

Introduction to Android

Ambient intelligence: technology and design

Teodoro Montanaro

Politecnico di Torino, 2015/2016



ANDROID



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
- Application fundamentals
- Application Main Components: Activity
- Android app development:
 - Design
 - MVC
 - Tools
 - App structure
 - Hands-on: simple calculator
- Application Main Components: part2
- Other important stuff:
 - Intent
- Hands-on

History

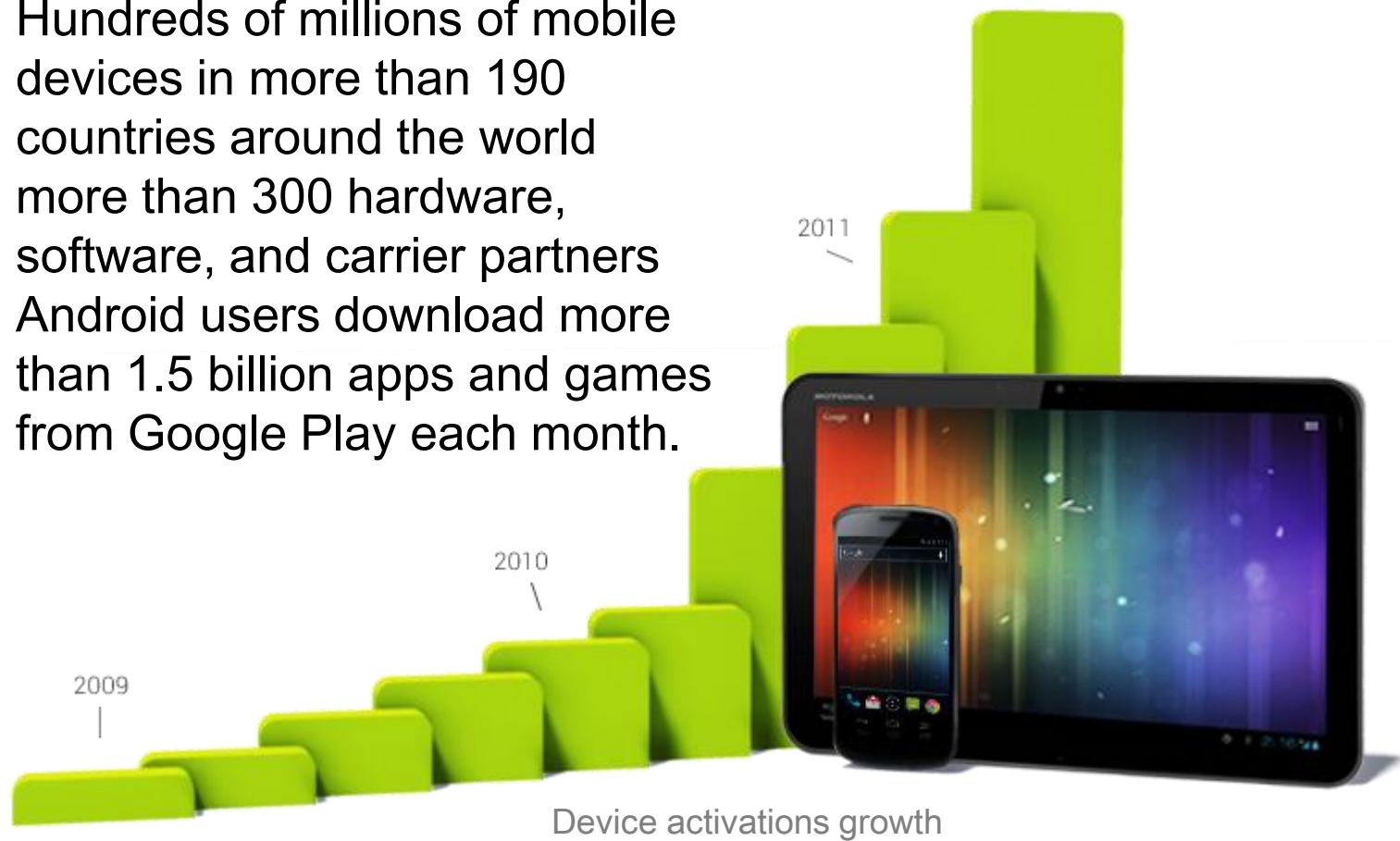
- Originally created by Andy Rubin
- Acquired by Google Inc. in 2005
- Now it is maintained by the Open Handset Alliance (OHA) (since 2007)
- Several stable releases since then

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2%
4.1.x	Jelly Bean	16	7.8%
4.2.x		17	10.5%
4.3		18	3.0%
4.4	KitKat	19	33.4%
5.0	Lollipop	21	16.4%
5.1		22	19.4%
6.0	Marshmallow	23	4.6%

Last update:
3 May 2016
(<http://developer.android.com/about/dashboards/index.html>)

