Introduction to Android

Ambient intelligence

Alberto Monge Roffarello
Politecnico di Torino, 2018/2019

Some slides and figures are taken from the
Mobile Application Development (MAD) course

POLITECNICO
DI TORINO

e-Lite

Disclaimer

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

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

ONLINE DOCUMENTATION:
https://developer.android.com/guide/

Summary

- Short history
- Platform
- Application Fundamentals
- Application Lifecycle
- Tools

Android app development

ANDROID HISTORY

POLITECNICO
DI TORINO

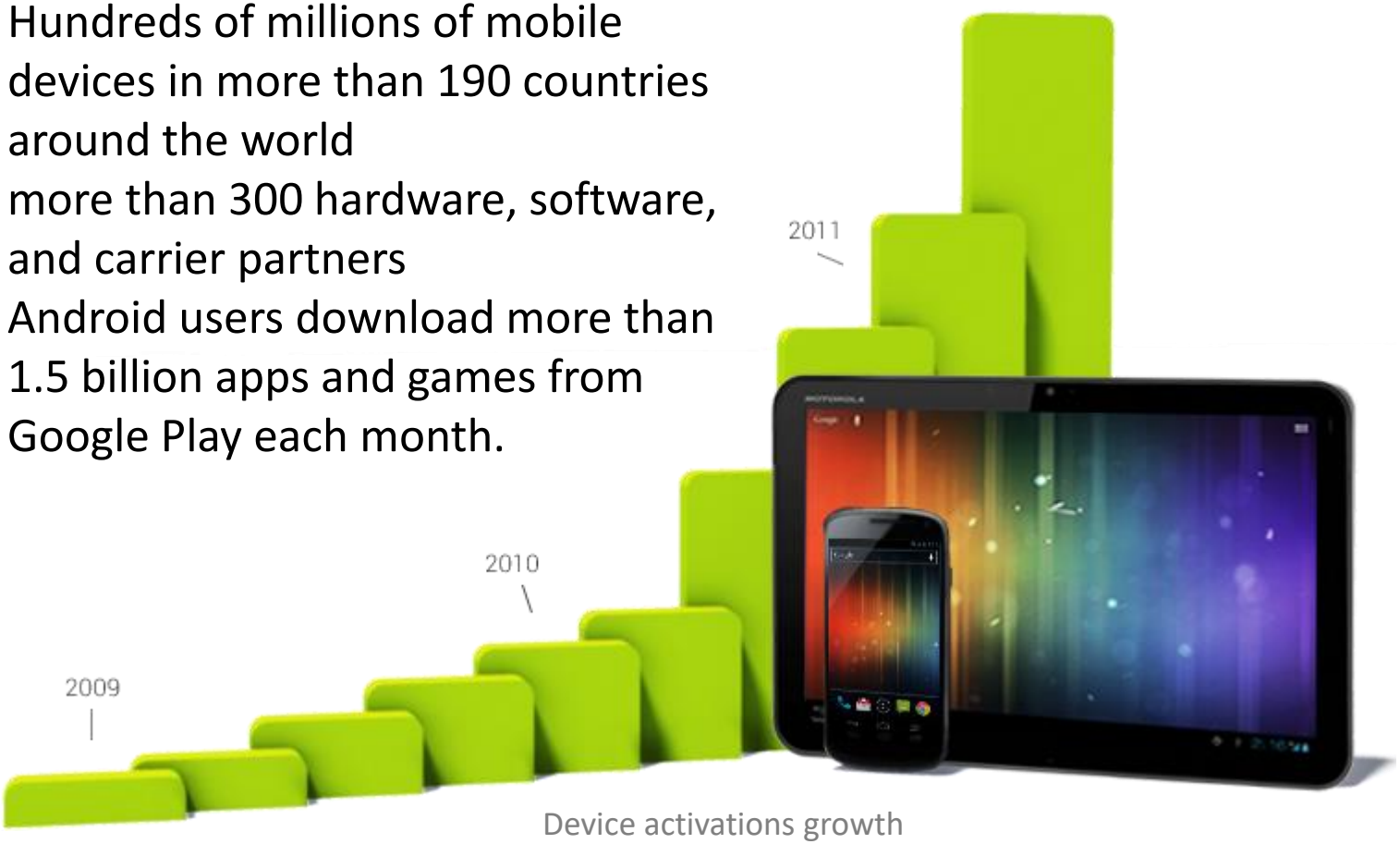
e-Lite

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

Market share

- 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

Versions

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%

https://developer.android.com/about/dashboards/index.html

Android app development

THE ANDROID PLATFORM

POLITECNICO
DI TORINO

e-Lite

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

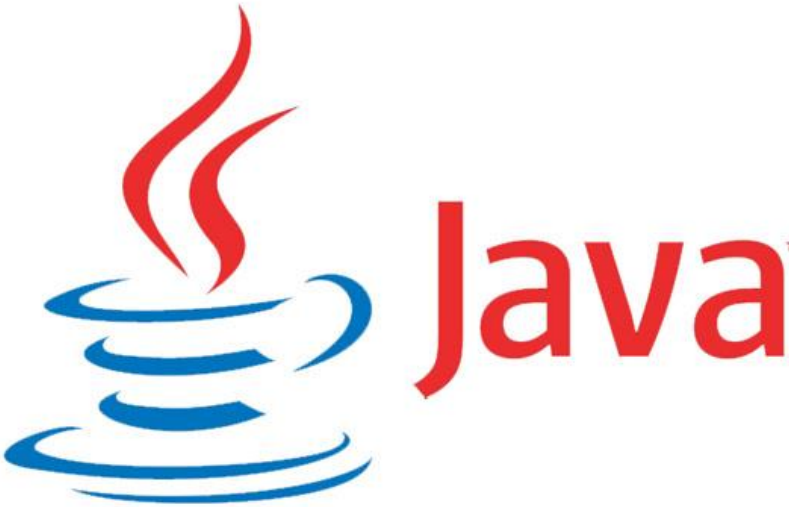
[1] *https://source.android.com/*

Android Platform

- Development Tools are free:
  - Android applications are (mostly) written in Java programming language (6 or higher)
  - Alternatively, a C++ API is available
- There is no distinction between native and third-party applications
  - All the applications use the same Software Development Kit (SDK)
  - All the applications can access the underlying hardware

Java

- General-purpose computer-programming language
  - Concurrent
  - Class-based
  - Object-oriented
  - Portable



