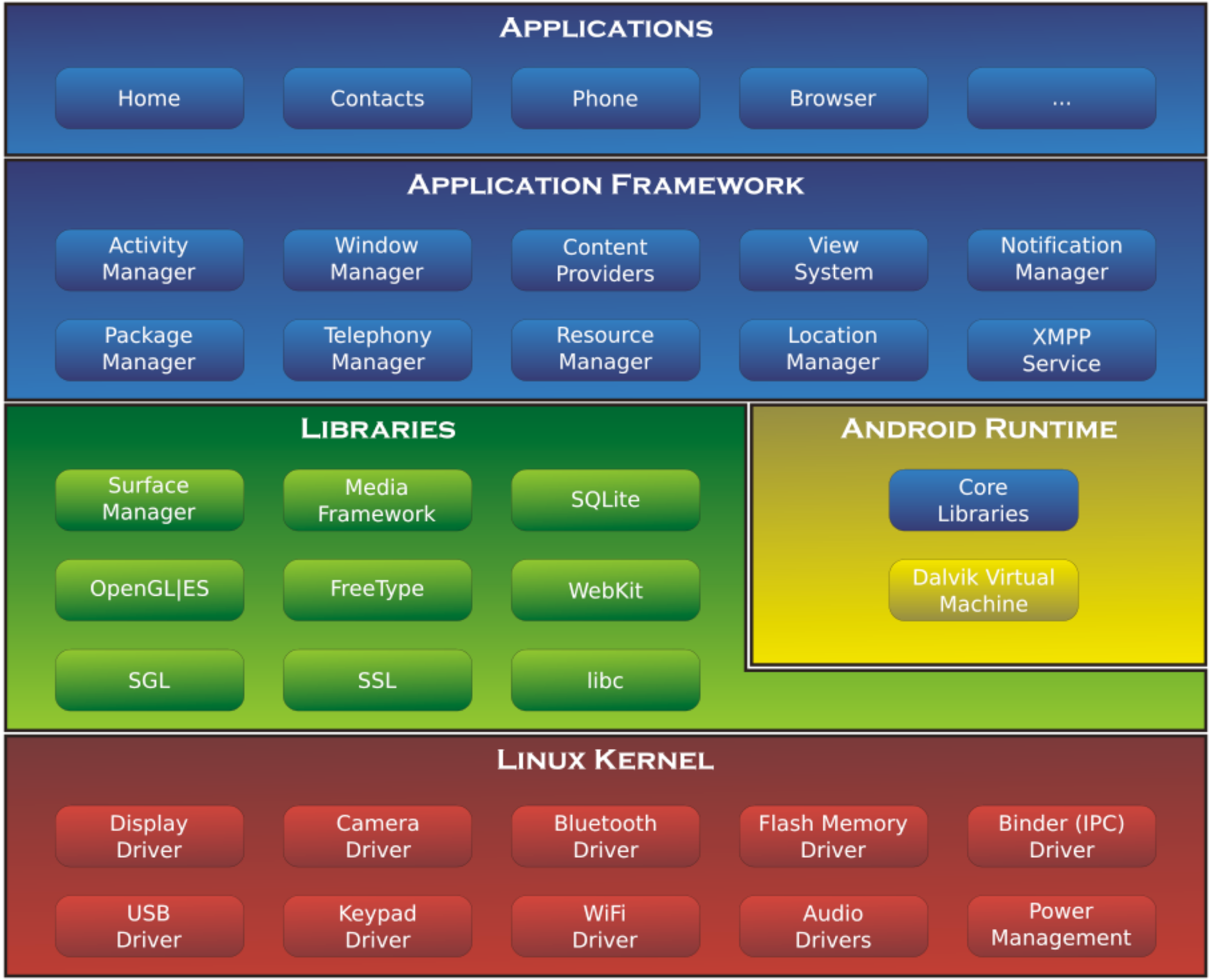
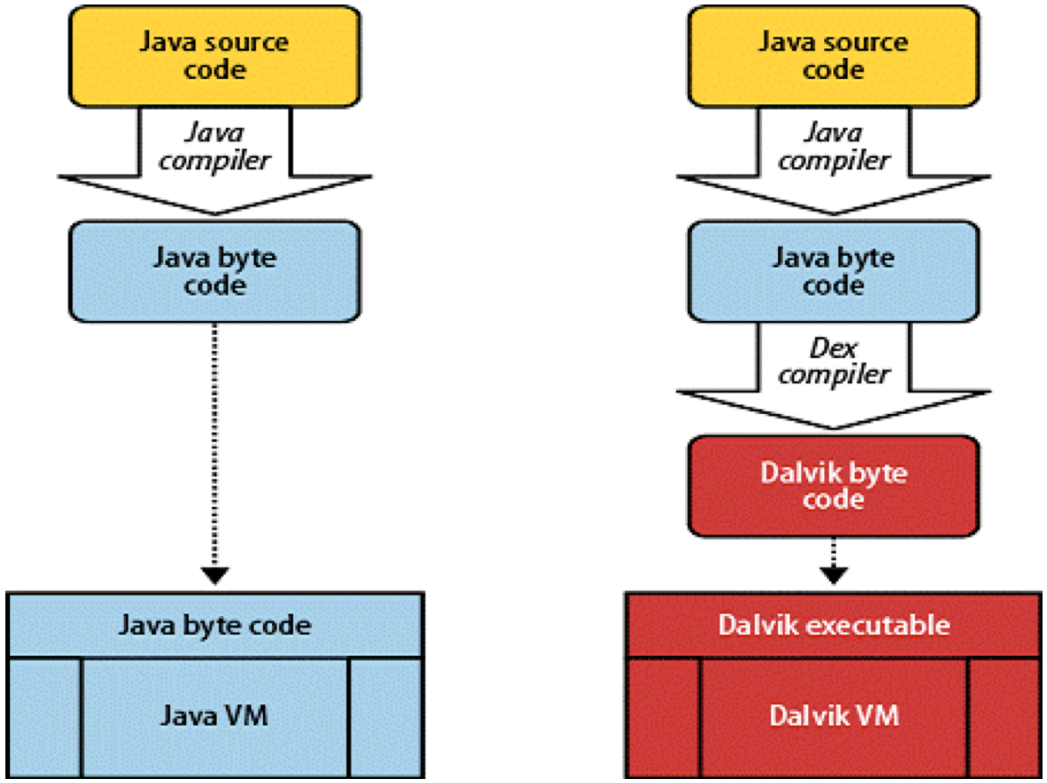
Android