Figures

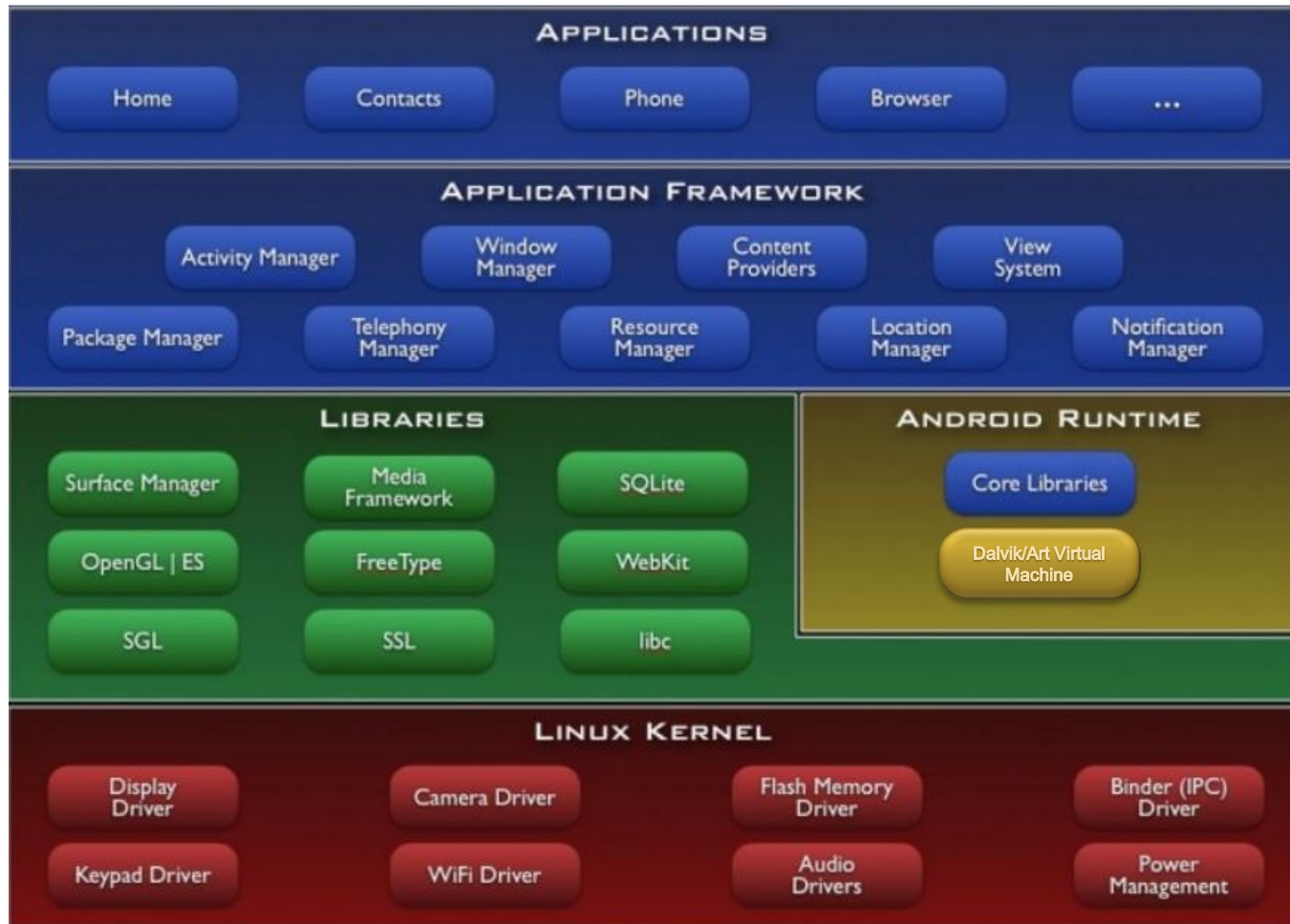
- Hundreds of millions of mobile devices in more than 190 countries around the world
- more than 300 hardware, software, and carrier partners
- Android users download more than 1.5 billion apps and games from Google Play each month.



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”.
- 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

Architecture



Architecture: ART/Dalvik VM

- **Android Runtime (ART)**: an application runtime environment used by the Android operating system.
- In Android 5.0 **ART** replaces **Dalvik**, which is the process virtual machine originally used by Android.
- Every Android application runs in its own process, with its own instance of the ART (Dalvik) virtual machine
- Android programs are compiled into Dalvik/ART executable files (.dex) which are then zipped into Android packages (.apk).

Application fundamentals

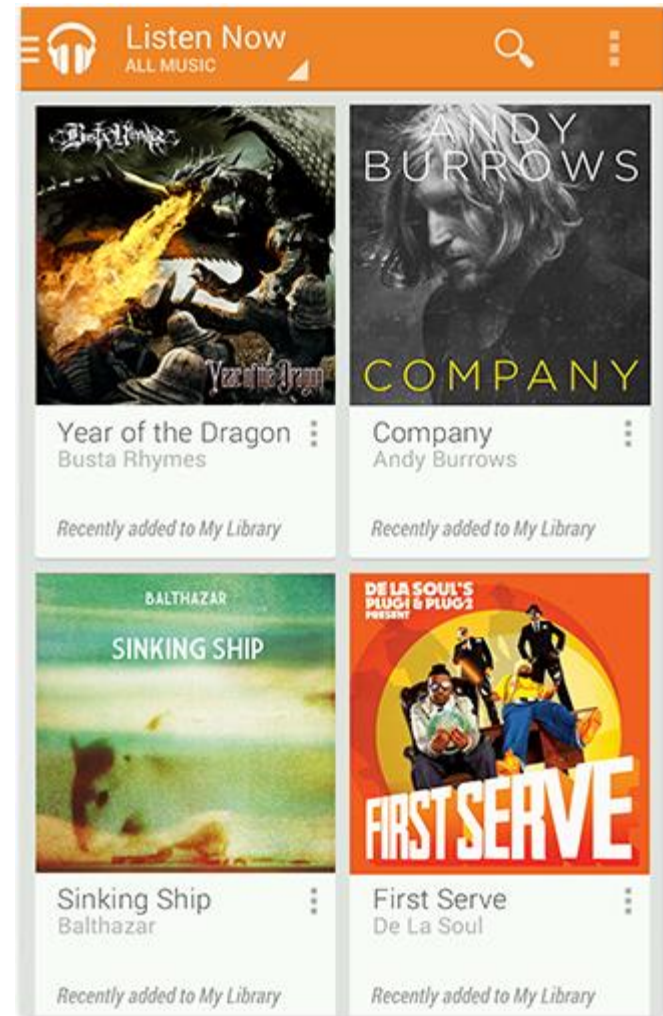
- Android apps are written in the Java programming language.
- Each process is run in its own virtual machine (VM), so an app's code runs in isolation from other apps.
- Once installed on a device, each Android app lives in its own security sandbox:
 - The Android operating system is a multi-user Linux 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

