Android Introduction

Architecture

Activities, Lifecycle

Development Environment

Why Android?

- Pervasive Mobile Platform
  - Runs on hundreds of millions of mobile phones, tablets
  - World's most popular & frequently installed mobile OS
  - Open Source (minimum-definition).

- Developing on Android
  - Familiar language (e.g. Java)
  - Multi-platform support
  - Android Studio, IntelliJ support

- See http://developer.android.com/tools/index.html

Android Design Constraints

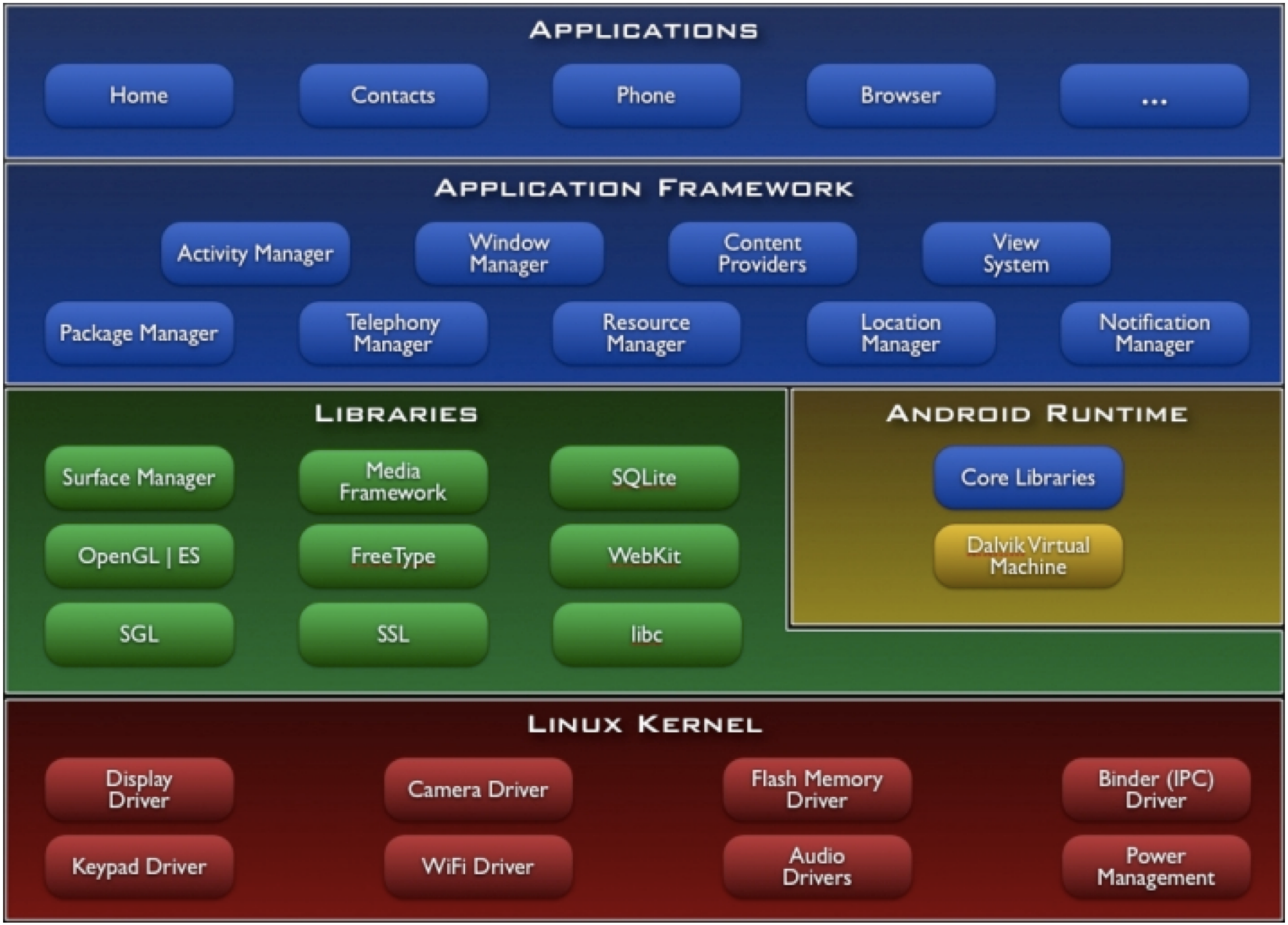
- Mobile OS with limited resources
  - Limited memory, processing power
  - Battery life is critical
- Architectural tradeoffs
  - The OS aggressively flushes memory
  - Background computation is limited
  - Single application focus
- Unusual device characteristics
  - Small screen
  - Multiple orientations, dynamic layout
  - Multi-touch input