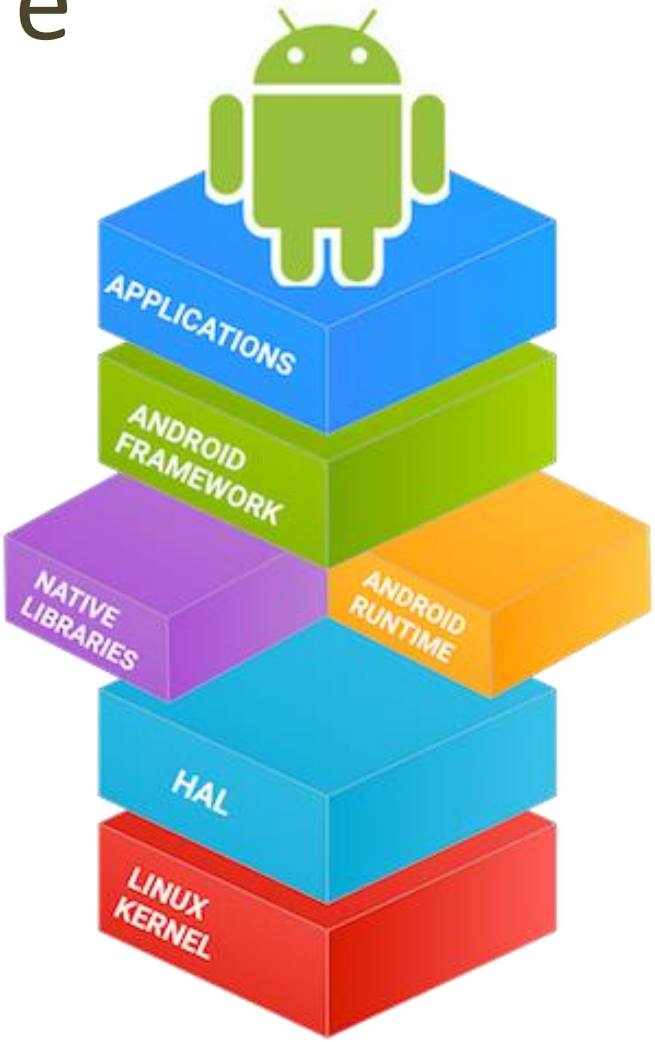
Java

- Java applications are typically compiled to bytecode that can run on any Java virtual machine (**JVM**) regardless of computer architecture
- Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (**VM**) written specifically for the host hardware.

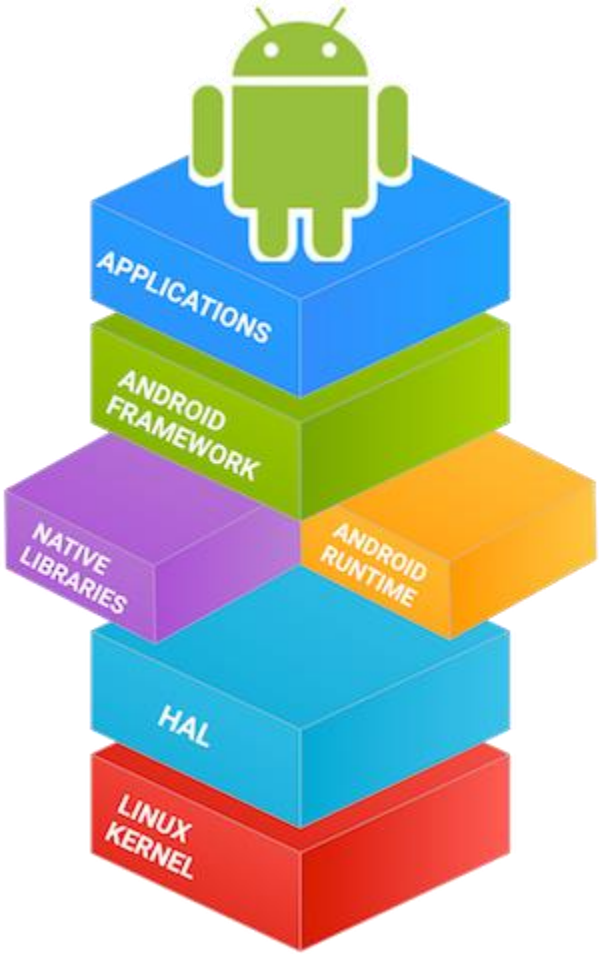
Android Architecture

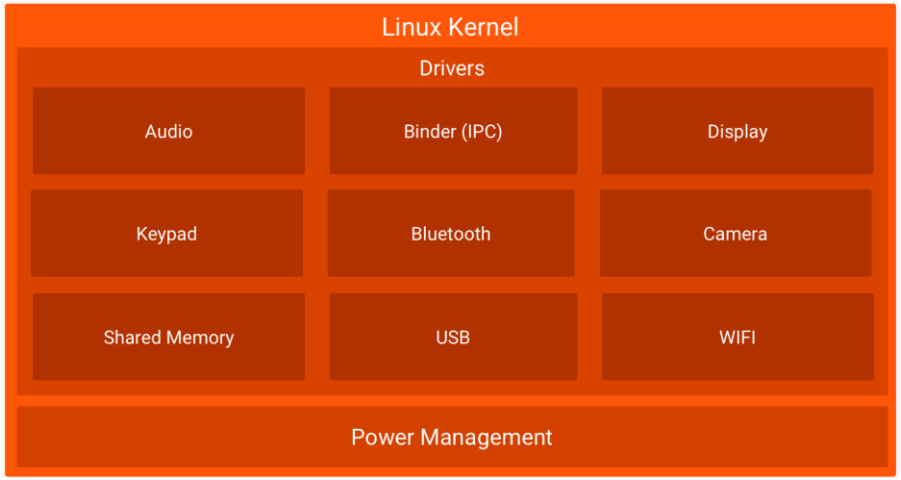
- Android is composed of an operating system and a software platform for creating apps and games
  - Android includes a set of minimal applications (browser, email client)
    - These basic features can be easily included in other applications
- Android has been designed to be robust
  - It is based on the Linux Operating System Kernel
  - Every Android application runs in its own process, with its own instance of the virtual machine

