Android: introduction

Ferruccio Damiani

Università di Torino
www.di.unito.it/~damiani

Mobile Device Programming
(Laurea Magistrale in Informatica, a.a. 2018-2019)

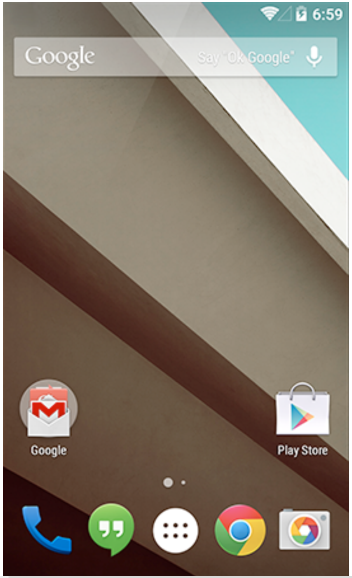
Outline

Outline

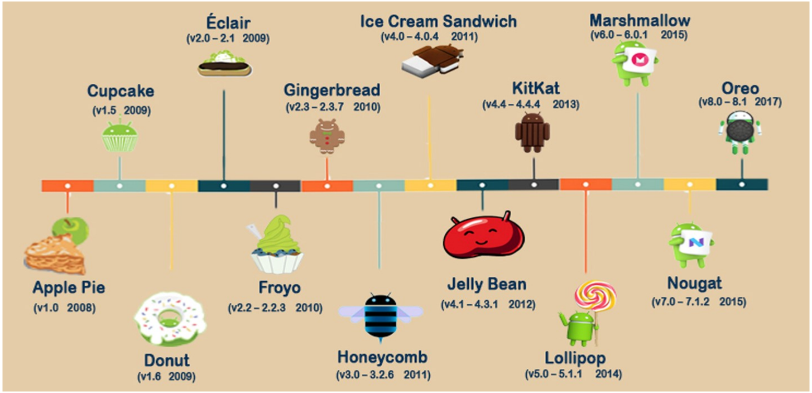
Android [https://www.android.com/]

- Open software platform for mobile development
- A complete stack
  - OS, Middleware, Applications
- An Open Handset Alliance (OHA) project
- Powered by Linux operating system
- Open source under the Apache 2 license

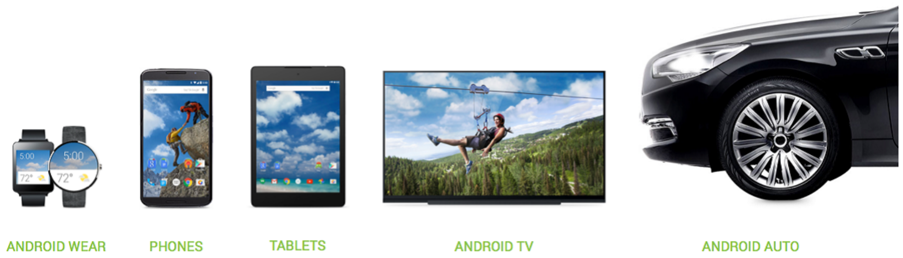
- Android ships with a rich set of applications
  - Email, calendar, browser, maps, text messaging, contacts, camera, dialer, music player, settings and others



Current (August 2018) version: **Android Pie 9.0**



Powering screens of all sizes



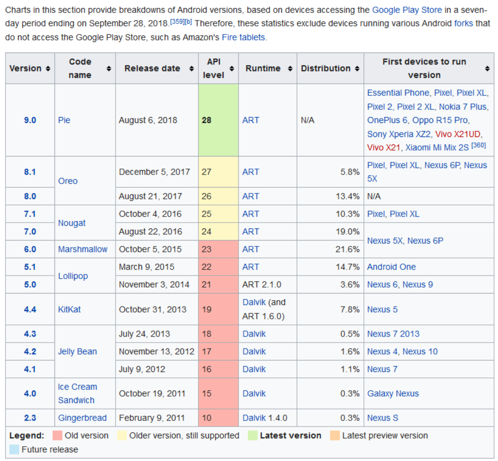
ANDROID FOR SANBOT

Many different versions

[https://en.wikipedia.org/wiki/Android_(operating_system)]
[http://developer.android.com/guide/topics/manifest/uses-sdk-element.html]

[http://d.android.com/resources/dashboard/platform-versions.html]

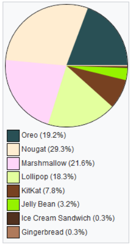
The table below specifies the API Level supported by each version of the Android platform.