Android

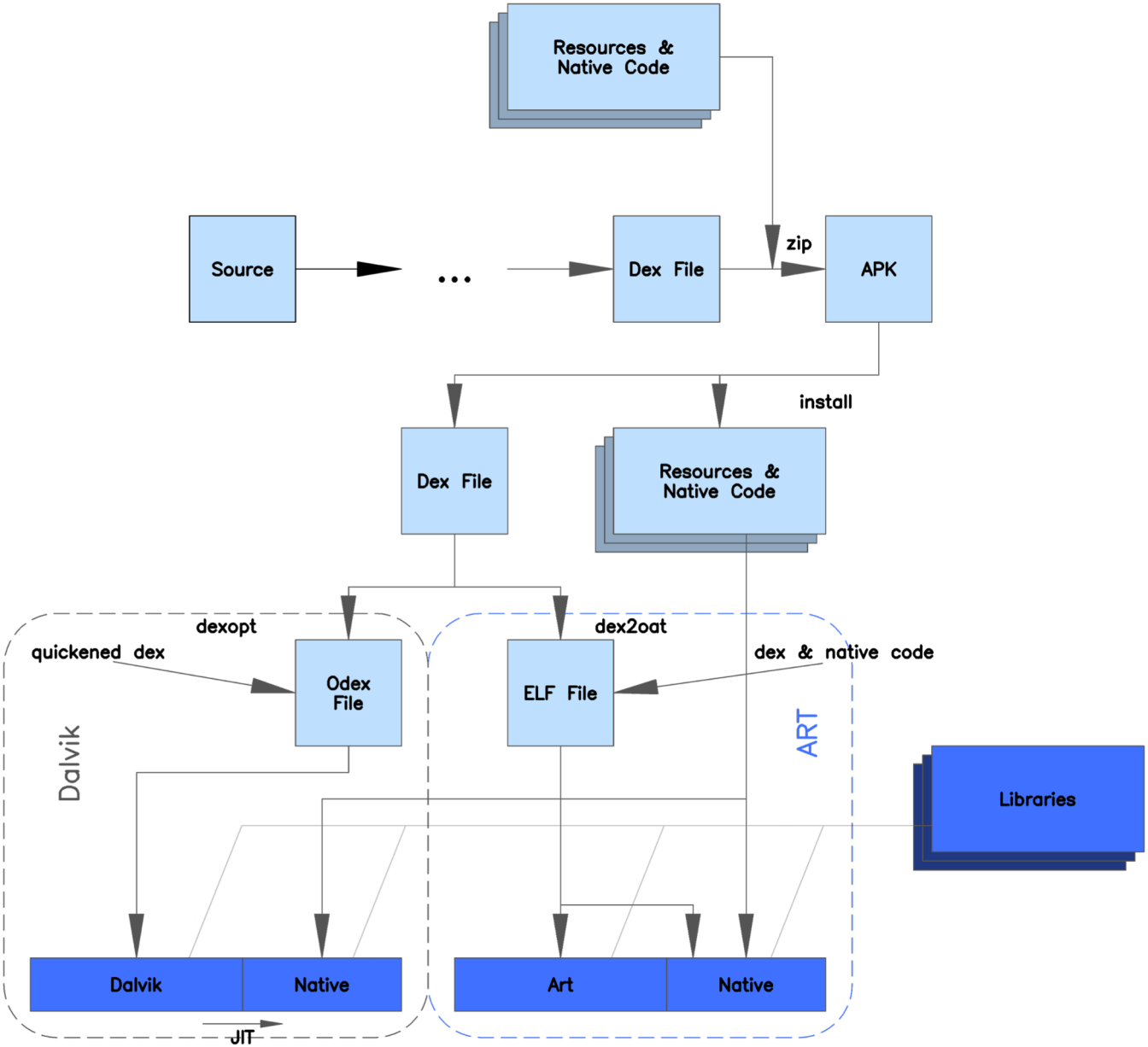
- Mobile operating system developed by Google
- A complete stack
  - OS, framework
  - A rich set of applications
    - Email, calendar, browser, maps, text messaging, contacts, camera, dialer, music player, settings and others
- Based on the Linux kernel
- Open source under the Apache 2 license