Architecture



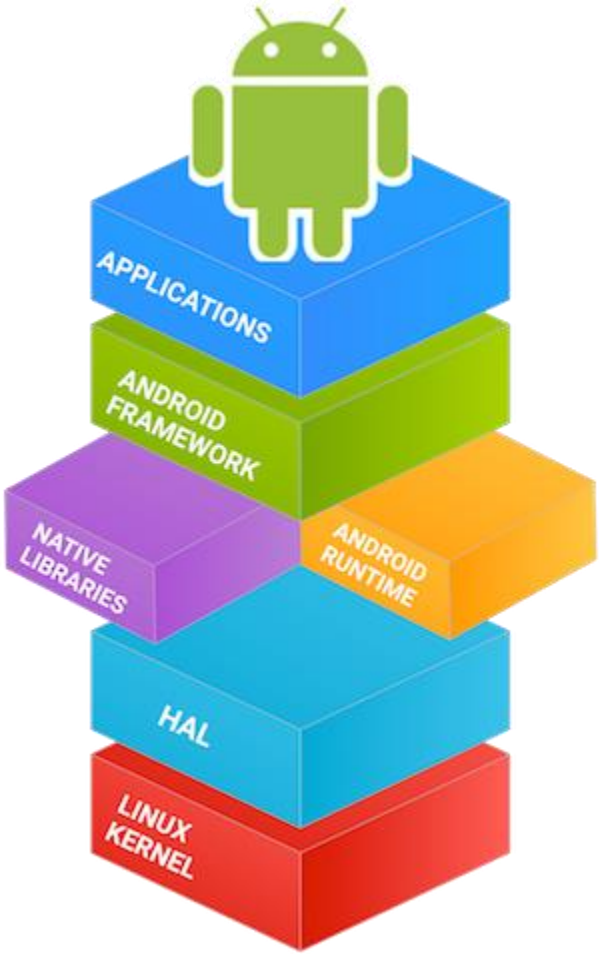
Architecture

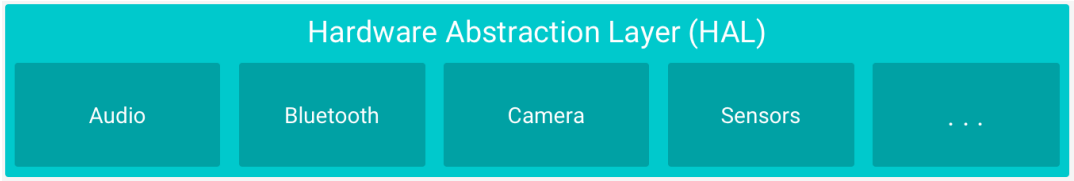




- Android is based on the Linux Kernel:
  - takes advantage of the Linux Kernel key security features
  - allows device manufacturers to develop hardware drivers for a well-known kernel

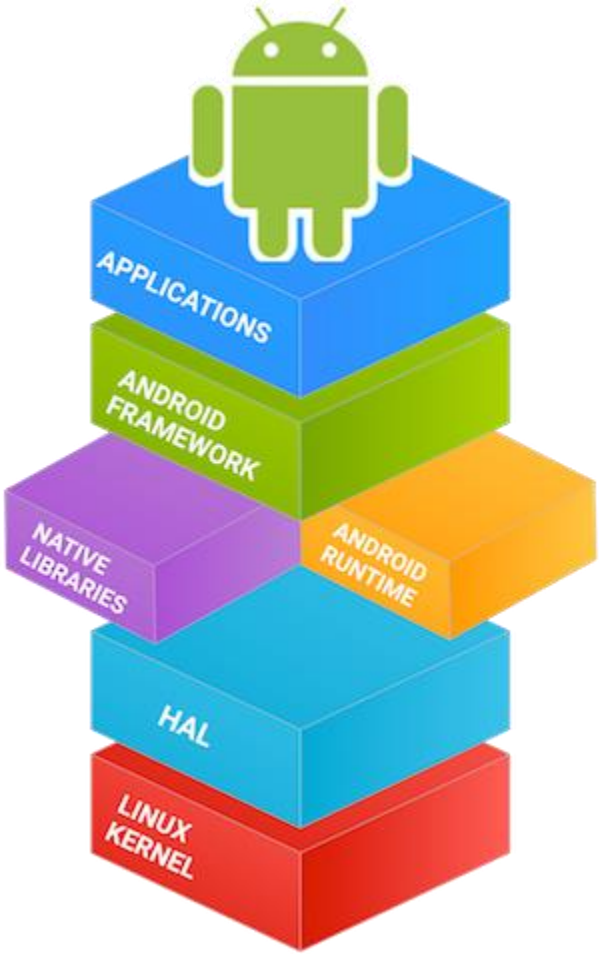
Architecture

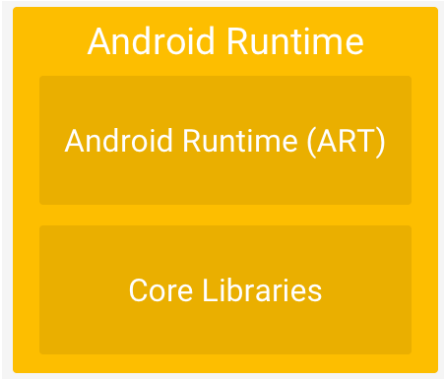




- Provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.
- It consists of multiple library modules that implement interfaces for specific type of hardware components (e.g., camera, Bluetooth …)

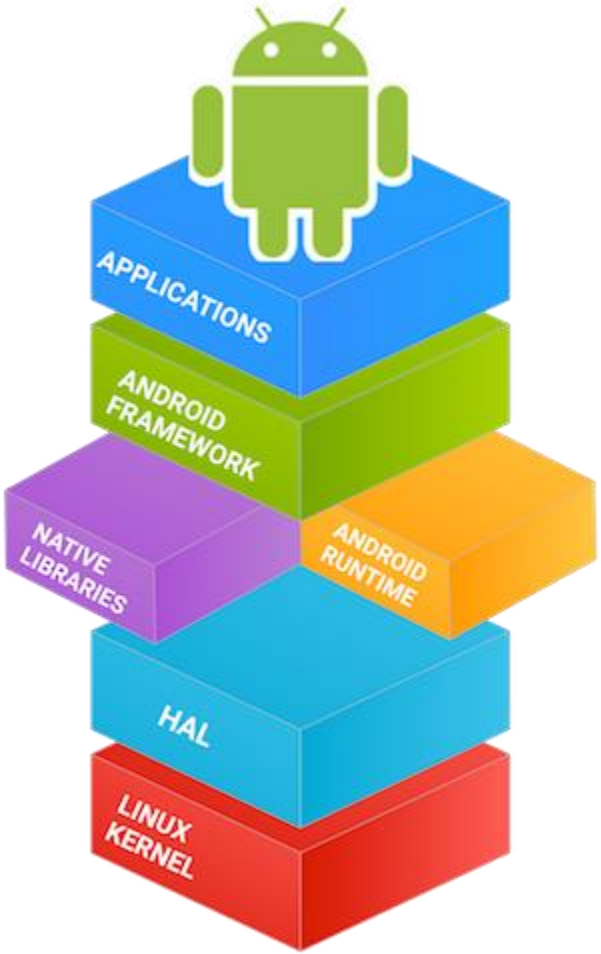
Architecture

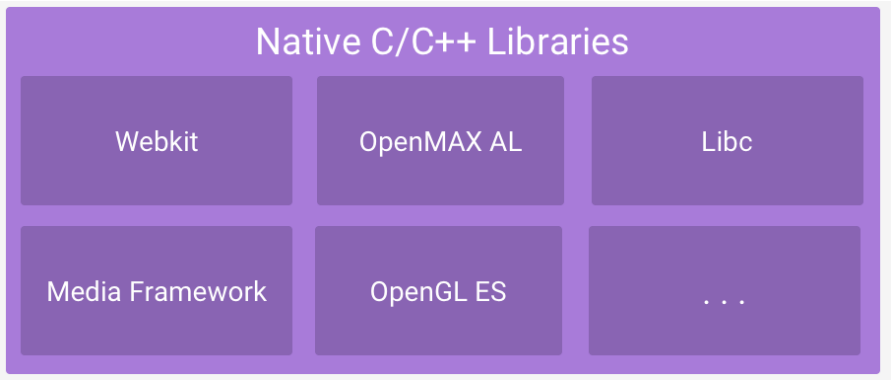




- ART is an application runtime environment (prior to Android 5.0, Dalvik used instead of ART)
- It is written to run multiple virtual machines, one for each running application
- Each app runs in its own process within its own instance of the Android Runtime (ART)