Architecture

- Applications are built in Java (or Kotlin, or C++)
  - Android compiler generates an Android Package (.apk file), which contains code, resources etc.
  - Package is installed using SDK, which sets up environment for that app.

- Applications run securely in the environment
  - Android is a Linux environment where every app is a distinct user!
  - When the app is installed, permissions are set to restrict access (resources, data).
  - Every app runs in its own process.
  - Apps must request access to shares resources (e.g. file system, camera)

- As a developer, you create a *manifest* file in your package that describes how your application should be installed, what permissions it requires etc.


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

App Components

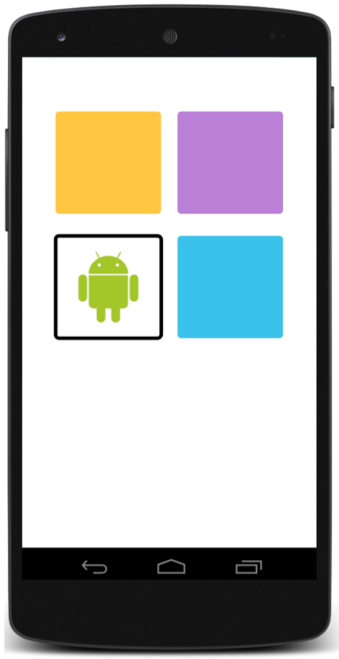
Android uses the term application components to describe the type of "building block" that is supported.

- An application can consist of multiple application components.
- Each component is an entry point through which the system or a user can enter or access your application.

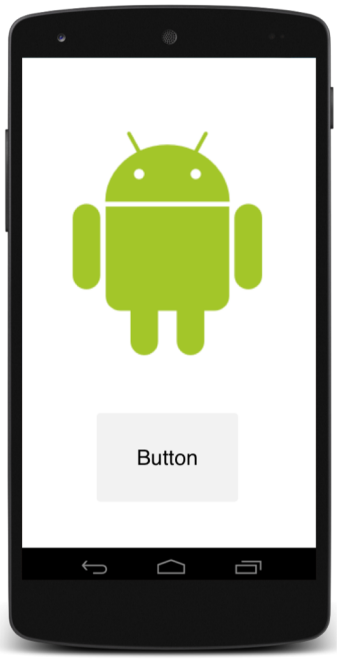
Components	Description
Activity	Single-screen of an application
Service	Long-running background process
Content provider	Provides shared set of application data
Broadcast receiver	Responds to system broadcast events

Activities

- The “standard” application component is an Activity
- Typically represents a single screen of your application (and you may have multiple activities, one of which will be running at a time).
  - Not a view, since it can contains both model + view
- One activity will be the Main entry point for your application (aka Main).



