Software Engineering Large Practical: Android concepts and programming

Stephen Gilmore
(`Stephen.Gilmore@ed.ac.uk`)
School of Informatics

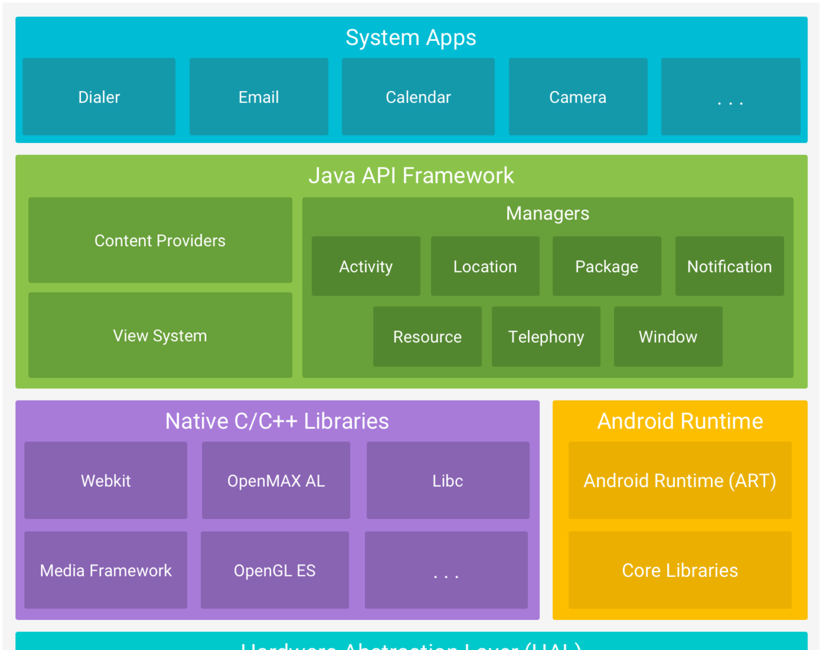
September 28, 2016

Contents

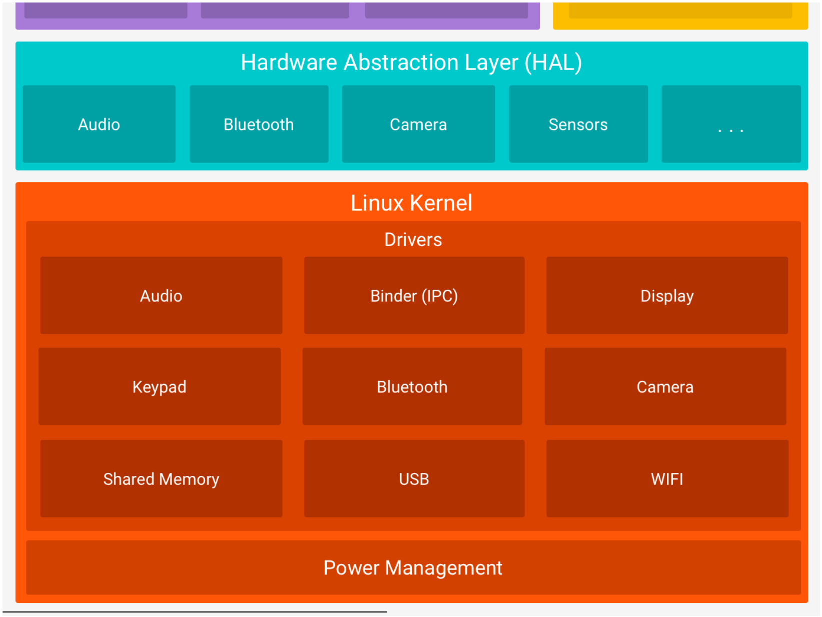
The Android platform

- Android is an open source, Linux-based software stack.
- The *Android Runtime (ART)* relies on the Linux kernel for functionalities such as threading and memory management.
- ART is written to run multiple virtual machines on low-memory devices by executing *DEX* files, a bytecode format designed specially for Android.
- The *hardware abstraction layer (HAL)* provides interfaces that expose device capabilities to the higher-level *Java API framework.*
- Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++.