Charts in this section provide breakdowns of Android versions, based on devices accessing the Google Play Store in a seven-day period ending on September 28, 2018.[359][b] Therefore, these statistics exclude devices running various Android forks that do not access the Google Play Store, such as Amazon's Fire tablets.

Version	Code name	Release date	API level	Runtime	Distribution	First devices to run version
9.0	Pie	August 6, 2018	**28**	ART	N/A	Essential Phone, Pixel, Pixel XL, Pixel 2, Pixel 2 XL, Nokia 7 Plus, OnePlus 6, Oppo R15 Pro, Sony Xperia XZ2, Vivo X21UD, Vivo X21, Xiaomi Mi Mix 2S [360]
8.1	Oreo	December 5, 2017	27	ART	5.8%	Pixel, Pixel XL, Nexus 6P, Nexus 5X
8.0		August 21, 2017	26	ART	13.4%	N/A
7.1	Nougat	October 4, 2016	25	ART	10.3%	Pixel, Pixel XL
7.0		August 22, 2016	24	ART	19.0%	Nexus 5X, Nexus 6P
6.0	Marshmallow	October 5, 2015	23	ART	21.6%	
5.1	Lollipop	March 9, 2015	22	ART	14.7%	Android One
5.0		November 3, 2014	21	ART 2.1.0	3.6%	Nexus 6, Nexus 9
4.4	KitKat	October 31, 2013	19	Dalvik (and ART 1.6.0)	7.8%	Nexus 5
4.3	Jelly Bean	July 24, 2013	18	Dalvik	0.5%	Nexus 7 2013
4.2		November 13, 2012	17	Dalvik	1.6%	Nexus 4, Nexus 10
4.1		July 9, 2012	16	Dalvik	1.1%	Nexus 7
4.0	Ice Cream Sandwich	October 19, 2011	15	Dalvik	0.3%	Galaxy Nexus
2.3	Gingerbread	February 9, 2011	10	Dalvik 1.4.0	0.3%	Nexus S

Legend: 🟥 Old version 🟨 Older version, still supported 🟩 **Latest version** 🟧 Latest preview version 🟦 Future release