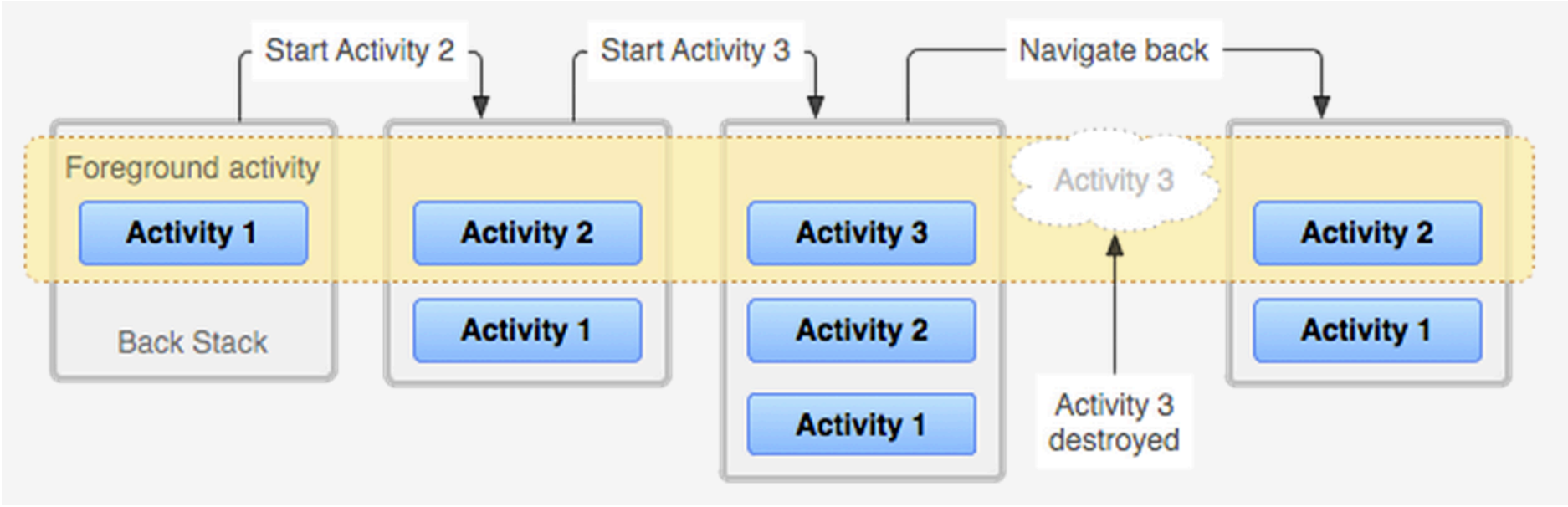
Activity 1



Activity 2

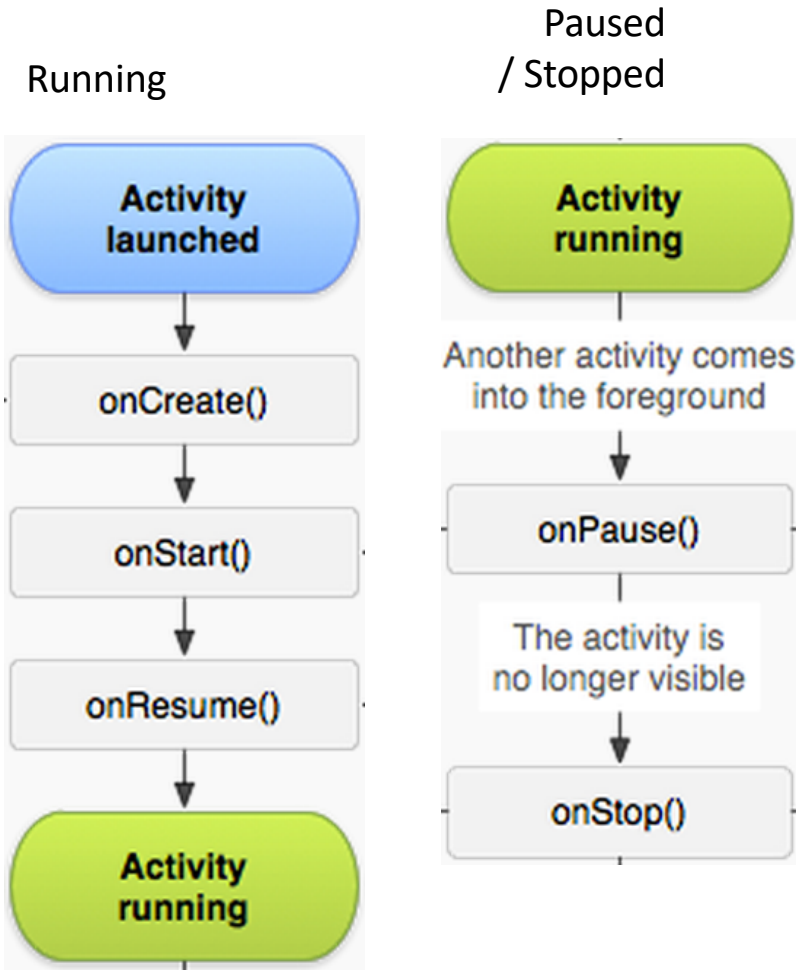
Activity Navigation

- You can have multiple activities in your application (e.g. different screens), and switch between them as-needed.
  - Activities can create other activities (i.e. “back stack” of activities that you can reach by using the back button)
  - Navigation forward/back through activities is typically triggered by user actions