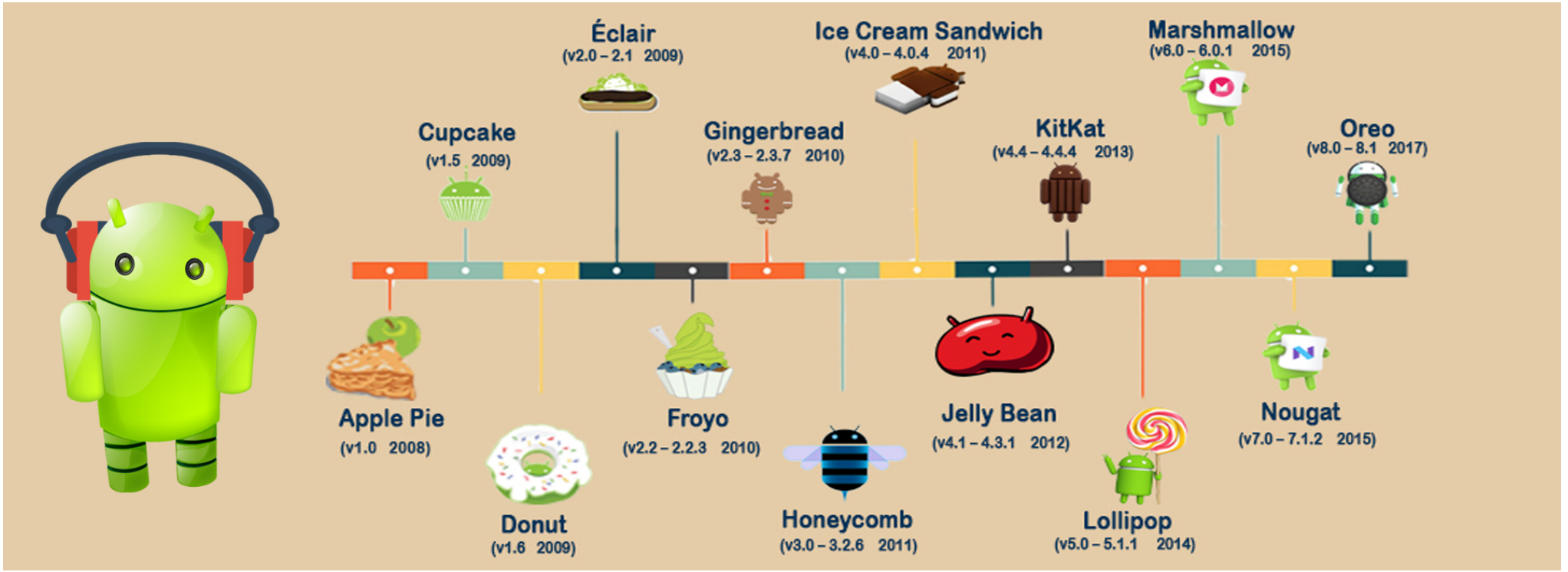

Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
XMPP Service
Libraries
Surface Manager
Media Framework
SQLite
OpenGL|ES
FreeType
WebKit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

Dalvik Virtual Machine (<= 4.4)



Android Runtime (>= 5.0)