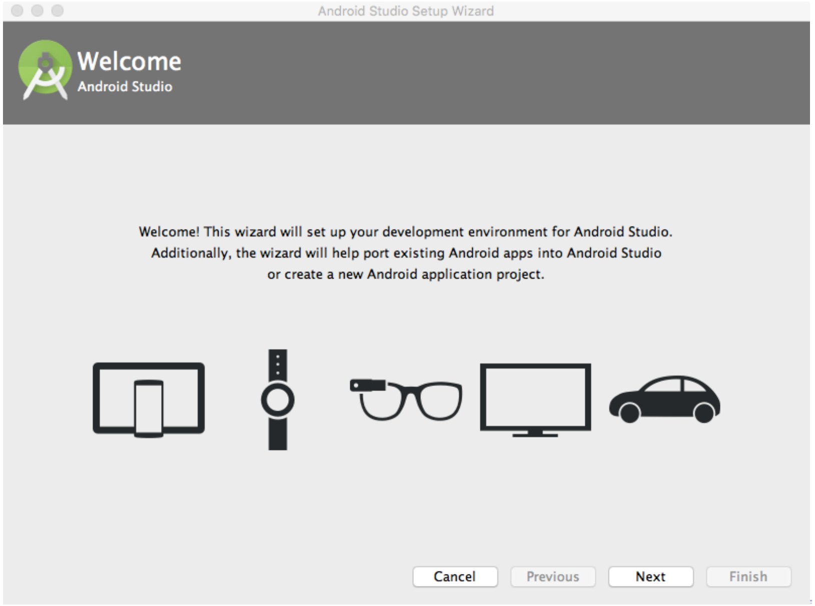


Outline

Set up

- Set up the development environment
  - Java Development Kit (JDK) 7.0+
  - Android Studio (v. 3.0.1)[1]
    - Integrated Development Environment Based on IntelliJ
    - Editor with advanced features
    - Debugger
    - Compiler
    - Build Automation using Gradle
    - Graphical Interface Builder based on XML
    - Android Emulator with performance enhancements
- Set up Android Virtual Devices (AVDs) and devices for testing
  - Create Android Virtual Devices (AVDs)
  - Connect real (hardware) devices
  - Use external emulators (Genymotion)

[1] An alternative was Eclipse: in May 2013, Google introduced Android Studio, a new development environment based on IntelliJ IDEA by JetBrains [`http://www.jetbrains.com/idea`]. Before, most Android developers used the Eclipse IDE [`http://www.eclipse.org`] and the Android Development Tools.



Android Studio Setup Wizard

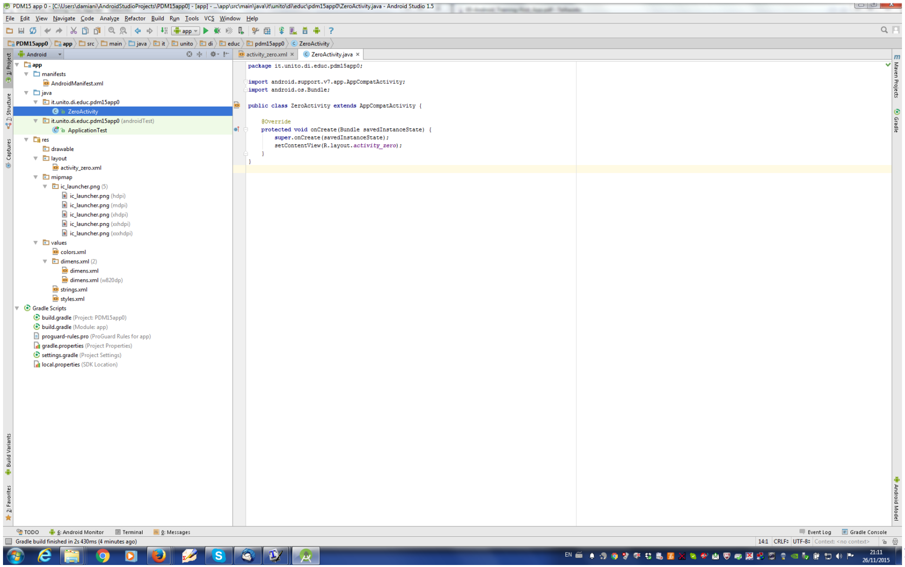
Welcome
Android Studio

Welcome! This wizard will set up your development environment for Android Studio. Additionally, the wizard will help port existing Android apps into Android Studio or create a new Android application project.

Development

1. Think/Prototype
2. Design
3. Develop
    - Create an application
        - Source code,
        - resource files, and
        - Android manifest file
    - Build and run the application
    - Test and debug the application
    - Publish the application