Activity Lifecycle

- Activities have an explicit lifecycle, and have a state reflecting what they are doing (e.g. “started” or “stopped”)
- Changing state fires a *callback method* that corresponds to that state (e.g. going from “stopped” to ”started” causes the *onStart()* method to fire).

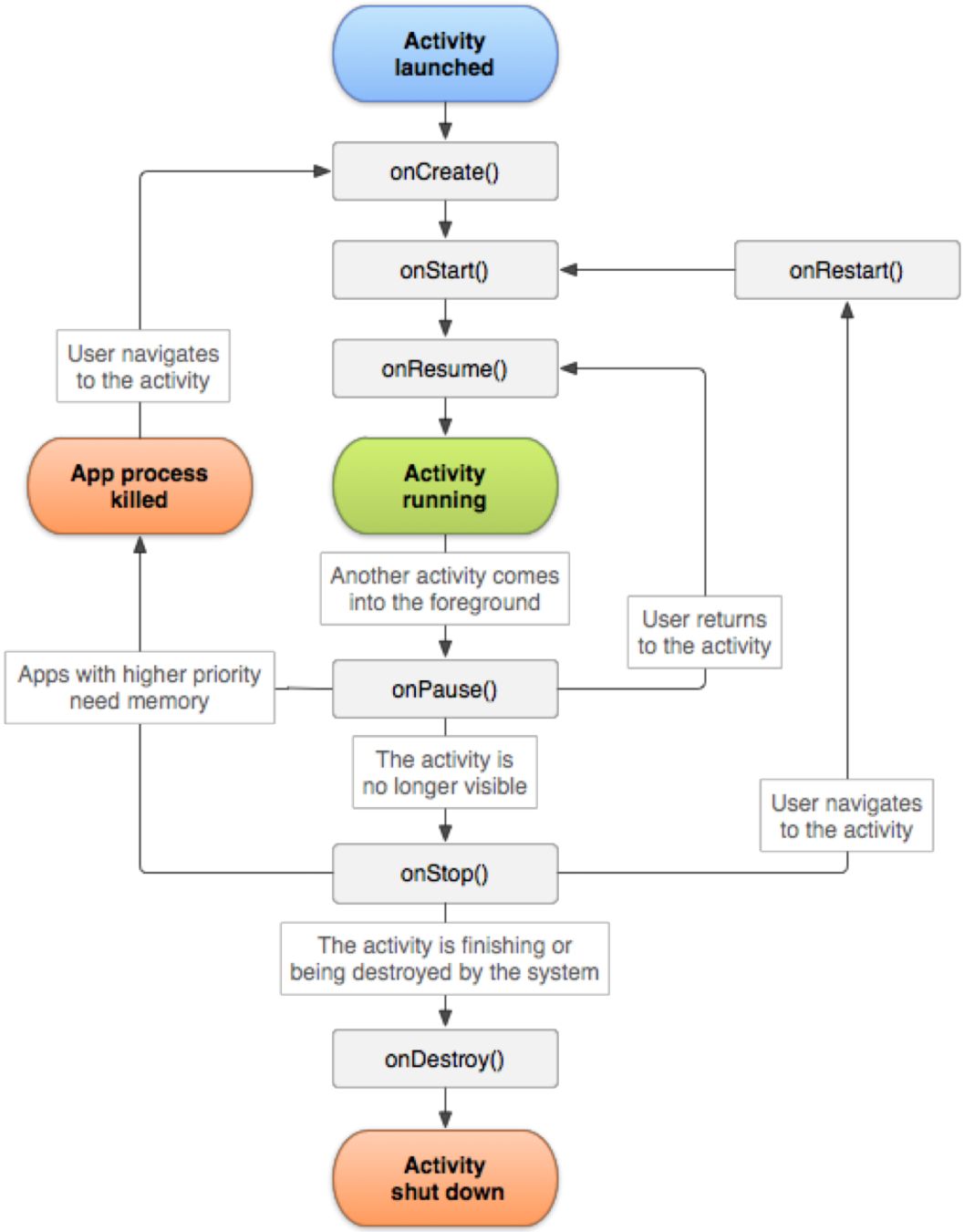


https://developer.android.com/guide/components/activities/activity-lifecycle.html