Android Platform Architecture (Upper)

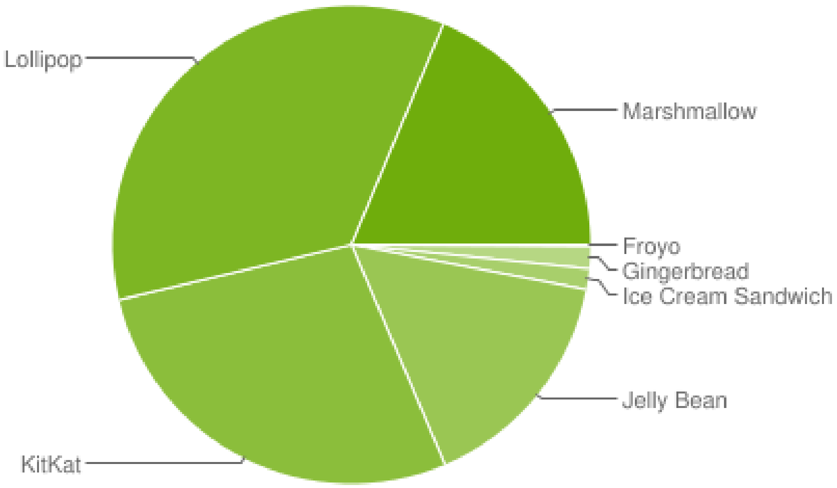


Android Platform Architecture (Lower)



Credit: `https://developer.android.com/guide/platform/index.html`

Android versions by market share (as of September 2016)

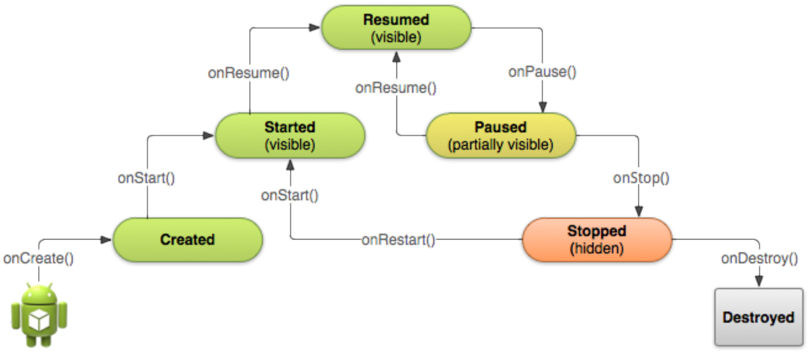


From `https://developer.android.com/about/dashboards/`

Android activities

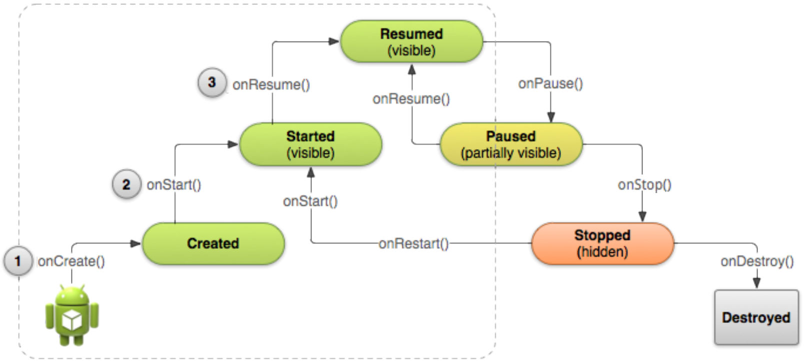
- An Android app is split up into a number of different *activities*, which are subclasses of `android.app.Activity`, or subclasses of that class, such as `android.support.v7.app.AppCompatActivity`.
- An activity represents a single screen with a user interface.
- Activities differ in nature from the main class of a Java application, in that it must be possible to pause, suspend, and resume them and have the app take action depending on which of these events happens. (For example, an incoming phone call would cause an app to be interrupted.)
- The allowable calls to methods such as `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, `onRestart()` and `onDestroy()` makes up the Android activity lifecycle.