Activity

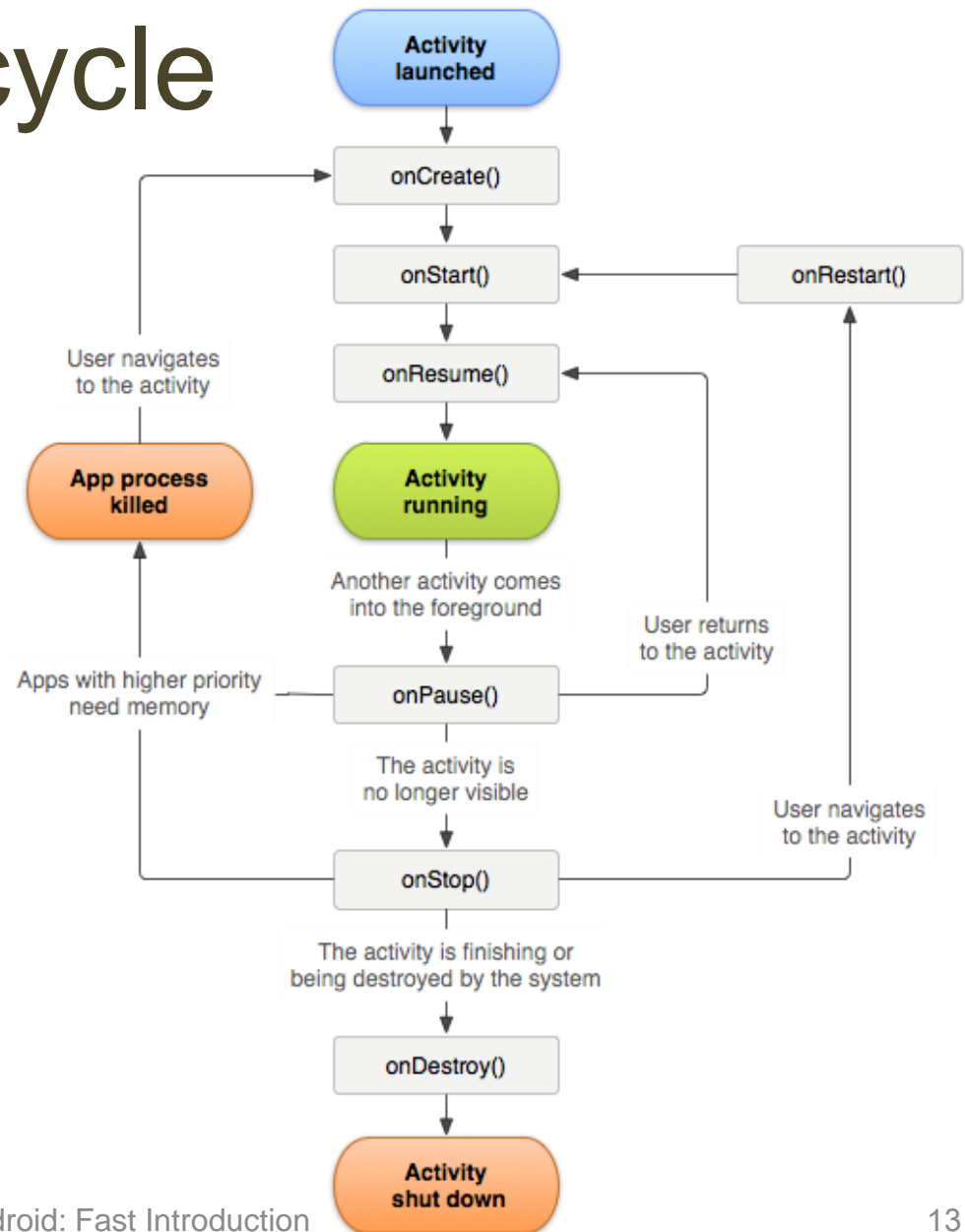
- An activity represents a single screen with a user interface.
- An application is composed by one or more activities.
 - For example, an email app might have the following activities
 - one that shows a list of new emails;
 - one to compose an email;
 - one for reading emails.



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



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 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 - 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.

Android app development: Design

- In Android 5 (lollipop) Google introduced **Material Design**:
A new design metaphor inspired by paper and ink that provides a reassuring sense of tactility.



Android app development: Design

- Material Design is based on **3 main principles**
 - **Material is the metaphor:**

Material is the metaphor. And Material is an Object. 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

More details at <https://www.google.com/design/spec/material-design/introduction.html>

Android app development: MVC

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

Android app development: MVC

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

Android app development: MVC

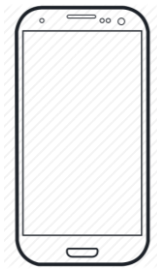
- Best practice in developing an Android application: use the Model View Controller (MVC) pattern
- MVC is composed by 3 components:



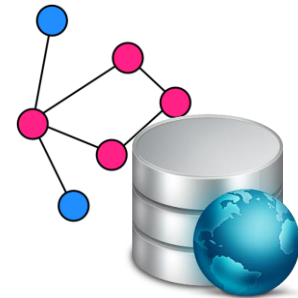
View

Android app development: MVC

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



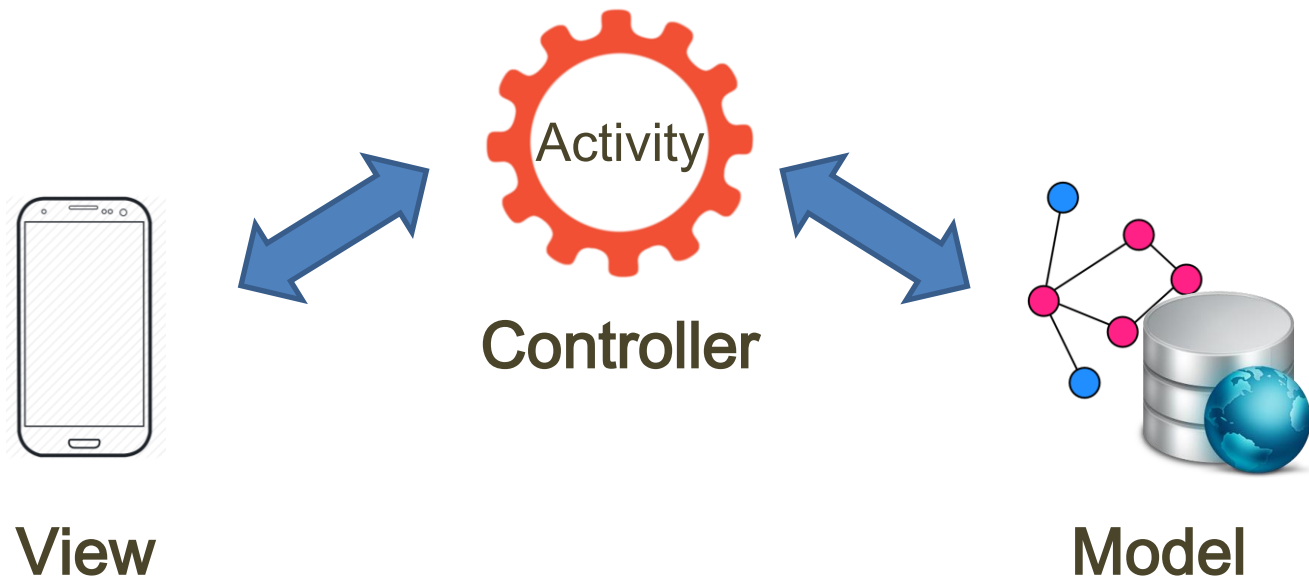
View



Model

Android app development: MVC

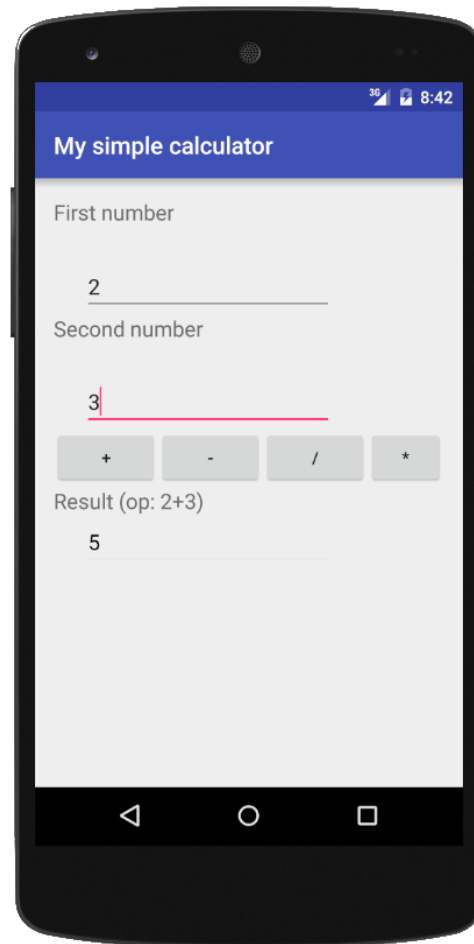
- Best practice in developing an Android application: use the Model View Controller (MVC) pattern
- MVC is composed by 3 components:



Android app development: Tools

- Needed tools
 - **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

Hands on: simple calculator



Android Studio: Create a new app

Create New Project

New Project
Android Studio

Configure your new project

Application name: HelloWorld

Company Domain: it.polito.elite

Package name: elite.polito.it.helloworld

Project location: C:\Users\Teo\Desktop\HelloWorld

Create New Project

Target Android Devices

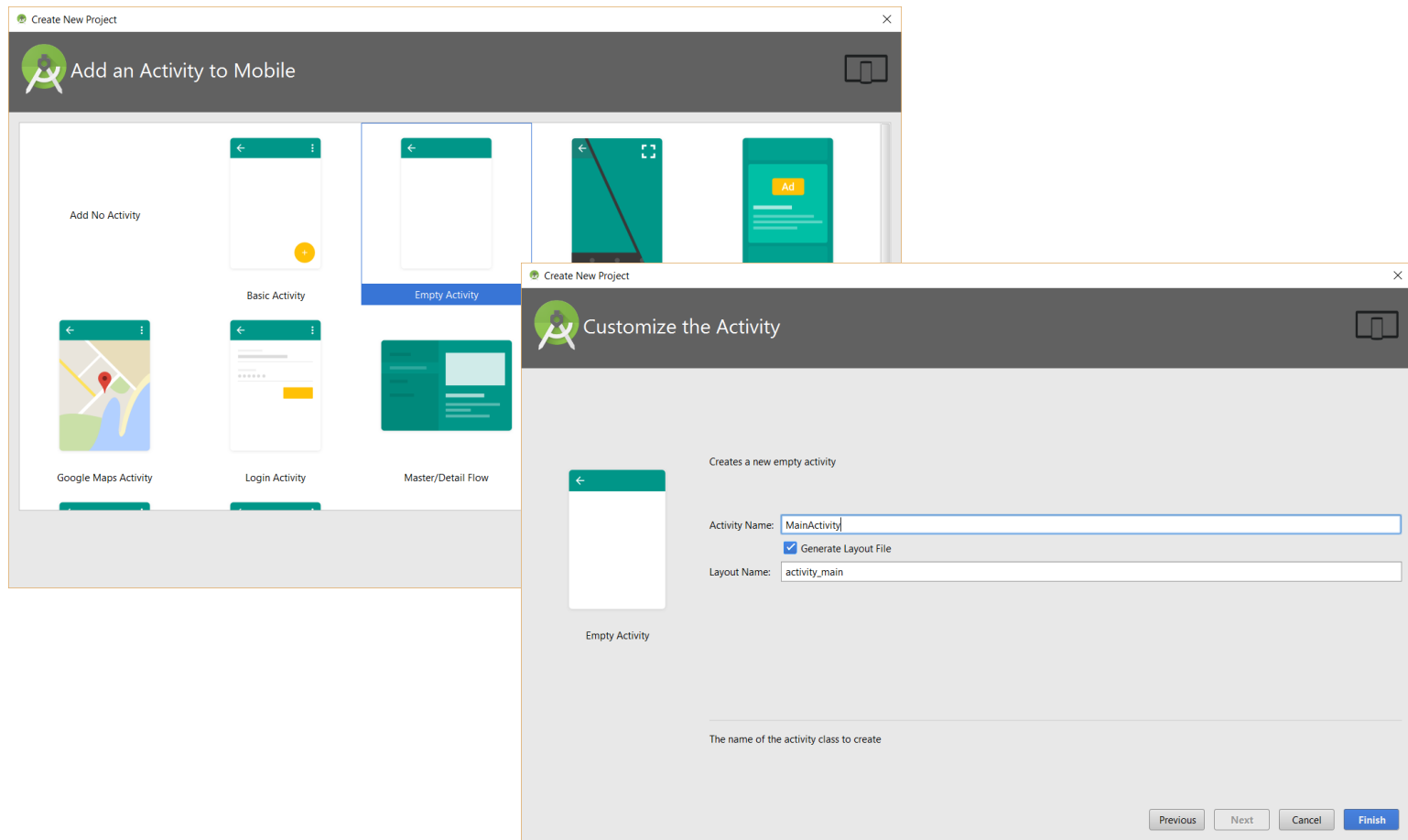
Select the form factors your app will run on

Different platforms may require separate SDKs

- Phone and Tablet
 - Minimum SDK: API 19: Android 4.4 (KitKat)
 - Lower API levels target more devices, but have fewer features available.
By targeting API 19 and later, your app will run on approximately **73.9%** 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