Managing the Activity Lifecycle

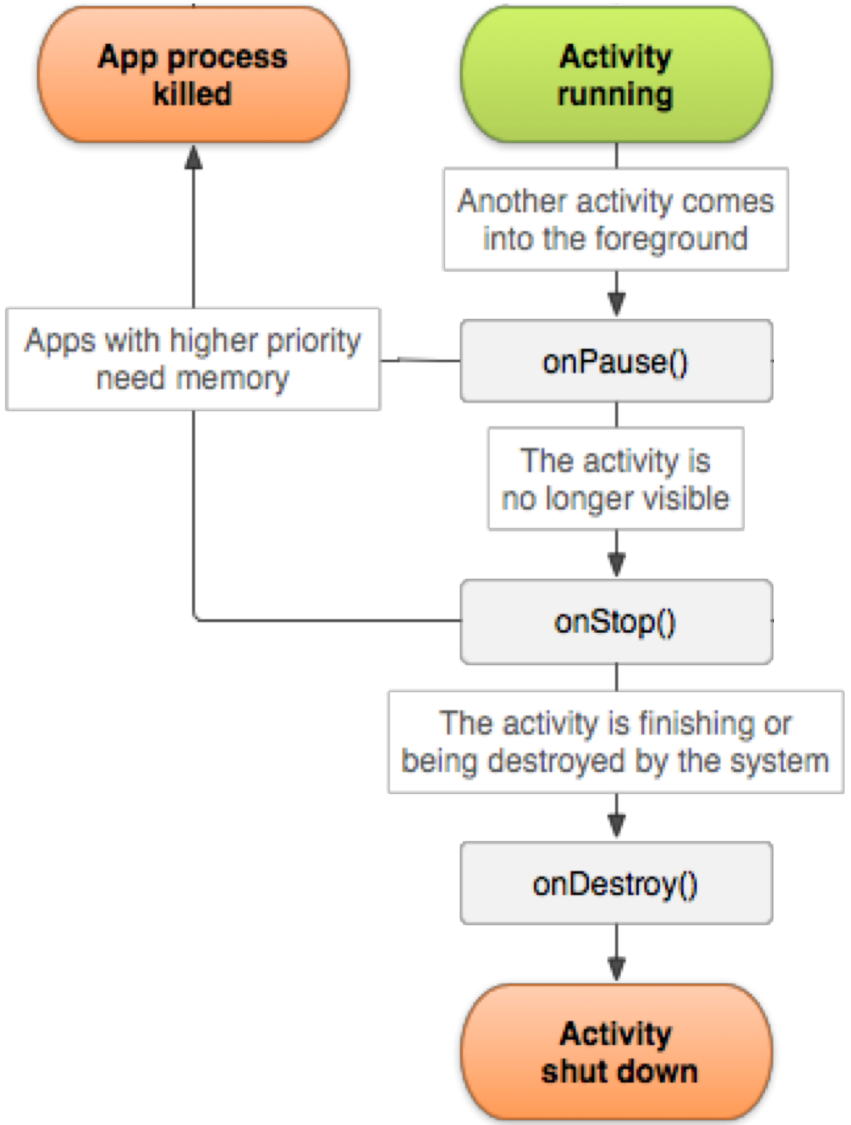
Core callback functions:

- onCreate()
  - being created or launching
- onStart()
  - becomes visible to user
- onResume()
  - prior to user interaction
- onPause()
  - loses focus or background
- onStop()
  - no longer visible to user
- onDestroy()
  - being recycled and freed