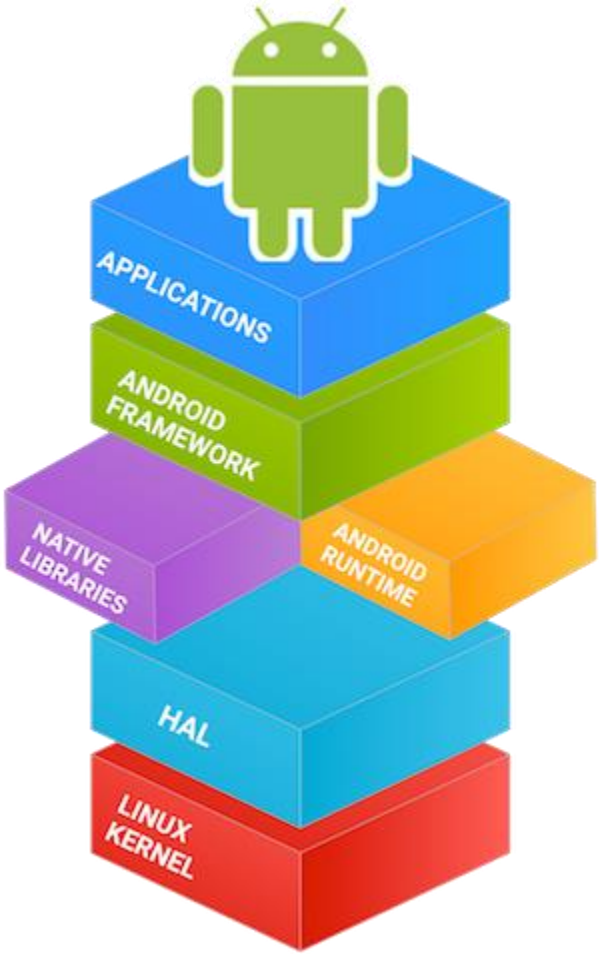
Architecture

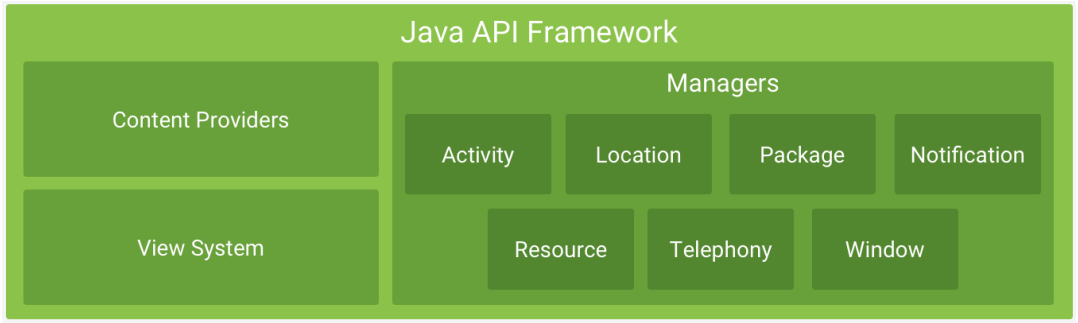




- Many core Android system components and services, (e.g., ART and HAL), are built from native code that require native libraries written in C and C++
- If you want to develop your app using C or C++, you can use the Android NDK

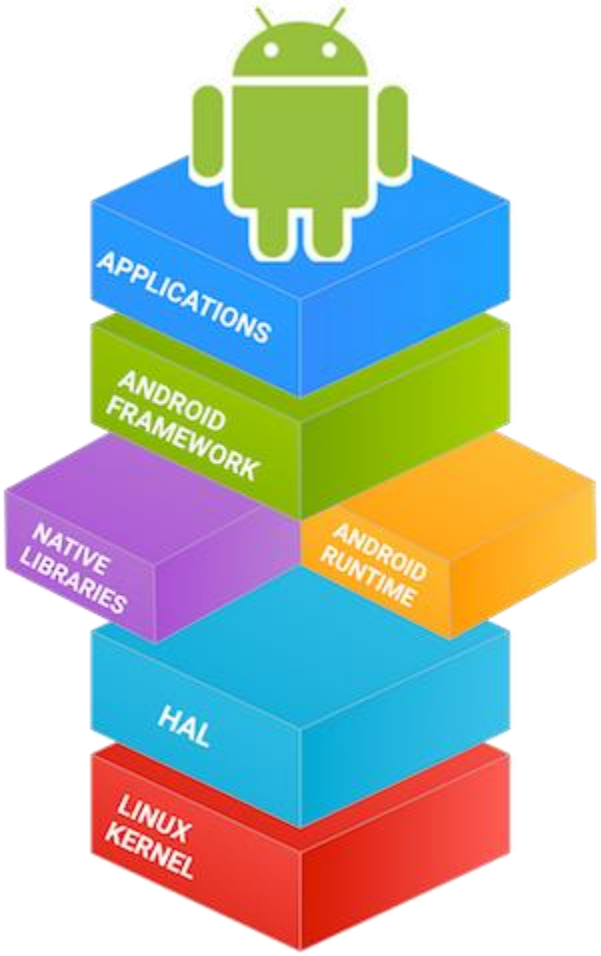
Architecture

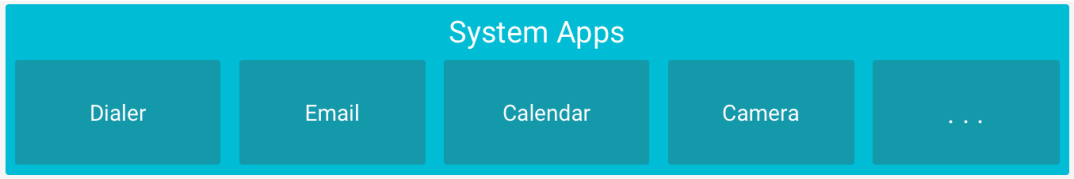




- The entire feature-set of the Android OS is available through Java APIs
- These APIs form the building blocks needed to create Android apps

Architecture





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

Security

- Every application runs with its own user
  - Once the application is installed, the operating system creates a new user profile associated with it
  - Filesystem permissions ensure that one user cannot alter or read another user's files
- Every application must declare which shared resources will use
  - For example, making phone calls, using the camera or other sensors
  - Android will block applications which try to use not declared resources
- Every application also requires the permission to access the user's private data
  - Such as preferences, user location, user contacts, …
  - If the permission is not granted, the installation fails

Android app development

APPLICATION FUNDAMENTALS

POLITECNICO
DI TORINO

e-Lite

RECAP

- The Android SDK exposes a set of APIs, which allows the access to the underlying hardware
  - No distinction between "native applications" and "third-party applications"
  - Every application, if equipped with the appropriate permissions, can use them
- Android includes a set of minimal applications such as a browser, and an email client
  - Third-party provided applications can integrate, extend or even replace them
- The main programming language is Java
  - But it is possible to develop applications using C++, as well

