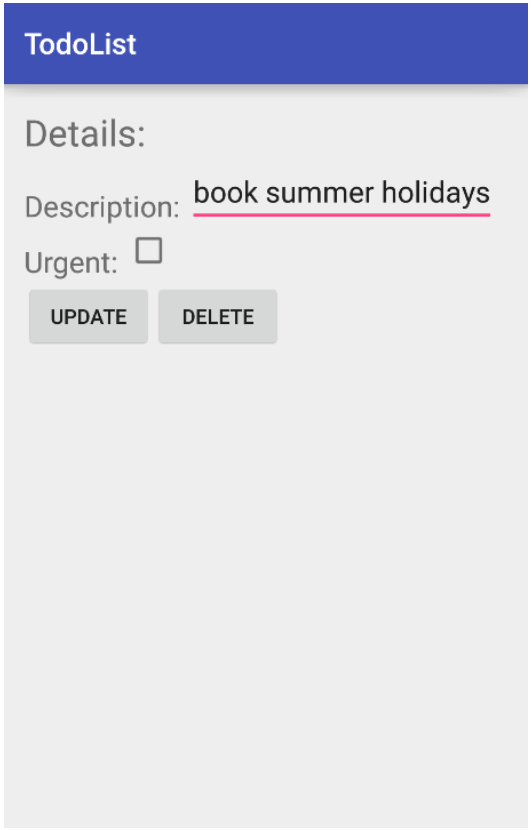
Process status	Description	Priority
Foreground	A process that: a) is running an Activity at the top of the screen that the user is interacting with, b) has a BroadcastReceiver that is currently running, c) has a Service that is currently executing code in one of its callbacks (Service.onCreate(), Service.onStart(), or Service.onDestroy())	1
Visible	User is not interacting with the activity, but the activity is still (partially) visible. This may occur, for example, if the foreground Activity is displayed as a dialog that allows the previous Activity to be seen behind it.	2
Service	A process that is holding a Service that has been started with the startService() method. These processes are not directly visible to the user	3
Background	A process that is holding an Activity that is not currently visible to the user. Android keeps them in a least recent used (LRU) list and if requires terminates the one which was least used.	4
Empty	Process that doesn't hold any active application components	5

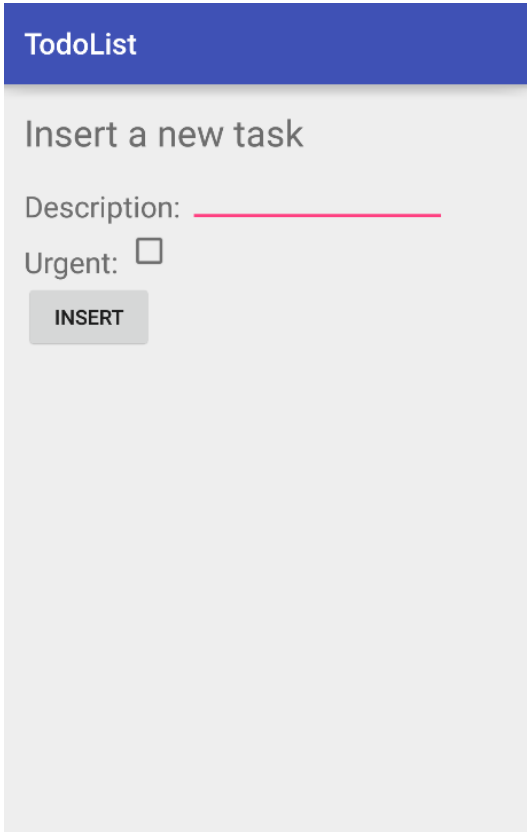
Other important stuff: **Intents**

- An Intent is a messaging object you can use to request an action from another app component.
- 3 main uses:
  - To start an activity
    - by passing an Intent to startActivity()
  - To start a service
  - To deliver a broadcast
    - by passing an Intent to sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().

Hands-on: Todo List







Questions?

01QZP AMBIENT INTELLIGENCE

Teodoro Montanaro
teodoro.montanaro@polito.it