Interrupted Workflow

- Applications can stop at any time (i.e. user quits, OS kills it, user presses the Back button
  - the activity transitions through the onPause(), onStop(), and onDestroy() callbacks.
  - the activity is also removed from the back stack.
- To preserve simple transient-state data
  - Override the onSaveInstanceState() method to save the data, and then use onCreate() or onRestoreInstanceState() callbacks to recreate the instance state



Intents

- An **Intent** is a messaging object you can use to request an action from another application component
  - Starting an activity
  - Starting a service
  - Delivering a broadcast

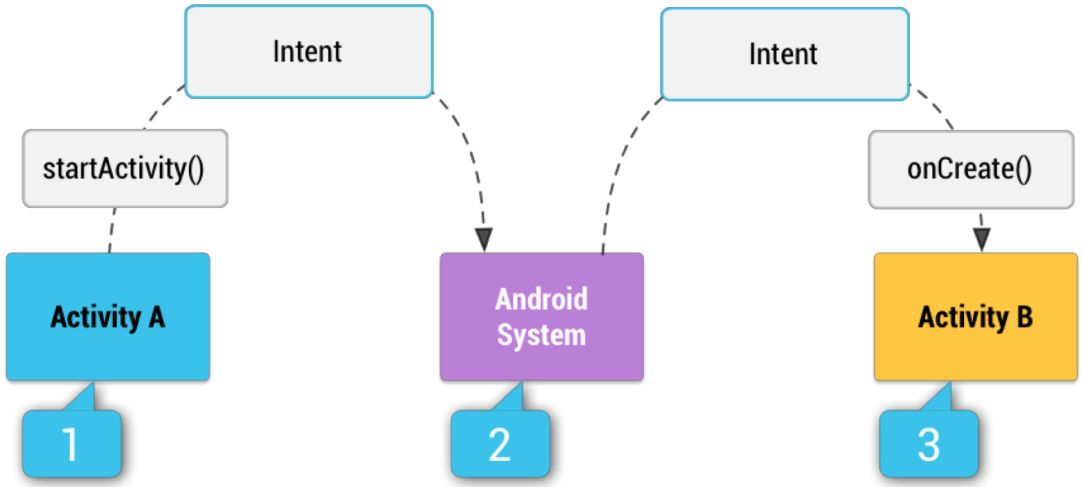
- This allows an application to use other application services! e.g. Instagram app can request access to the Camera activity to take a picture.
  - Eliminates the need to have functionality embedded in an application.
  - Allows the OS to control access/permissions to services.

- We use **intents** to pass data between activities.
  - Basically a data structure holding an abstract description of an action

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

Intents

- Use *startActivity()* method to launch an activity with an intent.
  - Can call *explicit* named activity (e.g. mySettingsActivity) or an *implicit* activity based on its capabilities (e.g. some camera activity)



Development Environment

- Java JDK
  - https://www.oracle.com/technetwork/java/javase/downloads/index.html
  - https://jdk.java.net/11

- Android SDK
  - https://developer.android.com/studio
  - Included in the Android Studio installation

- Development Environment
  - Plan on using IntelliJ or Android Studio
  - Examples are provided as IntelliJ projects
  - https://www.jetbrains.com/idea/download