Android Studio: Create a new app



App structure

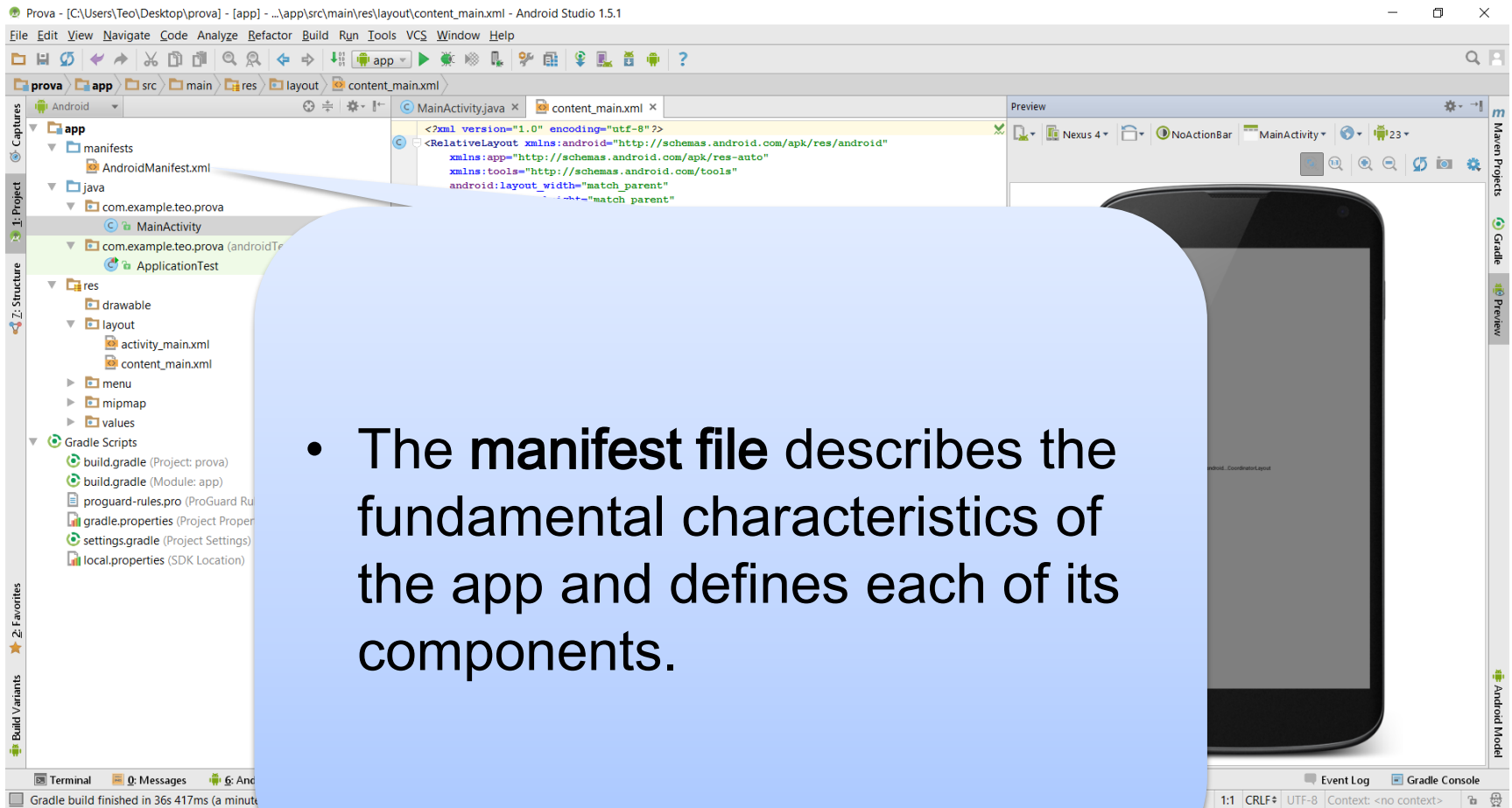
The screenshot displays the Android Studio 1.5.1 interface. The top menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The toolbar contains icons for file operations and development actions. The main workspace is divided into three panes:

- Left Pane (Project Structure):** Shows the project hierarchy for 'prova'. It includes 'manifests' (AndroidManifest.xml), 'java' (com.example.teo.prova with MainActivity and ApplicationTest), 'res' (drawable, layout with activity_main.xml and content_main.xml, menu, mipmap, values), and 'Gradle Scripts' (build.gradle, proguard-rules.pro, gradle.properties, settings.gradle, local.properties).
- Center Pane (Code Editor):** Displays the XML code for 'content_main.xml'. The code defines a RelativeLayout containing a single TextView with the text 'Hello World!'.

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout 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"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.teo.prova.MainActivity"
    tools:showIn="@layout/activity_main">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```
- Right Pane (Preview):** Shows a visual preview of the app on a Nexus 4 device. The screen displays a dark gray background with the text 'android.CoordinatorLayout' at the bottom.

The bottom status bar shows 'Gradle build finished in 36s 417ms (a minute ago)', 'Event Log', 'Gradle Console', and encoding information: '1:1 CRLF UTF-8 Context: <no context>'.

App structure



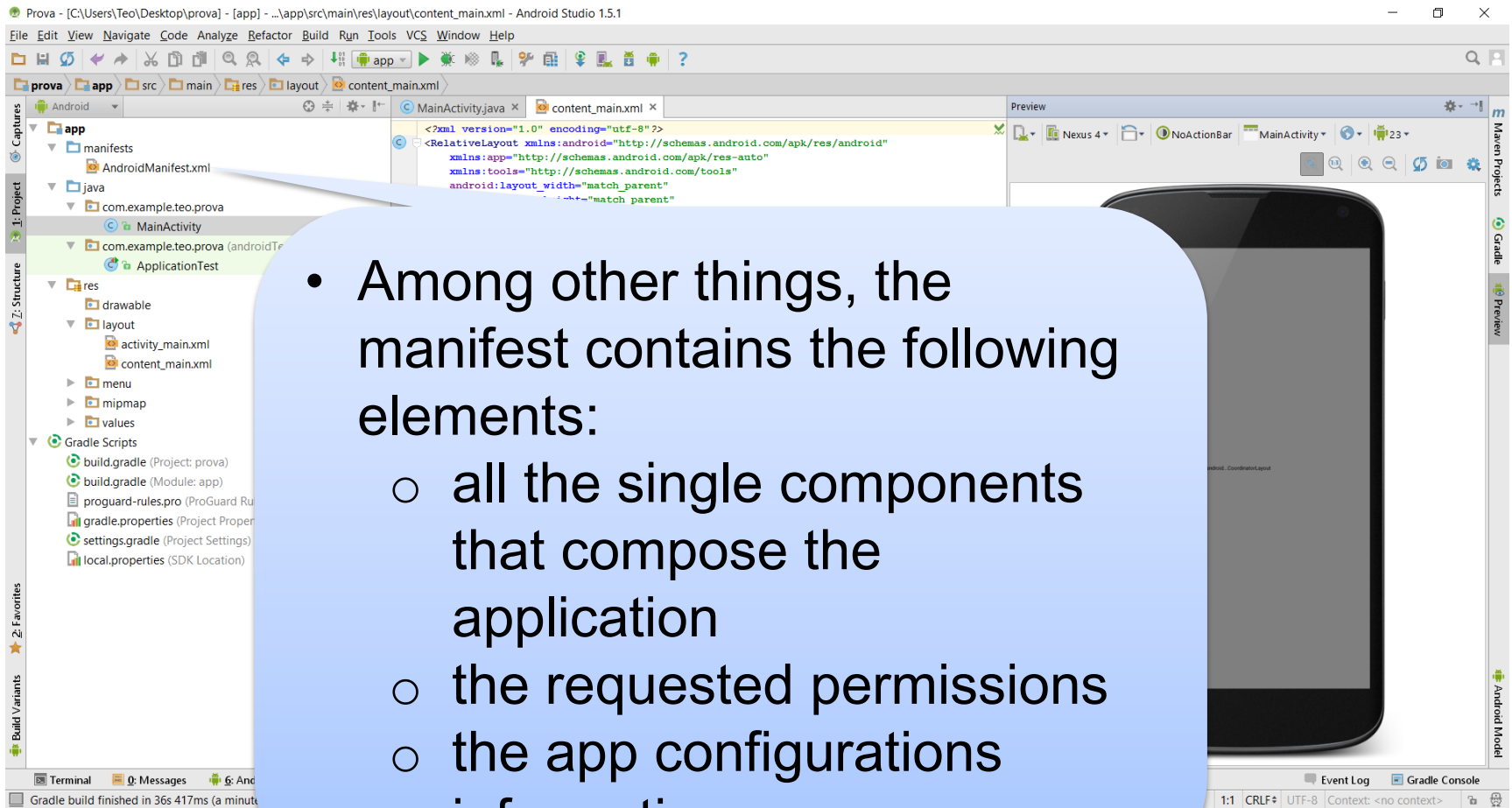
The screenshot displays the Android Studio interface. On the left, the 'Structure' pane shows the project hierarchy: 'prova' (Project) contains 'app' (Module), which includes 'manifests' (AndroidManifest.xml), 'java' (com.example.teo.prova), and 'res' (drawable, layout, menu, mipmap, values). The 'layout' folder is expanded, showing 'activity_main.xml' and 'content_main.xml'. The 'Code' editor shows the content of 'content_main.xml' with the following XML code:

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout 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">
```

The 'Preview' pane on the right shows a virtual device (Nexus 4) displaying the app's UI. A blue callout box points to the 'AndroidManifest.xml' file in the Structure pane.

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

App structure



The screenshot shows the Android Studio interface. The left sidebar displays the project structure for 'prova', including 'manifests', 'java', 'res', and 'Gradle Scripts'. The main editor shows the 'content_main.xml' file with the following XML code:

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout 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">
```

A blue callout box contains the following text:

- 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">

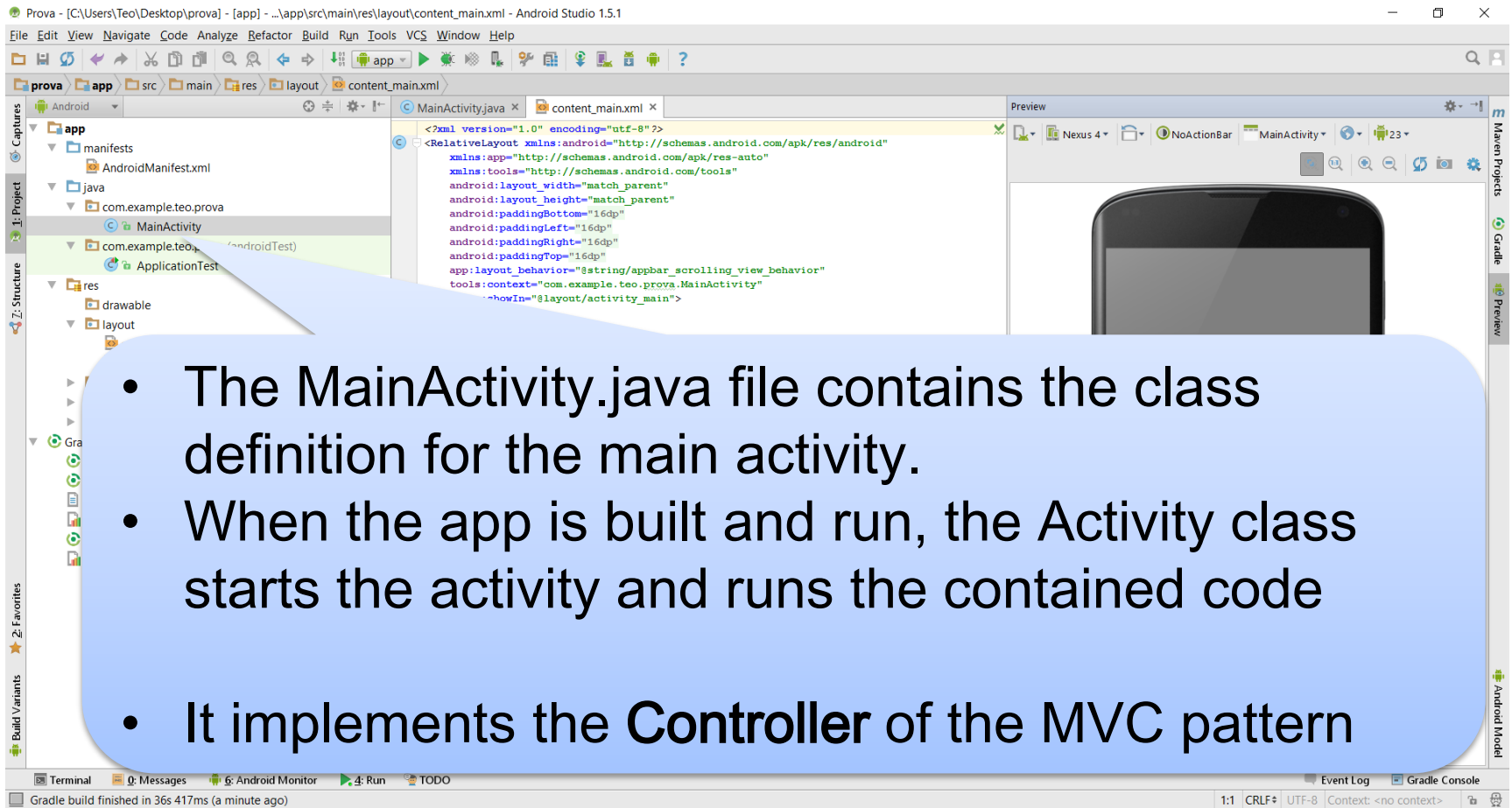
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <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 screenshot shows the Android Studio interface. On the left, the Project view displays the app structure: `app` (manifests, java, res), `com.example.teo.prova` (MainActivity), and `com.example.teo.prova (AndroidTest)` (ApplicationTest). The main editor shows the `content_main.xml` file with the following XML code:

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout 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"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.teo.prova.MainActivity"
    android:showIn="@layout/activity_main">
```

On the right, the Preview window shows a mobile device mockup. A blue callout box is overlaid on the image, containing the following list:

- 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

The bottom status bar shows "Gradle build finished in 36s 417ms (a minute ago)" and "Context: <no context>".

App structure

Example: MainActivity.java

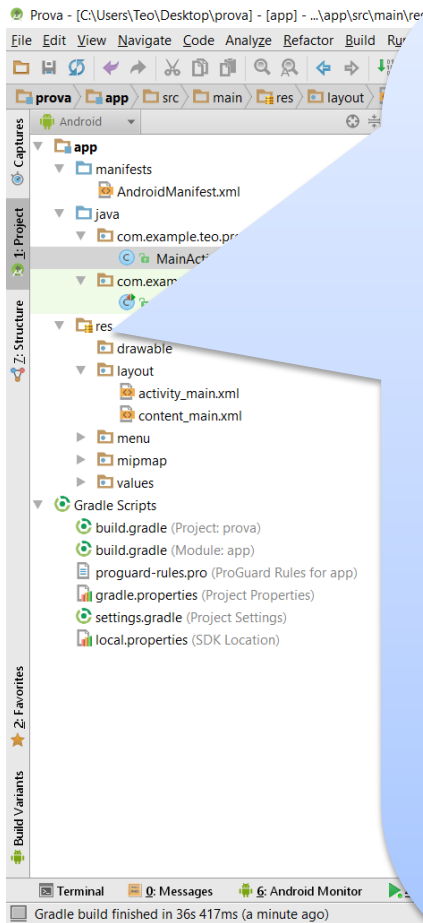
```
package elite.polito.it.prova;

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

public class MainActivity extends AppCompatActivity {

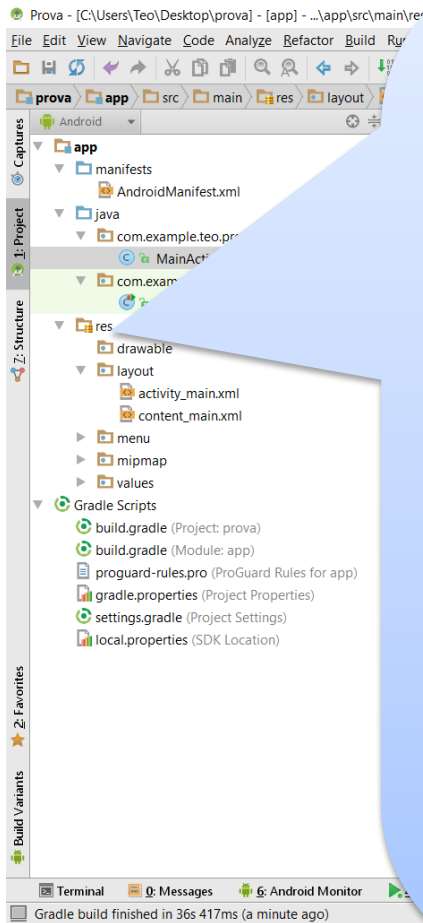
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

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_my.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