Android Activity lifecycle



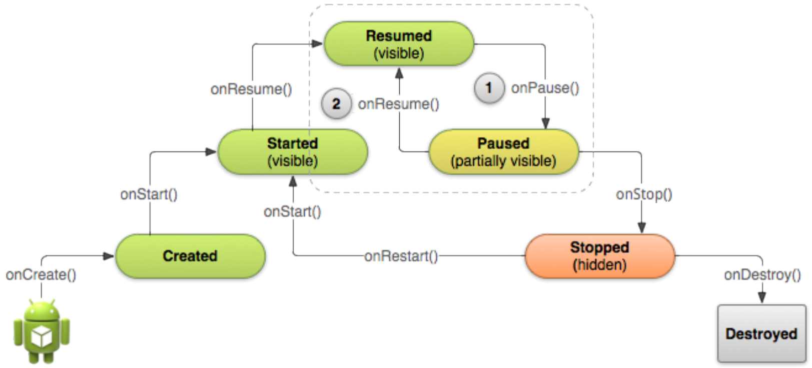
From `https://developer.android.com/training/basics/activity-lifecycle/starting.html`

Android Activity lifecycle (create)



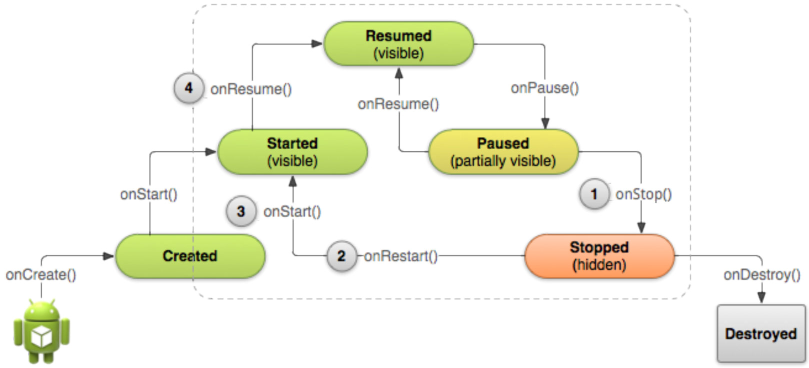
From `https://developer.android.com/training/basics/activity-lifecycle/starting.html`

Android Activity lifecycle (paused)



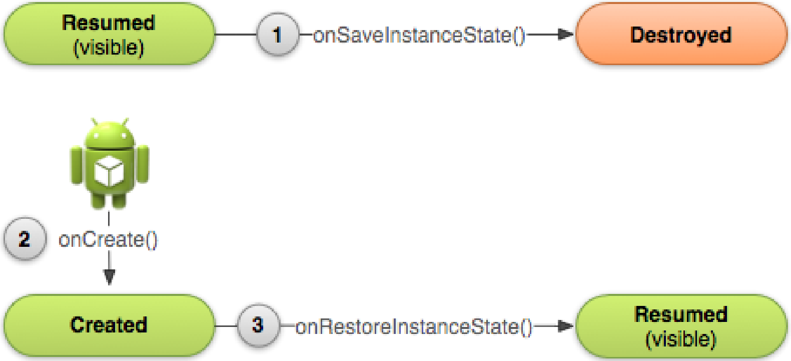
From https://developer.android.com/training/basics/activity-lifecycle/pausing.html

Android Activity lifecycle (stopping)



From `https://developer.android.com/training/basics/activity-lifecycle/stoping.html`

Android Activity lifecycle (saving state)