Application Structure

- Conceptually, an application consists of a set of data and code designed to perform a given set of tasks

- Android applications do not have a single entry point, as it happens in other operating systems
  - Each application consists of one or more components, activated by the operating system, at its own will

Application Structure

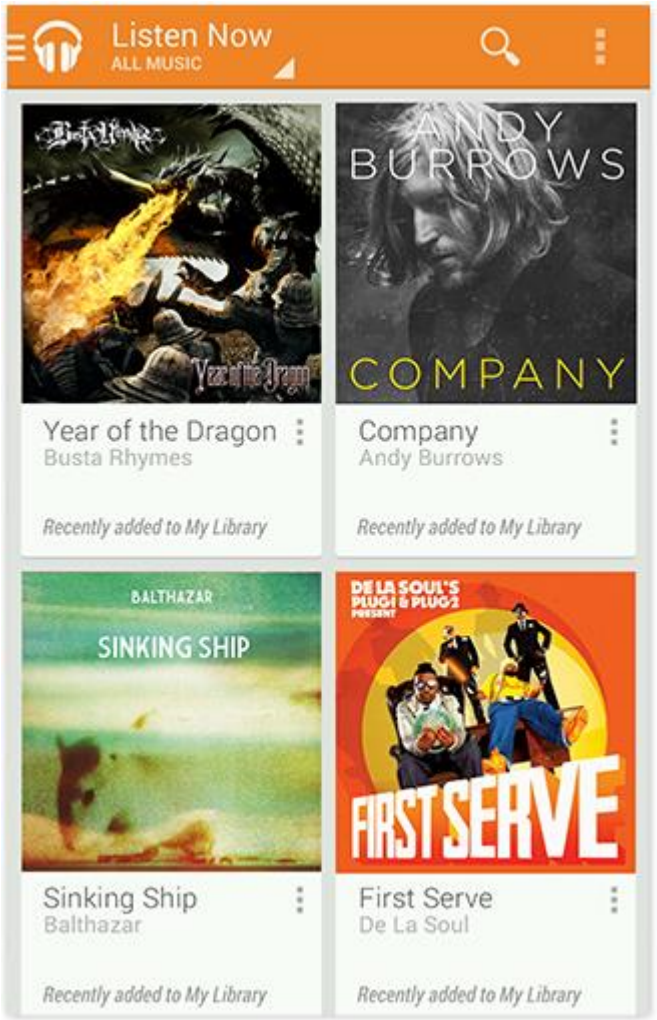
- Each component takes care of a specific interaction with the operating system and/or the user
  - Component creation, operation, and destruction follow a well defined life-cycle

Application Structure

- Each application consists of one or more of the following components:
  - Activity
  - Service
  - Content Provider
  - Broadcast Receiver

Activity

- An activity is a software component that:
  - Has a Graphical User Interface
  - Can perform a task inside the application
- An application is composed by one or more activities. 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.



Service

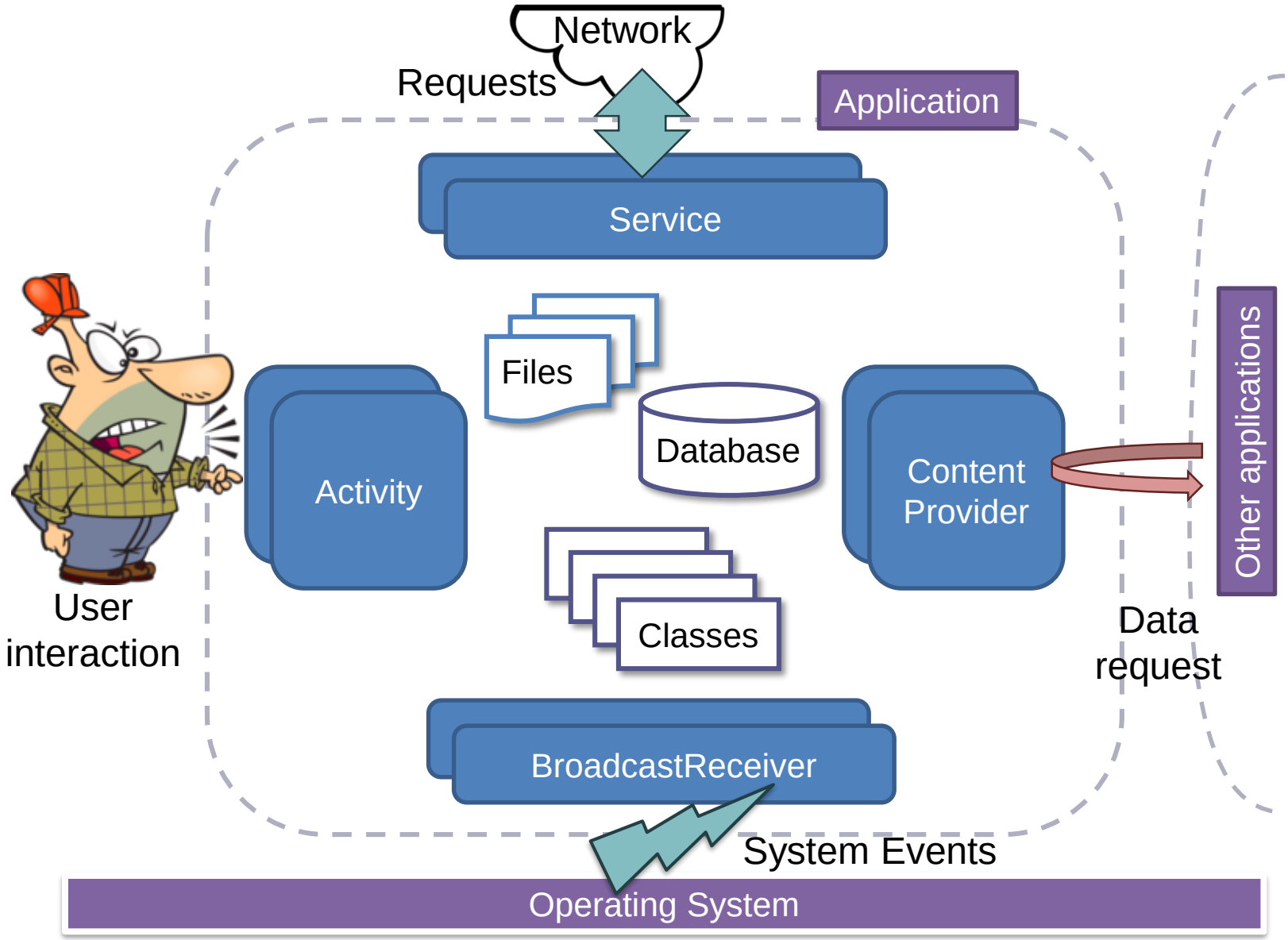
- A service is a component that can run in the background
  - It does not provide a user interface
- Usually services are used to perform long tasks
  - A service could play music while the user is using another application
  - A service could gather network data without blocking the user interaction with another activity