Setup

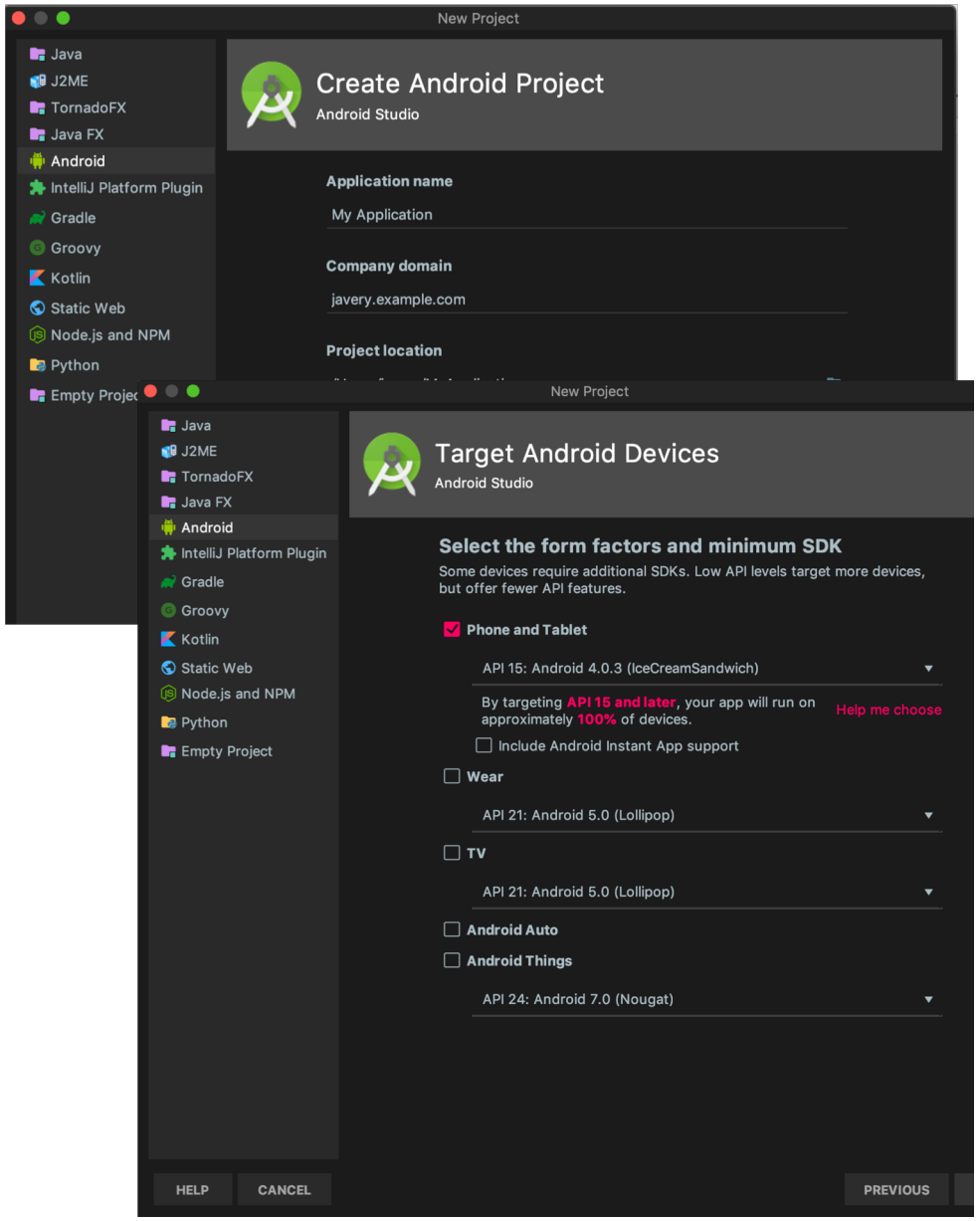
1. Download and install Java JDK 8 *
2. Download and install the Android SDK 28.
3. Download and install IntelliJ.
   - Enable the Android plugin during installation.
4. Open a sample project (or create one) to check that it works.
5. IntelliJ, Tools -> Android, SDK Manager
   - Update everything

* Currently the version that is supported by IntelliJ and Android Studio. You can use a later version of Java but you will need to manually update a bunch of other things, including Gradle.

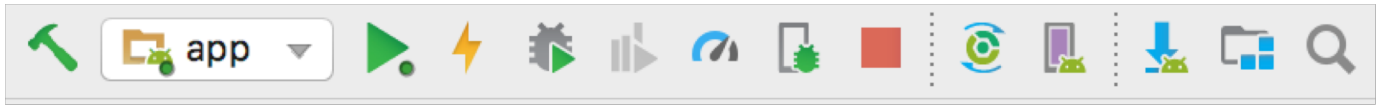
https://www.jetbrains.com/help/idea/getting-started-with-android-development.html