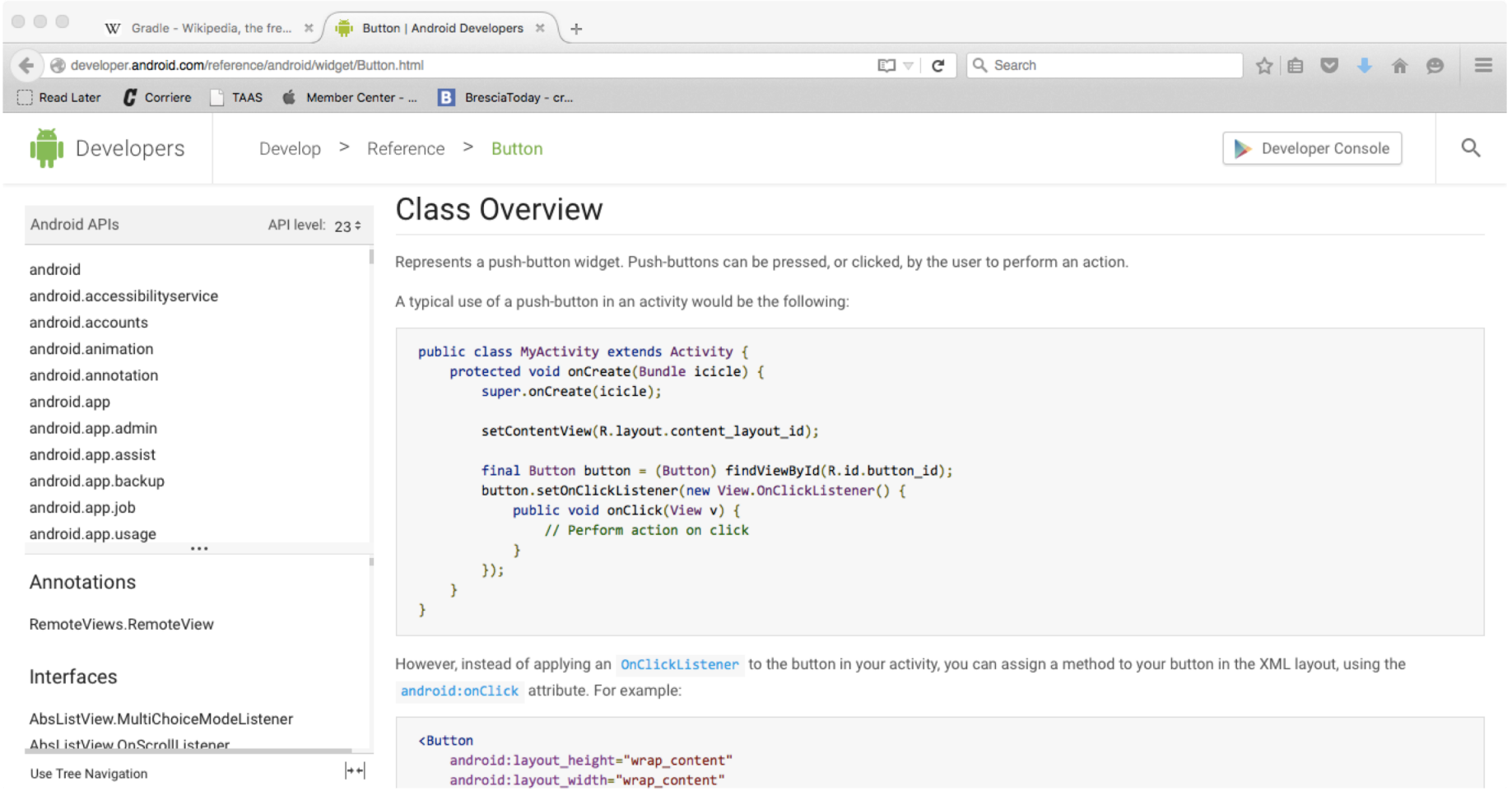



Apple Pie
(v1.0 2008)
Cupcake
(v1.5 2009)
Donut
(v1.6 2009)
Éclair
(v2.0 – 2.1 2009)
Froyo
(v2.2 – 2.2.3 2010)
Gingerbread
(v2.3 – 2.3.7 2010)
Honeycomb
(v3.0 – 3.2.6 2011)
Ice Cream Sandwich
(v4.0 – 4.0.4 2011)
Jelly Bean
(v4.1 – 4.3.1 2012)
KitKat
(v4.4 – 4.4.4 2013)
Lollipop
(v5.0 – 5.1.1 2014)
Marshmallow
(v6.0 – 6.0.1 2015)
Nougat
(v7.0 – 7.1.2 2015)
Oreo
(v8.0 – 8.1 2017)