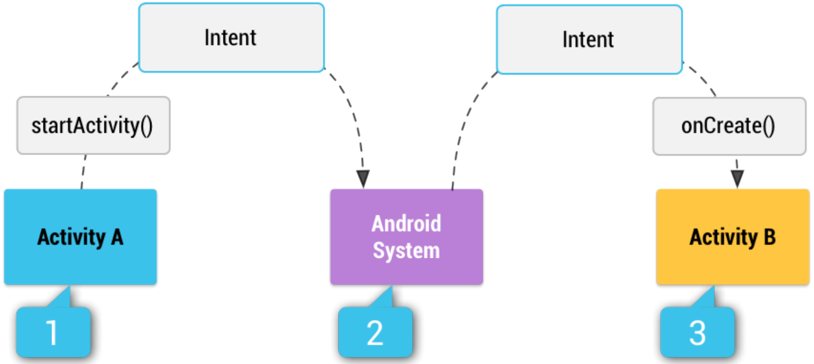


From `https://developer.android.com/training/basics/activity-lifecycle/recreating.html`

Using Intents

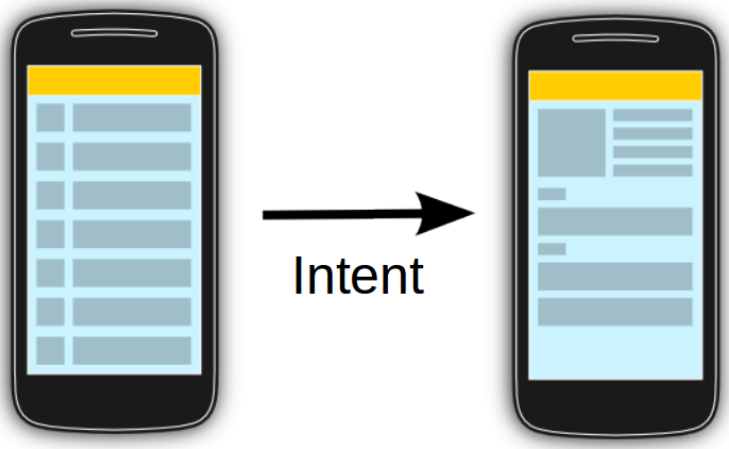
- An *intent* of `android.content.Intent` is a messaging object which can be used to communicate with another app component such as another `Activity`.
- You can start a new instance of an `Activity` by passing an `Intent` to `startActivity()`. The `Intent` describes the activity to start and carries any necessary data.
- If a result is expected then `startActivityForResult()` is called instead.
- Intents can also be used to start a `Service` of class `android.app.Service`.

One mechanism of activity starting another



From `https://developer.android.com/guide/components/intents-filters.html`

The effect of one activity starting another



From http://www.vogella.com/tutorials/AndroidIntent/article.html

Android projects

- Android projects contain a mix of Java and XML code in a structured project which contains

manifests	Contains the `AndroidManifest.xml` file which provides essential information about your app to the Android system, to allow it to run your code.
java	Contains the Java source code files, separated by package names, including JUnit test code.
res	Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

- In addition, Android projects also contain *build files* for compiling the project source code into an executable. Android uses the *Gradle* build system.
- Java code describing resources is automatically generated from XML source code by Android Studio.

Android compiler toolchain

Android previously used a different back-end format from Java, now it has a different compiler as well.

Typical Java javac toolchain

javac (.java $\rightarrow$ .class) $\rightarrow$ **java**

Legacy Android javac toolchain

javac (.java $\rightarrow$ .class) $\rightarrow$ **dx** (.class $\rightarrow$ .dex)

New Android Jack toolchain

Jack (.java $\rightarrow$ .jack $\rightarrow$ .dex)

Jack does not support all Java 8 features but does support *default and static interface methods*, *lambda expressions*, *repeatable annotations*, *method references*, and *type annotations*.

Tutorial: Android Studio, from zero knowledge to something basic

Jonathan Warner

https://www.youtube.com/watch?v=-igAiudpBng

Links

- `https://developer.android.com/` — Android information
- `https://developer.android.com/studio/` — to download Android Studio
- `http://www.oracle.com/technetwork/java/javase/downloads/` — to download Java 8