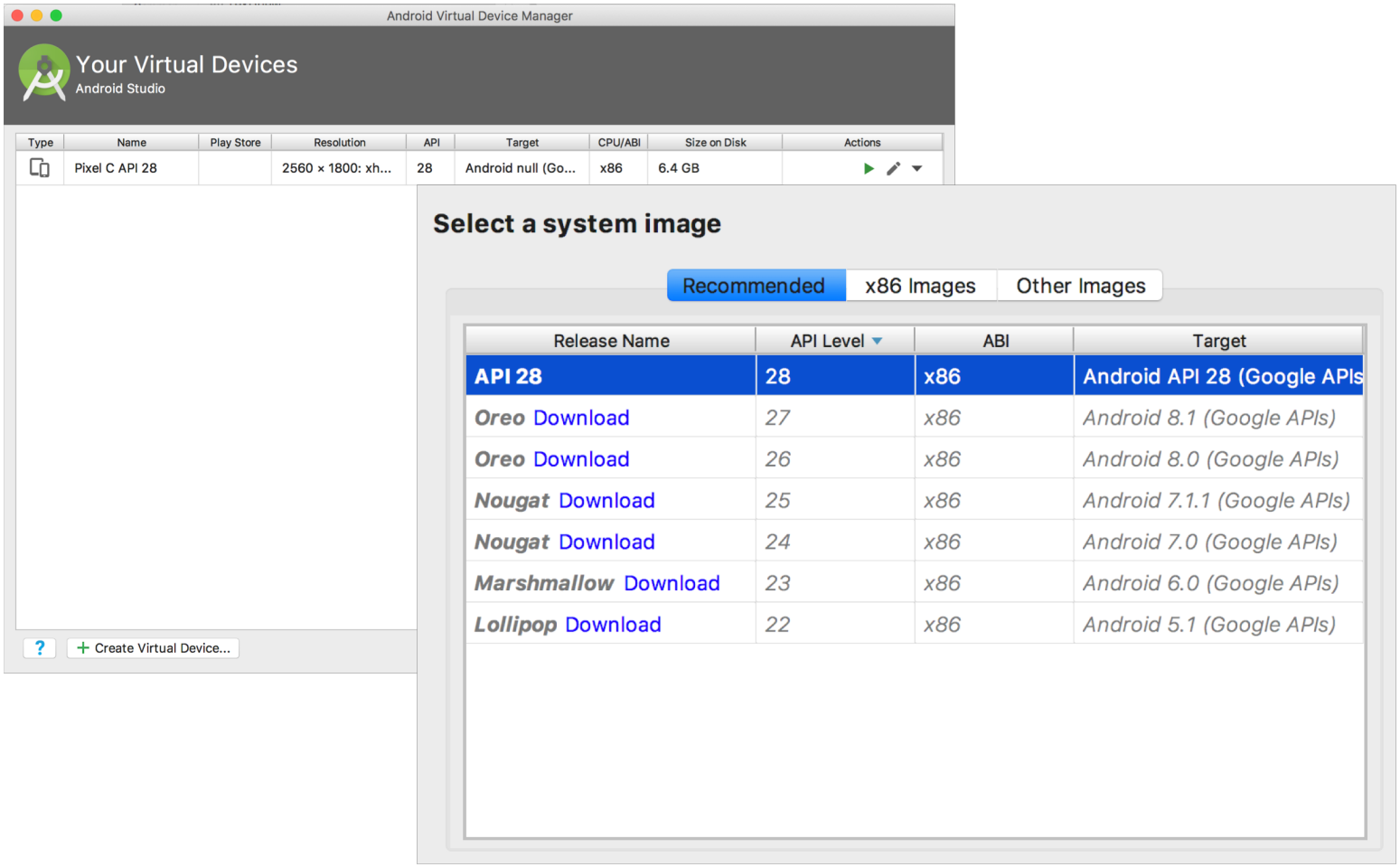
Project Wizard

- Company Domain
  - only important if you release your app, can just use something like: `cs349.uwaterloo.ca`
- API to target is the minimum Android version on target devices
  - Use API 15 for Phone and Tablet (we won't be doing anything restrictive)
- SDK version is the version of the dev tools, libraries etc.
  - Android Studio defaults to 28
- Activity: Whatever you start with, do NOT use fragments



Create Android Virtual Device (AVD) for Testing





Sample Code

- File -> Open (Project from Examples Directory)
- Run, Select Deployment Target
  - Can launch AVD or push to connected device