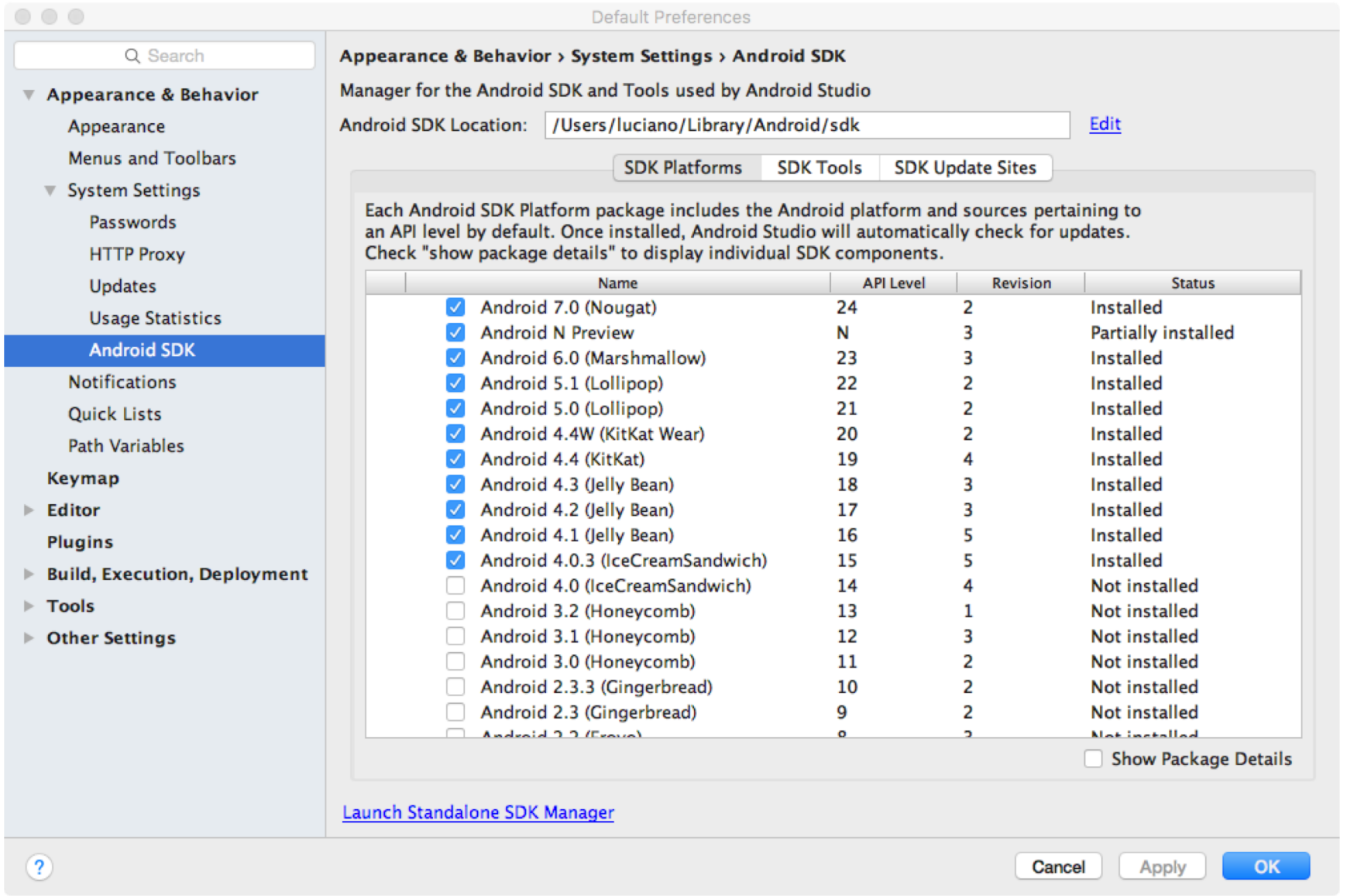
August 6, 2018

Set up

- Set up your development environment
  - Android Studio
    - Uses Gradle
- Set up AVDs and devices for testing
  - Create Android Virtual Devices (AVDs)
  - Connect real devices
  - Use external emulators (Genymotion)


Gradle - Wikipedia, the fre...
Button | Android Developers
developer.android.com/reference/android/widget/Button.html
Search
Read Later
Corriere
TAAS
Member Center - ...
BresciaToday - cr...
Developers
Develop > Reference > Button
Developer Console
Android APIs
API level: 23
android
android.accessibilityservice
android.accounts
android.animation
android.annotation
android.app
android.app.admin
android.app.assist
android.app.backup
android.app.job
android.app.usage
Annotations
RemoteViews.RemoteView
Interfaces
AbsListView.MultiChoiceModeListener
AbsListView.OnScrollListener
Use Tree Navigation
Class Overview
Represents a push-button widget. Push-buttons can be pressed, or clicked, by the user to perform an action.
A typical use of a push-button in an activity would be the following:
public class MyActivity extends Activity {
protected void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.content_layout_id);
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
// Perform action on click
}
});
}
}
However, instead of applying an OnClickListener to the button in your activity, you can assign a method to your button in the XML layout, using the android:onClick attribute. For example:
<Button
android:layout_height="wrap_content"
android:layout_width="wrap_content"

SDK Manager


Default Preferences
Search
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Updates
Usage Statistics
Android SDK
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Tools
Other Settings
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: /Users/luciano/Library/Android/sdk
Edit
SDK Platforms
SDK Tools
SDK Update Sites
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.
Name
API Level
Revision
Status
Android 7.0 (Nougat) 24 2 Installed
Android N Preview N 3 Partially installed
Android 6.0 (Marshmallow) 23 3 Installed
Android 5.1 (Lollipop) 22 2 Installed
Android 5.0 (Lollipop) 21 2 Installed
Android 4.4W (KitKat Wear) 20 2 Installed
Android 4.4 (KitKat) 19 4 Installed
Android 4.3 (Jelly Bean) 18 3 Installed
Android 4.2 (Jelly Bean) 17 3 Installed
Android 4.1 (Jelly Bean) 16 5 Installed
Android 4.0.3 (IceCreamSandwich) 15 5 Installed
Android 4.0 (IceCreamSandwich) 14 4 Not installed
Android 3.2 (Honeycomb) 13 1 Not installed
Android 3.1 (Honeycomb) 12 3 Not installed
Android 3.0 (Honeycomb) 11 2 Not installed
Android 2.3.3 (Gingerbread) 10 2 Not installed
Android 2.3 (Gingerbread) 9 2 Not installed
Show Package Details
Launch Standalone SDK Manager
Cancel
Apply
OK

Development

- Create your application
  - Source code, resource files, and Android manifest file
- Build and run your application
- Test your application
- Publish your application