Android Project Anatomy

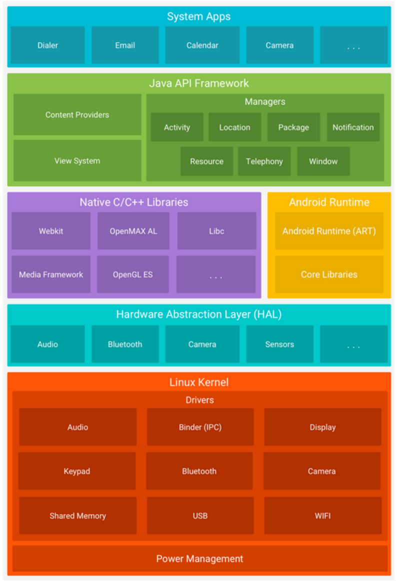

PDM15 app 0 - [C:\Users\damiani\AndroidStudioProjects\PDM15app0] - [app] - ...\app\src\main\java\it\unito\di\educ\pdm15app0\ZeroActivity.java - Android Studio 1.5
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PDM15app0 › app › src › main › java › it › unito › di › educ › pdm15app0 › ZeroActivity
Android
activity_zero.xml
ZeroActivity.java
TODO
6: Android Monitor
Terminal
0: Messages
Event Log
Gradle Console
Gradle build finished in 2s 430ms (4 minutes ago)

```
app
    manifests
        AndroidManifest.xml
    java
        it.unito.di.educ.pdm15app0
            ZeroActivity
        it.unito.di.educ.pdm15app0 (androidTest)
            ApplicationTest
    res
        drawable
        layout
            activity_zero.xml
        mipmap
            ic_launcher.png (5)
                ic_launcher.png (hdpi)
                ic_launcher.png (mdpi)
                ic_launcher.png (xhdpi)
                ic_launcher.png (xxhdpi)
                ic_launcher.png (xxxhdpi)
        values
            colors.xml
            dimens.xml (2)
                dimens.xml
                dimens.xml (w820dp)
            strings.xml
            styles.xml
Gradle Scripts
    build.gradle (Project: PDM15app0)
    build.gradle (Module: app)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

```java
package it.unito.di.educ.pdm15app0;

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

public class ZeroActivity extends AppCompatActivity {

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

Outline

Android system architecture [https://developer.android.com/guide/platform/index.html]



Linux Kernel

- Linux provides the hardware abstraction layer for Android.
- Android uses Linux for its memory management, process management, networking, and other operating system services.
  - The Android user is not supposed to see Linux.
  - Apps will not usually make Linux calls directly.
  - Some utilities needed during development interact with Linux. E.g., the `adb shell` command [`http://d.android.com/tools/help/adb.html`] opens a Linux shell in which the developer can enter other commands to run on the device.

Libraries (Android native libraries)

- Shared libraries (to be called by higher-level programs): written in C or C++, compiled for the particular hardware architecture used by the Android device, and preinstalled by the vendor.[2] Some of them are:
  - *Surface Manager:* drawing commands do not draw directly to the screen; they are saved into lists that are then combined with lists from other windows and are then used to form the display the user sees.
  - *2D and 3D graphics:* two- and three-dimensional elements are combined in a single user interface; everything is converted into 3D drawing lists.
  - *Media codecs:* play video and record and play back audio in various formats.
  - *SQL database:* lightweight SQLite database engine [`http://www.sqlite.org`] (same database used in Firefox and the Apple iPhone). Provides persistent storage in apps.
  - Browser engine: Chromium library [`http://www.chromium.org`] (same engine used in the Google Chrome). To display HTML content.

[2] You can write and deploy your own native libraries using the Native Development Toolkit (NDK) [`http://d.android.com/tools/sdk/ndk`]—not part of this course.

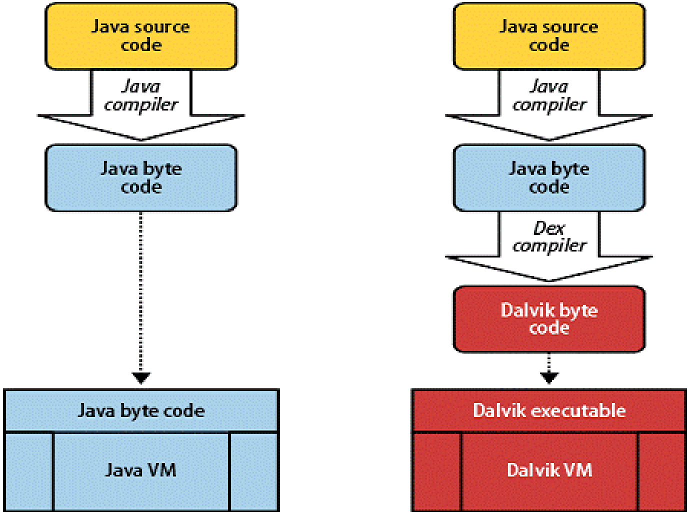
Android Runtime

- Includes the runtime environment and the core Java libraries.
- Depending on the version of Android, the environment uses either Dalvik or ART, which are Google's semi-compatible implementation of Java.
  - Dalvik is a virtual machine (VM) designed and written at Google. Code gets compiled into machine-independent instructions (the bytecodes), which are executed by the Dalvik VM on the device.
  - ART (Android Runtime) is an ahead-of-time compiler[3] that, when an application is installed onto an Android device, compiles it into machine code—this makes programs run faster (compared to Dalvik) at the expense of a longer install time.
- The core Java libraries that come with Android are different from both the Java Standard Edition (Java SE) libraries and the Java Mobile Edition (Java ME) libraries—although there is a substantial overlapping.