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

- A Broadcast Receiver is a component which “waits” for messages
  - Some messages are created by the Operating System
    - For example, whenever the display is turned off, when the battery is low …
  - Applications can produce messages, too
    - For example, when a data transfer is completed
- A broadcast receiver does not have a Graphical User Interface, but it can generate notifications in the status bar
  - To notify the user that a particular message is detected


Network
Requests
Application
Service
Files
Database
Activity
Content Provider
Other applications
User interaction
Classes
Data request
BroadcastReceiver
System Events
Operating System

Android app development

APPLICATION LIFECYCLE

POLITECNICO
DI TORINO

e-Lite

Application Lifecycle

- The functionality provided by an application are defined by its **manifest file**
  - It is an XML document that “signs a sort of contract” between the application and the execution environment
  - It lists all the single components that compose the application, the requested permissions and their configurations information

Application Lifecycle

- When an external event occurs, based on its type and on the components declared in the manifest file, Android creates a new process
  - Its owner is the one that was created when the application was installed
- For each application in execution, Android instantiates in its process a single object of class android.app.Application
  - It is possible to specify a subclass of it in the manifest file
  - This object can be used to store global information shared by all the app components

Application Lifecycle

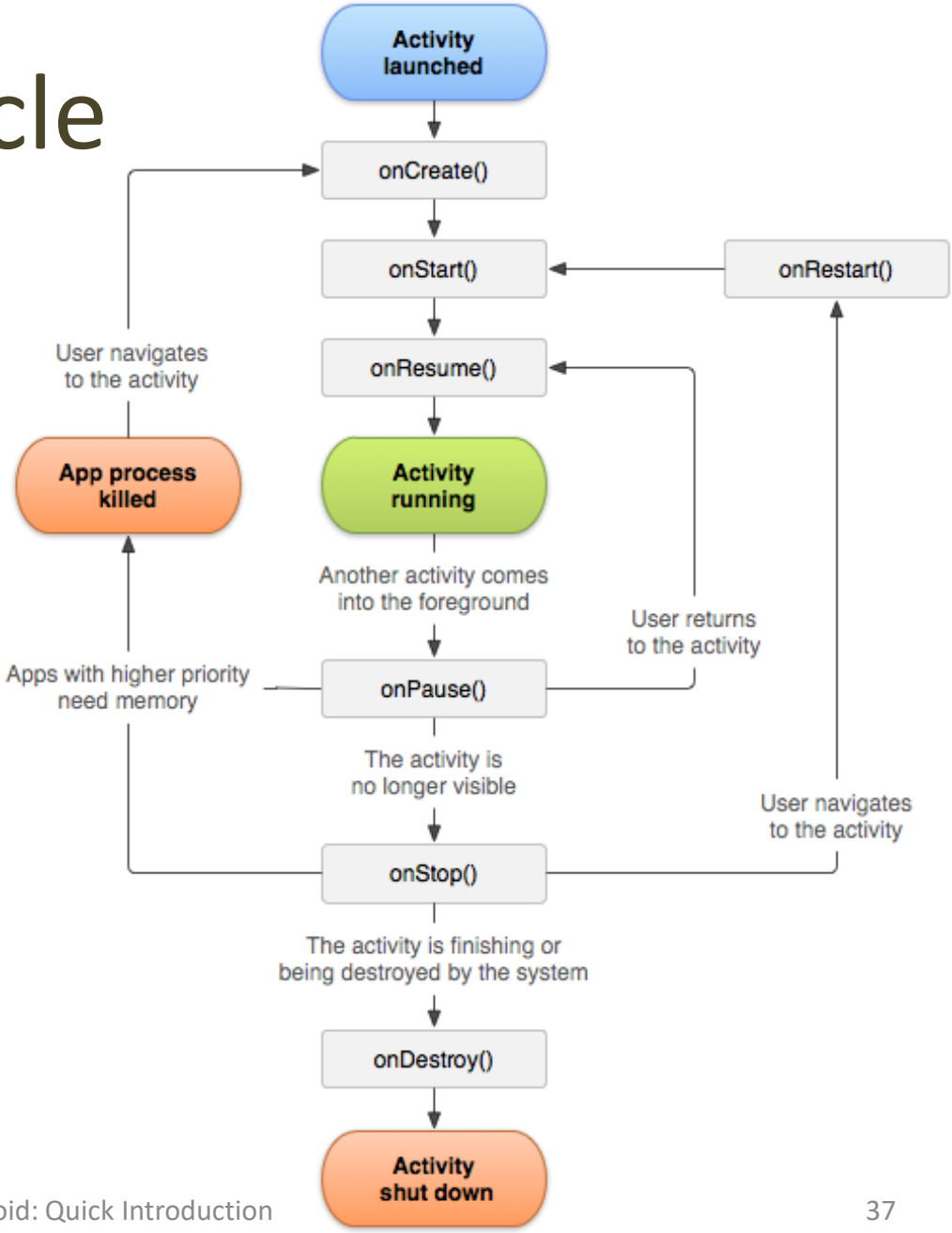
- Android notifies the application object with the evolving status of the ongoing elaboration
  - void onCreate()
  - void onConfigurationChanged(…)
  - void onLowMemory()
  - void onTerminate()

Application Lifecycle

- Once the application object has been created and initially notified of the beginning of the process, Android instantiates the **main** activity
- The activity receives some initial events:
  - void onCreate()
  - void onStart()
  - void onResume()
- The application object stays in memory as long as there are active components
  - The application object is removed from the memory when all the components end their lifecycle

Activity - Lifecycle

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



Intents

- To instantiate components, Android uses **intents**
- An intent defines an action to be performed and a set of data on which to operate
  - The operating system finds and instantiates the corresponding components that can handle the required action
- Intents can be **implicit** or **explicit**

Implicit Intents

- They consist of several parts, the most important of which are
  - The **action**, a unique string describing what is requested or what has happened
  - The **data** to operate upon, typically expressed as a URI
  - The **category**, one or more strings containing additional information about the kind of component that should handle the intent

Implicit Intents and Manifest File

- All the components exported by an application are listed in its manifest file
  - Together with zero or more intent-filters
- Each filter describes a capability of the component, a set of intents that the component is willing to receive
  - Listing fields corresponding to the action, data, and category fields of an Intent object

Implicit Intents and Manifest File

- When an intent is delivered, Android tries to match it against all filters, in order to detect which component should be activated
  - Filters are also used to learn something about the component itself: the launcher is populated with all activities that have filters reporting action MAIN and category LAUNCHER

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mycompany.myapplication"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/title_main_activity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER”/>
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Explicit Intents

- An explicit intent is one that you use to launch a specific app component, such as a particular activity or service
- You can create and send to Android explicit intents from your code
- Typically, you have to specify the context of your app, and the Java class of the component you are interested in

Android app development

DEVELOPING FOR ANDROID: TOOLS

POLITECNICO
DI TORINO

e-Lite

Environment setup

- The most convenient tool for developers today is Android Studio
  - http://developer.android.com/sdk/index.html
- Android Studio offers
  - A rich code editor
  - Several code templates and integration with GitHub
  - Instant preview for many different devices
  - Dependency support and build automation via Gradle