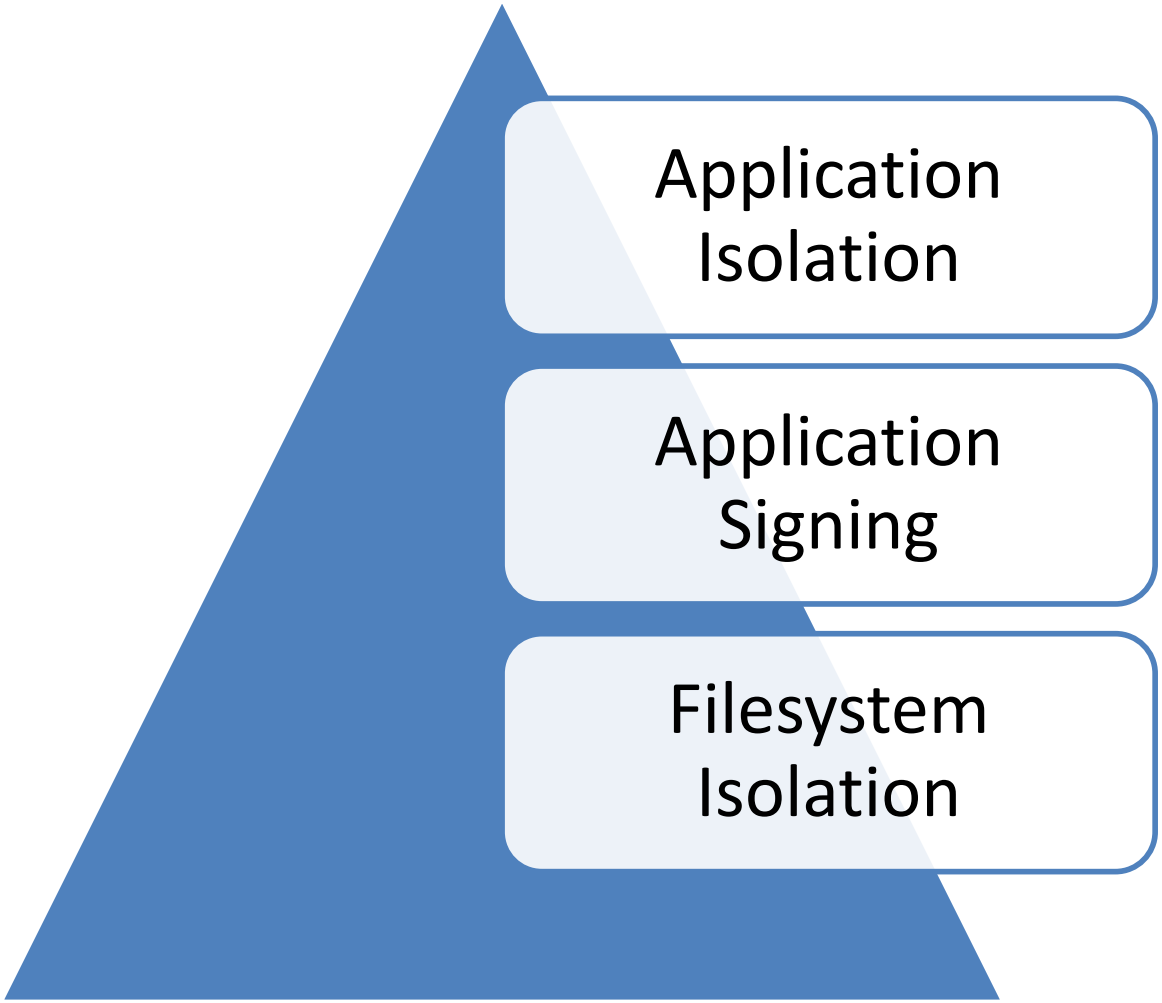
Packaging

- An application is a single APK (application package) file
- An APK file roughly has three main components
  - Executable code: all your compiled Java source code
  - Resources: everything that is not code (images, audio/video clips, XML files describing layouts, language packs, and so on)
  - Native libraries: e.g. C/C++ libraries
  - Manifest

Manifest

- Permissions the application requires
- Minimum API Level required by the application
- Hardware and software features used or required by the application
- Libraries the application needs to be linked with
- … and more