[3] It replaced Dalvik in Android 5.0 (Lollipop).

Dalvik Virtual Machine ($\leq$ 4.4)

- Source code compiled to a class file (.class)
- Transformed to Dalvik Executable (.dex)—compact and efficient

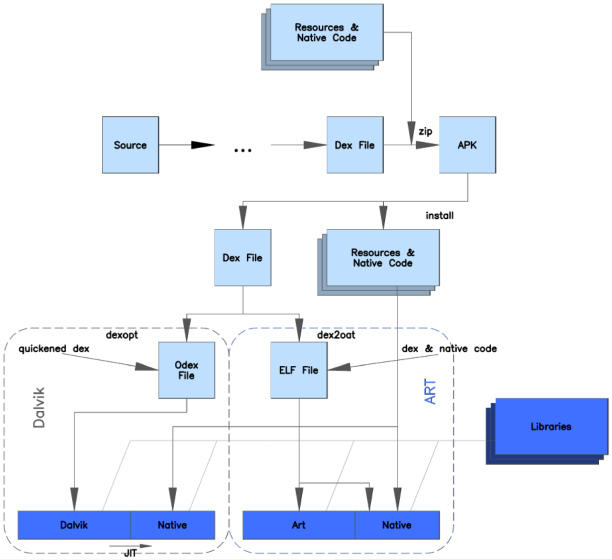


Android Runtime ($\geq$ 5.0)

- Introduces the use of ahead-of-time (AOT) compilation. I.e.:

  - Compiles entire applications into native machine code upon their installation

  - Eliminates Dalvik's interpretation and trace-based JIT compilation
- Improves overall execution efficiency and reduces power consumption
- Brings faster execution of applications, improved memory allocation and garbage collection



Application Framework

This layer provides the high-level building blocks to be used to create apps. It comes preinstalled with Android (but it can be extended with new components as needed). The most important parts are:

- *Activity manager:* controls the life cycle of applications and maintains a common “backstack” for user navigation.
- *Content providers:* these objects encapsulate data that needs to be shared between applications, such as contacts.
- *Resource manager:* resources are anything that goes with a program that is not code.
- *Location manager:* an Android device always knows where it is.
- *Notification manager:* to present to the user events such as arriving messages, appointments, proximity alerts, etc.

Applications and Services

Applications: programs that can take over the whole screen and interact with the user—end users see only the applications.

Services: operate invisibly to extend the application framework.

- Android devices come prepackaged with a number of standard system applications (e.g., Google Play Store, Phone dialer, Camera, Web browser,...)
- Play Store allows users to download new programs.
- The Android framework provides a number of building blocks that can be used to create new applications.

Outline

Two fundamental concepts about the Android application framework: (1) Apps provide multiple entry points

Android apps are built as a combination of distinct components that can be invoked individually.

Example

An individual *activity* provides a single screen for a user interface, and a *service* independently performs work in the background.

From one component you can start another component using an *intent*. You can even start a component in a different app, such as an *activity* in a maps app to show an address.

This model provides multiple entry points for a single app and allows any app to behave as a user's "default" for an action that other apps may invoke.

Two fundamental concepts about the Android application framework: (2) Apps adapt to different devices

Android provides an adaptive app framework that allows you to provide unique resources for different device configurations.

Example

You can create different XML layout files for different screen sizes and the system determines which layout to apply based on the current device's screen size.

You can query the availability of device features at runtime if any app features require specific hardware such as a camera. If necessary, you can also declare features your app requires so app markets such as Google Play Store do not allow installation on devices that do not support that feature.