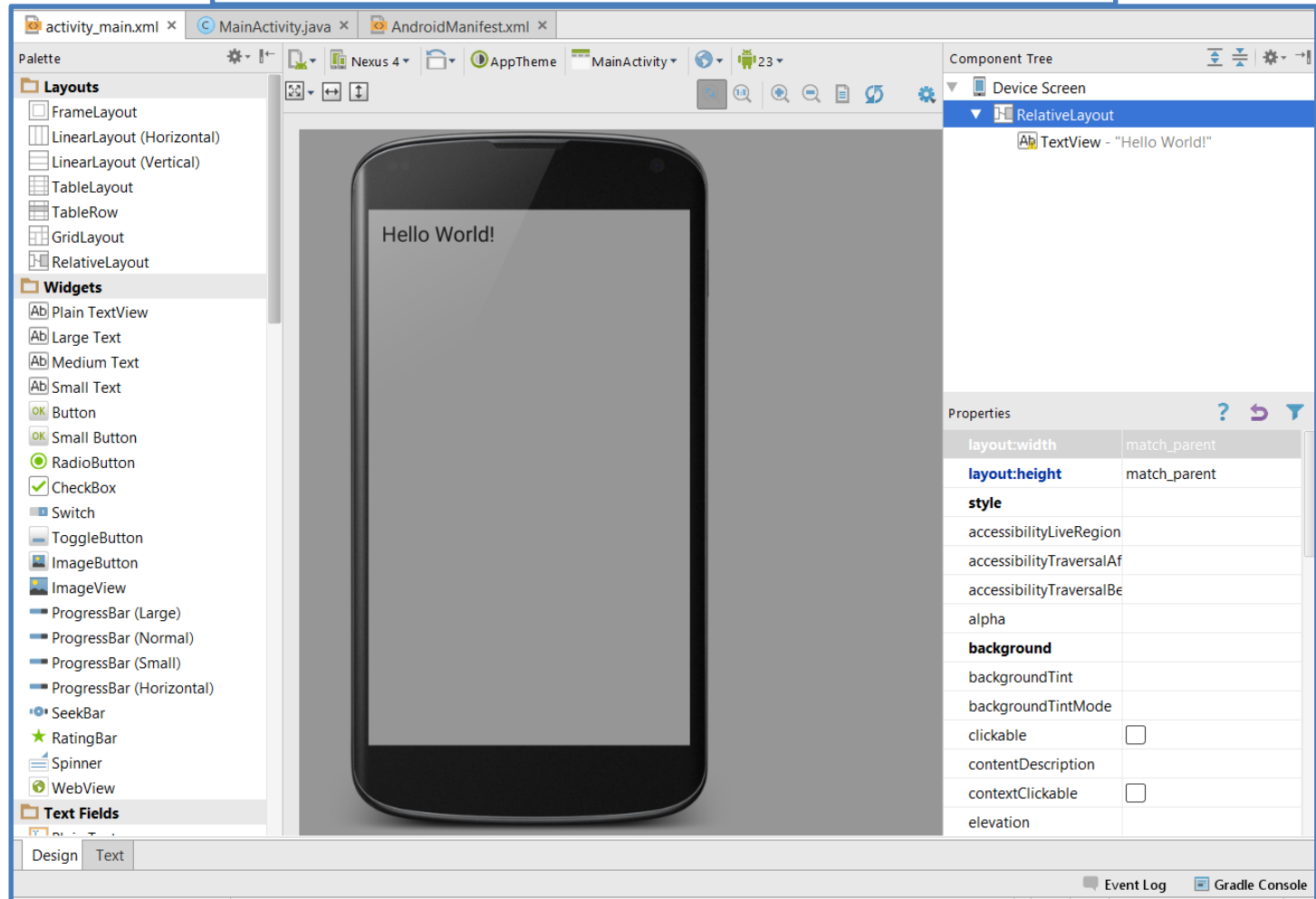
Example of activity_main.xml

```
<?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

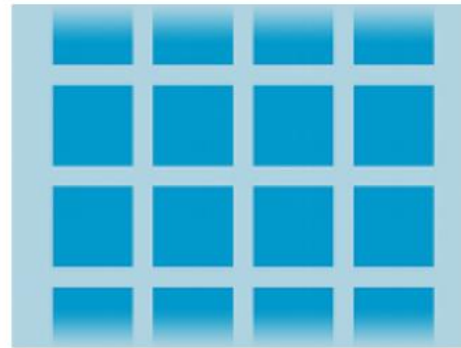
- When the content for your layout is dynamic or not pre-determined, you can use one of these 2 layouts.

List View



Displays a scrolling single column list.

Grid View

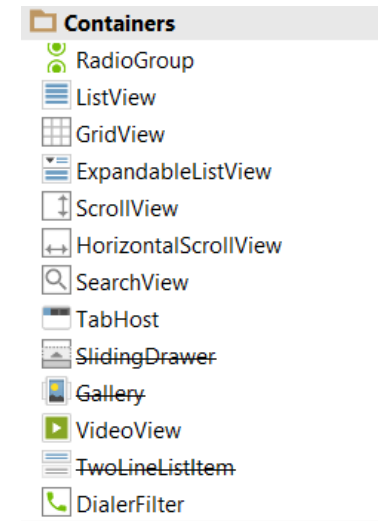
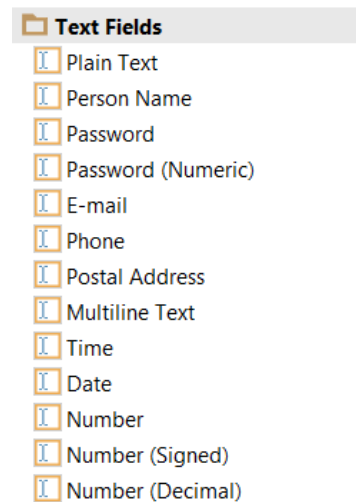
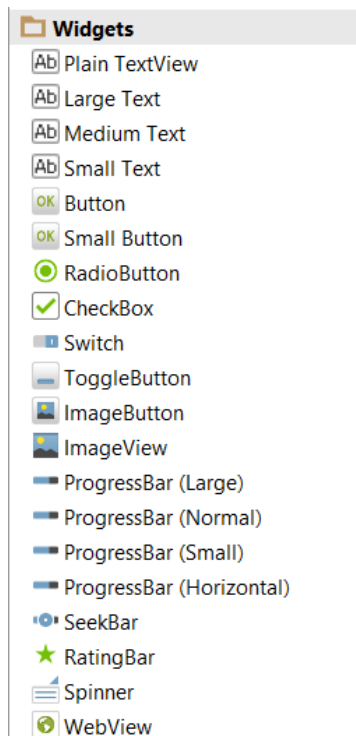


Displays a scrolling grid of columns and rows.

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

Widgets

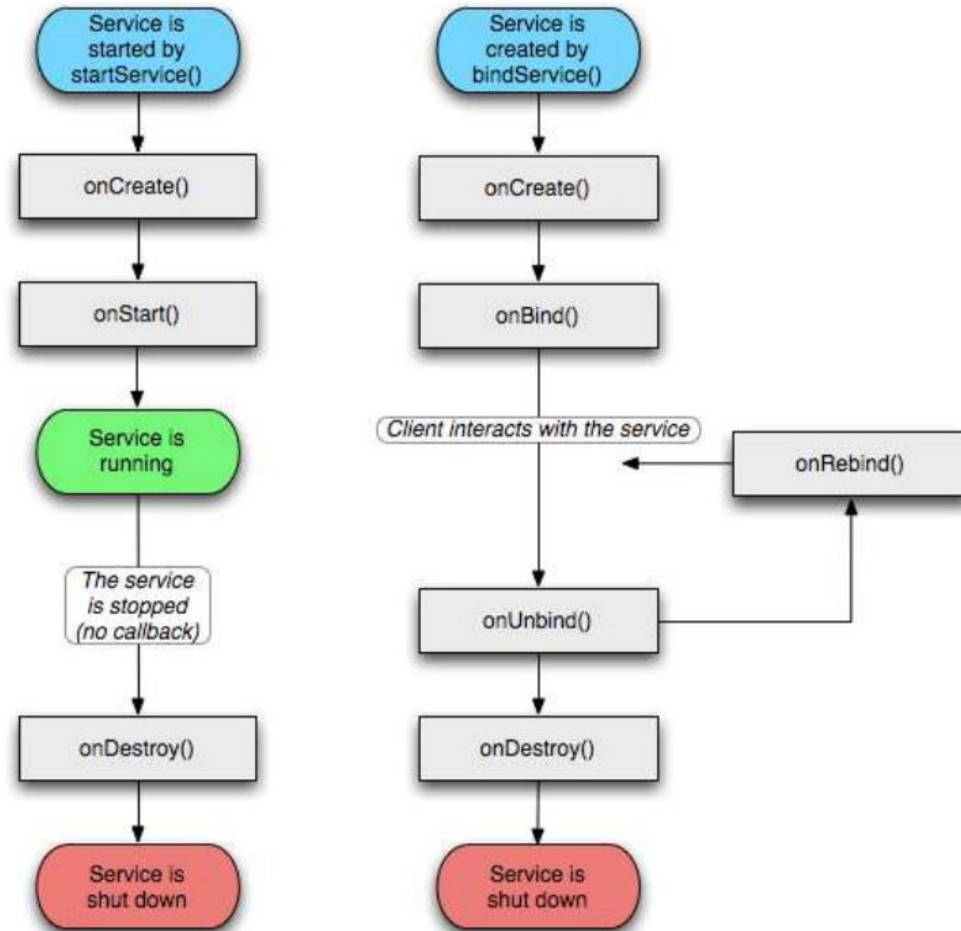
- Every widget (used to create interactive UI components (buttons, text fields, etc.)) is a subclass of the View class



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

Broadcast Receiver

- A Broadcast Receiver is a component which “waits” for messages that can be created by:
 - the Operating System (for example, when the Wi-Fi is turned on/off or when a picture was captured ...)
 - applications (for example, when a data transfer is completed)
- 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 Application process's 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

TodoList

- call Giovanni for Aml project organization
- buy a new mouse
- find a present for Angelina's birthday
- organize mega party (last week of April)
- book summer holidays
- whatsapp Mary for a coffee
- install Notification Collector from Play Store to help e-Lite research
- it works
- Pippo

INSERT A NEW TASK

TodoList

Details:

Description: book summer holidays

Urgent:

UPDATE DELETE

TodoList

Insert a new task

Description: _____

Urgent:

INSERT

Questions?

01PRD AMBIENT INTELLIGENCE: TECHNOLOGY AND DESIGN

Teodoro Montanaro

teodoro.montanaro@polito.it