Android Security Model



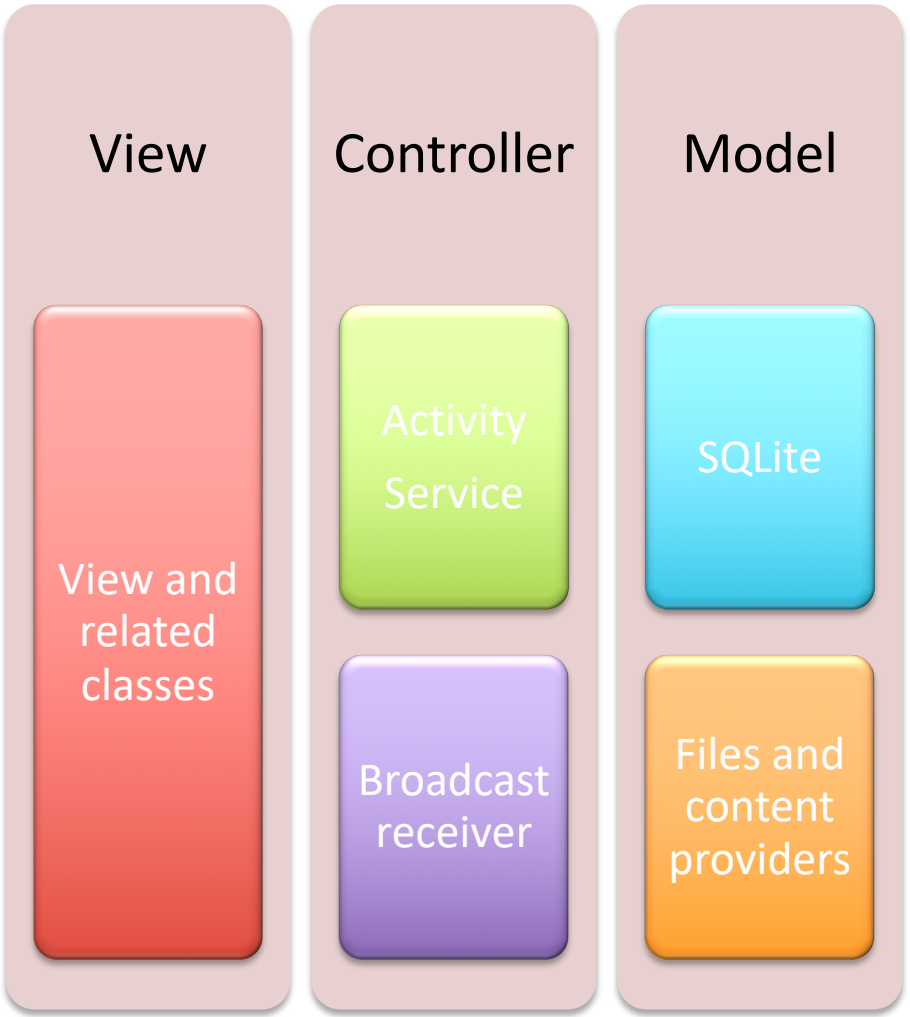
Security model

- Every application runs in its own Linux process
- Each process has its own virtual machine (VM)
  - An application runs in isolation with respect to other applications
- Different applications cannot access each other's data
  - Permissions required

Key terminology

Activity	Component for composing UI
Service	Component that runs on the background
Intent	Message element that sends actions and data to components
Intent filter	Defines components by setting receiving intents
Broadcast receiver	Receives/responds to a certain broadcast (e.g., low battery)
Content provider	Provides standardized interfaces for sharing data among applications
Notification	Notifies events to users

Android through MVC



Applications and activities

- Any application can start another application's component
- An application cannot directly activate a component from another application
  - To activate a component in another application, you must deliver a message to the system that specifies your intent to start a particular component
  - The system then activates the component for you

Starting activities

- The MAIN/LAUNCHER intent filter identifies the app's starting activity
- Nothing forces each app to have one and only one starting activity
- The launcher screen would have different icons for the same program

Android studio

strings.xml

```
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

. . . eventually