Environment Setup

- Beside an IDE, the Android SDK need to be installed
  - Automatically done by the Android Studio installer
- Android SDK consists of a bunch of programs broadly divided into SDK tools and platform tools

SDK Tools

- Set of tools for debugging and testing, and other utilities that are required to develop an app
  - Installed in folder <sdk>/tools
- The most relevant are:
  - The emulator, that need some configuration before being run
  - The Android Debug Monitor, that provides debugging and profiling support for both emulators and real devices

Deploying Apps on Phones

- To install applications on your phone through USB cable, the “Debug mode” must be enabled
  - You need to activate the Developer Options
  - Different phones have different ways to activate the Developer Options

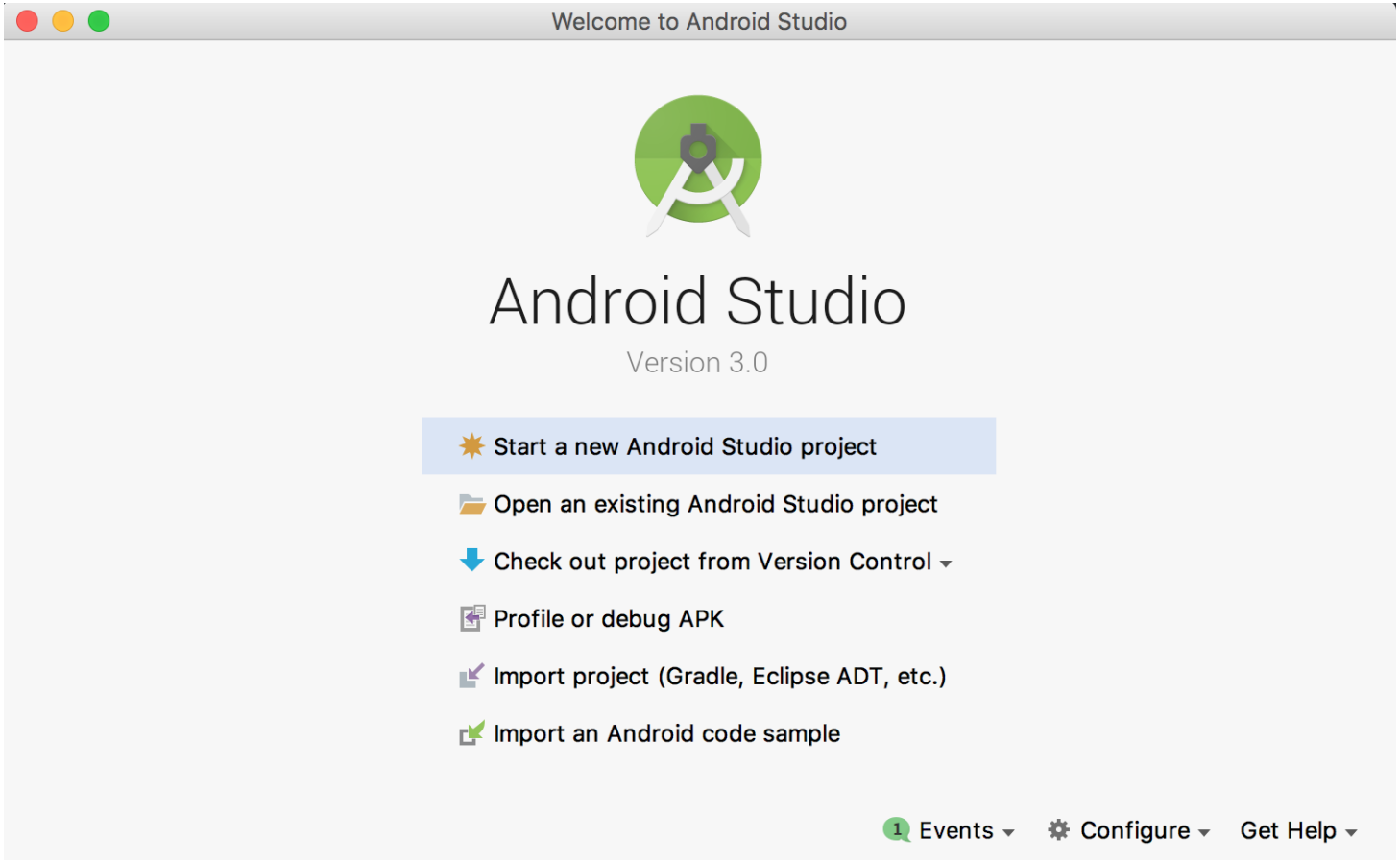
Android app development

PROJECT SETUP

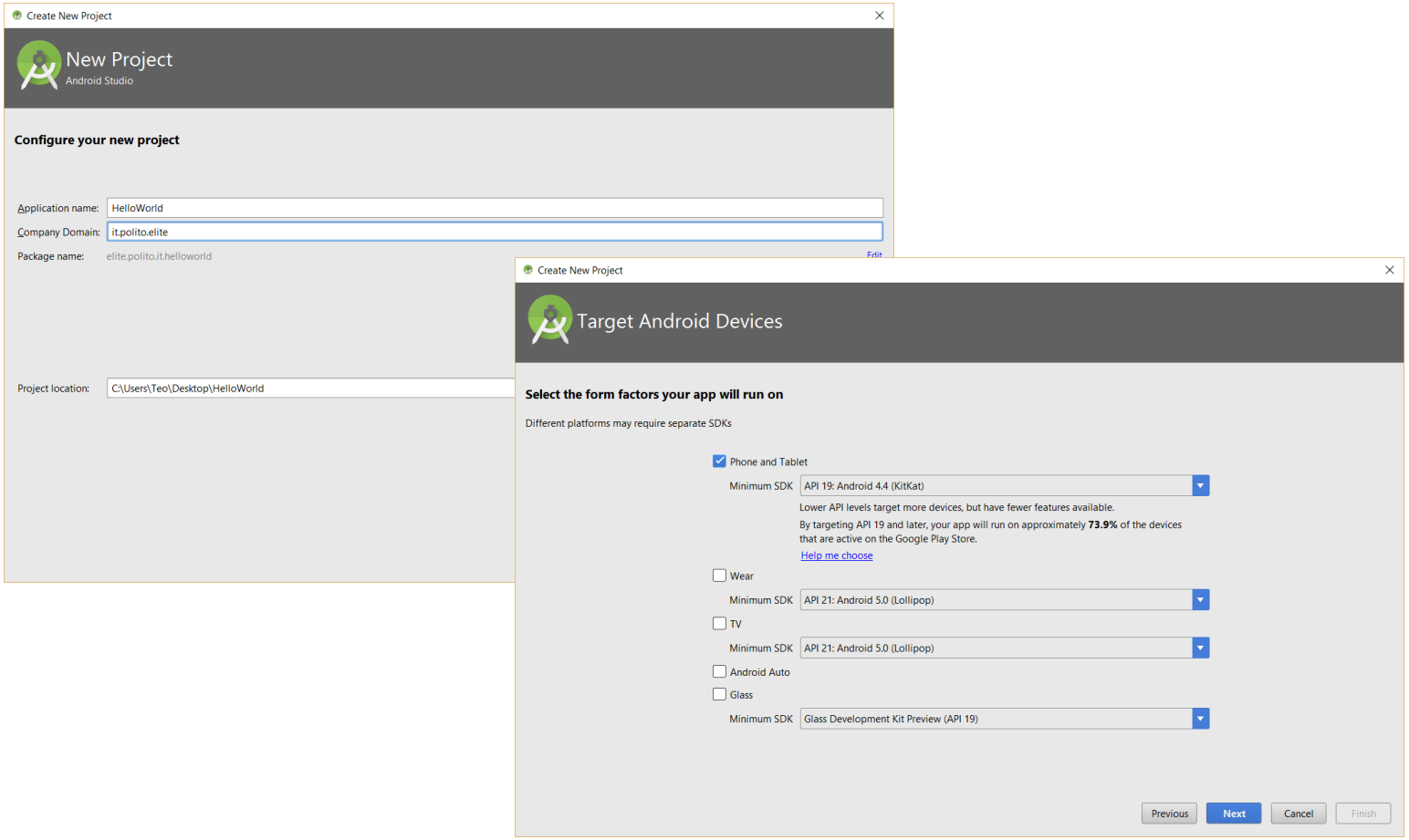
POLITECNICO
DI TORINO

e-Lite

Using Android Studio



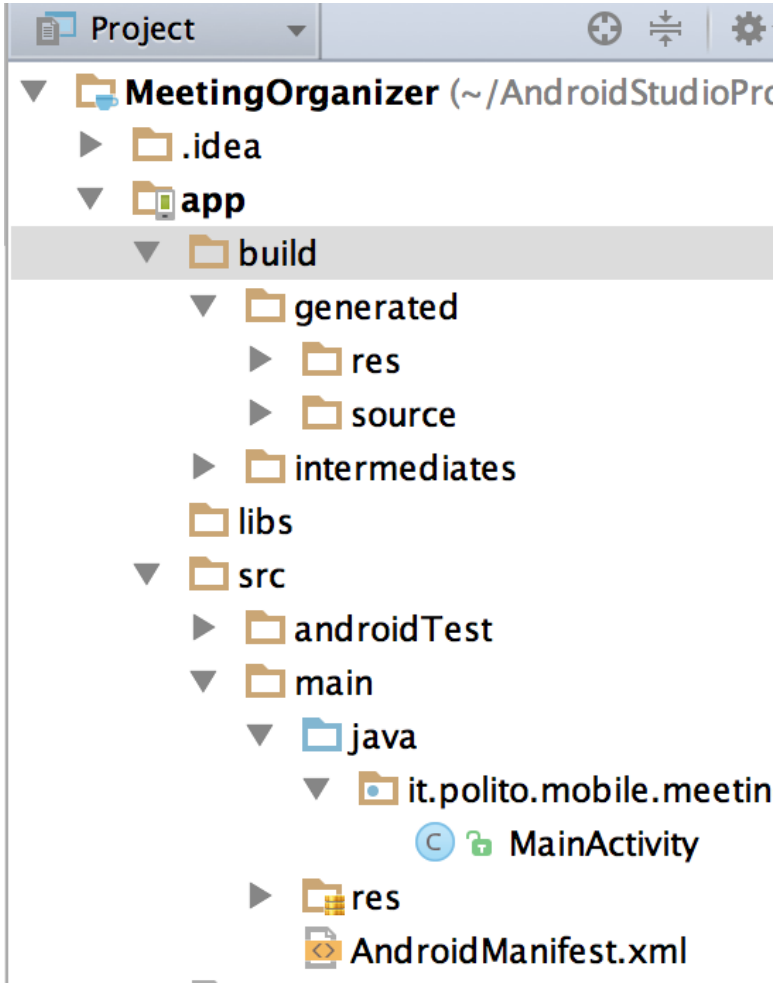
Using Android Studio

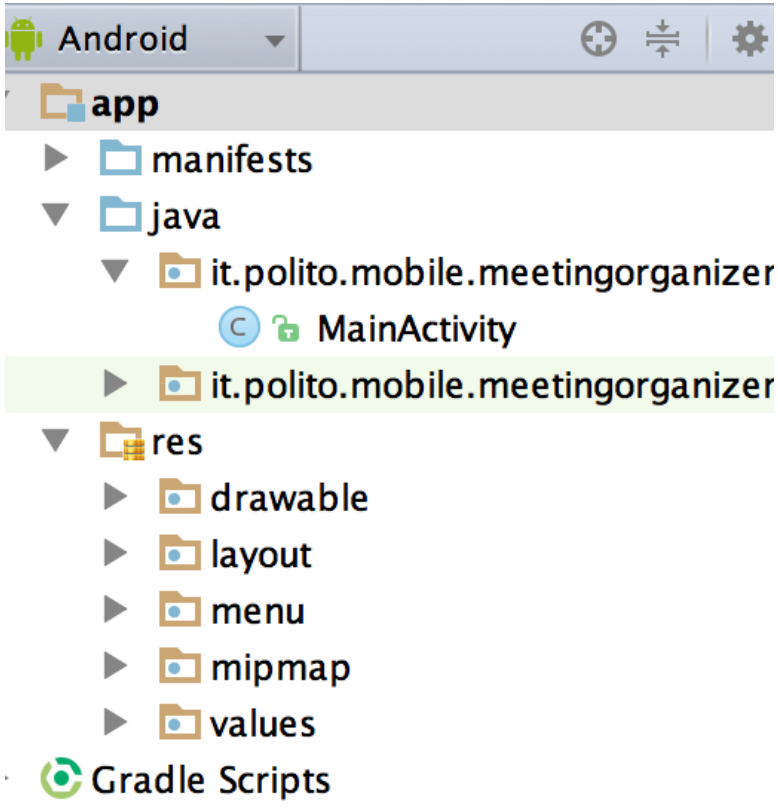


Project Structure

- Android Studio provides several alternative views for the project structure
  - The “Android” view shows a flattened version emphasizing source files
  - The “Project” view provides a more detailed vision of the folder structure, showing generated files

Project Structure





Source Files

- Source files are split into manifest, java, and resource files
  - A Manifest file describes the features, permissions and software components of the application
  - Java files are organized in packages and sub-packages according to the programmer's will
  - Resources are non-executable contents needed at program run-time (images, layout, values, …)

Setting up a virtual device

- To emulate the execution of an app, an Android Virtual Device (AVD) should be configured and run
  - Configuration provides information about the Android OS version, the device hardware capabilities and screen configuration, the size of an external SDCard, …
  - Common practice is to create several AVDs with different configuration to test various execution environments

Questions?

01QZP Ambient Intelligence

Alberto Monge Roffarello
alberto.monge@polito.it

License