```
public class Main extends Activity {

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


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

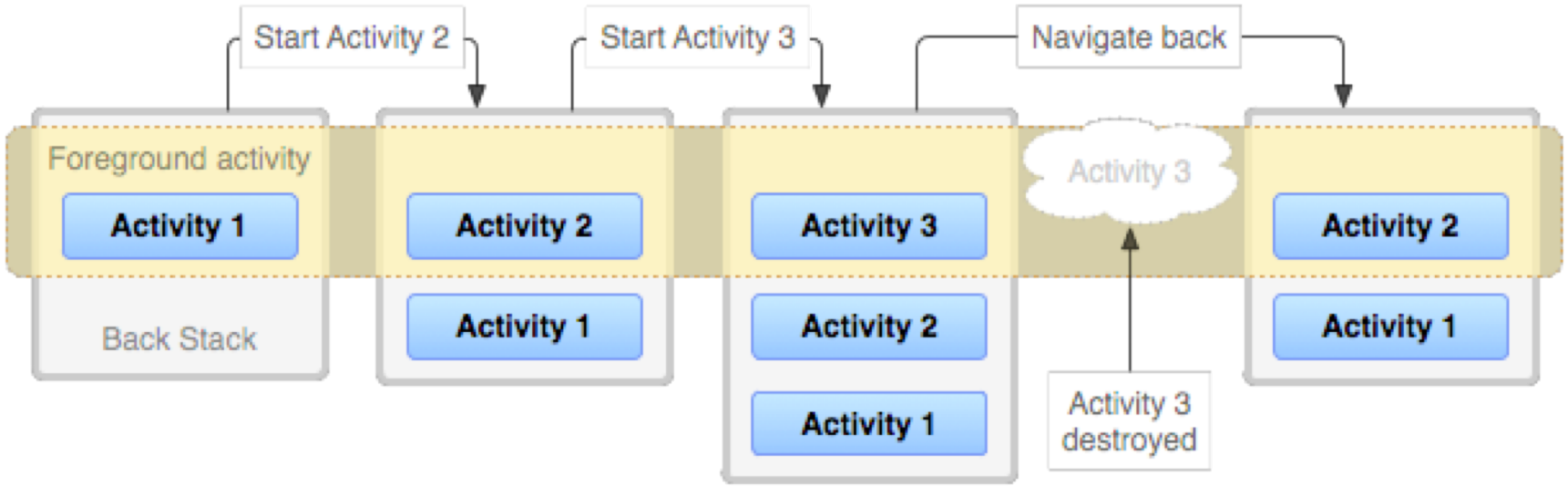
R Class

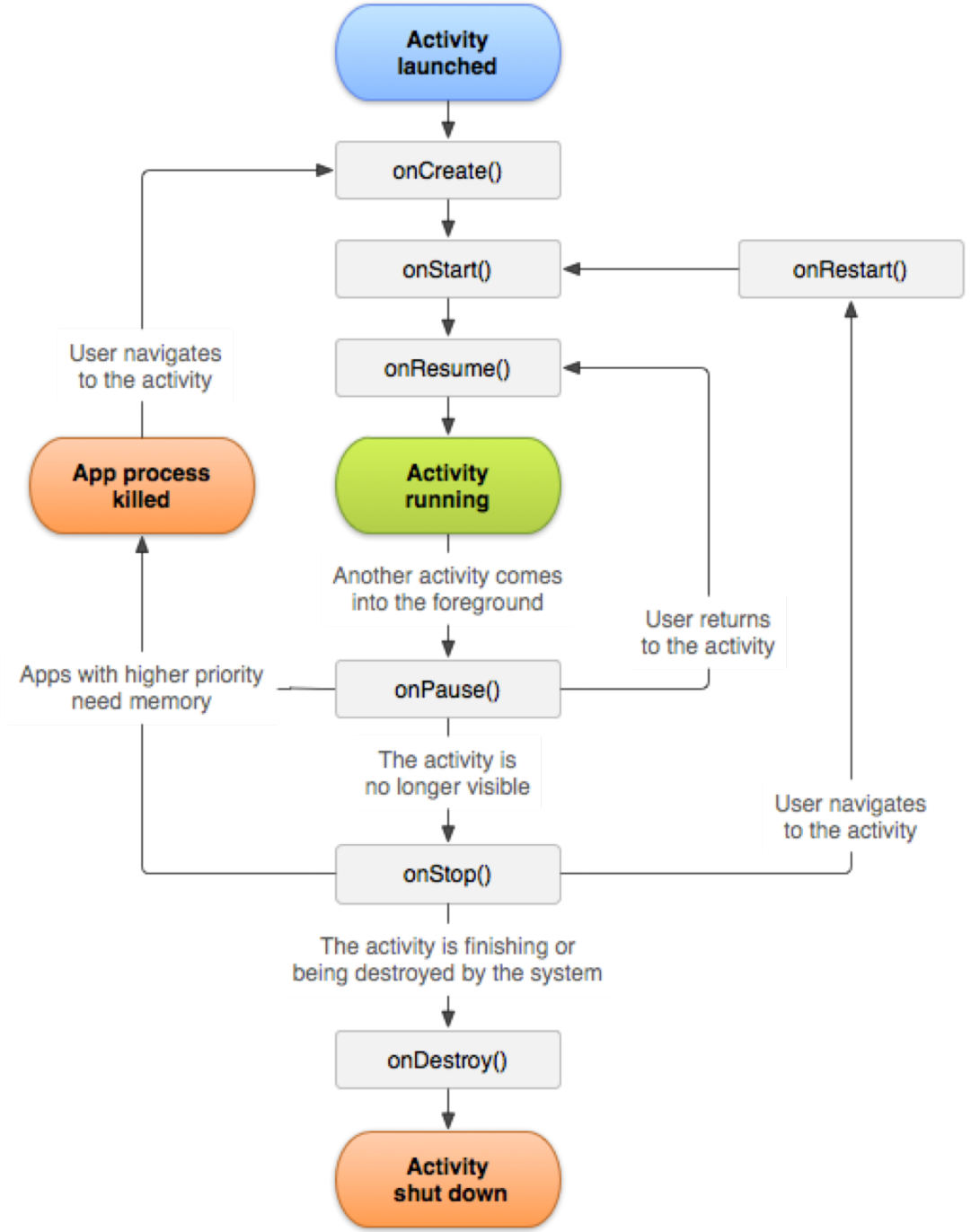
- When your application is compiled
  - aapt generates class R
  - It contains resource ids for all resources in res directory
- There is a subclass for each type of resources
  - For example, R.layout for all layout resources
  - For each resource of a type, there is a static integer
    - For example, R.layout.main
  - This integer is the resource id that you can use to retrieve your resource

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_action_search=0x7f020000;
        public static final int ic_launcher=0x7f020001;
    }
    public static final class id {
        public static final int button=0x7f060000;
        public static final int tf=0x7f060001;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int button_label=0x7f040004;
        public static final int hello=0x7f040001;
        public static final int output=0x7f040003;
        public static final int title_activity_main=0x7f040002;
    }
    public static final class style {
        public static final int AppTheme=0x7f050000;
    }
}
```

Activities

- Each activity can start another activity
  - Each time a new activity starts, the previous activity is stopped
- The system preserves the activities in a LIFO stack
  - The new activity is pushed on top of the stack and takes the focus




Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Three different phases

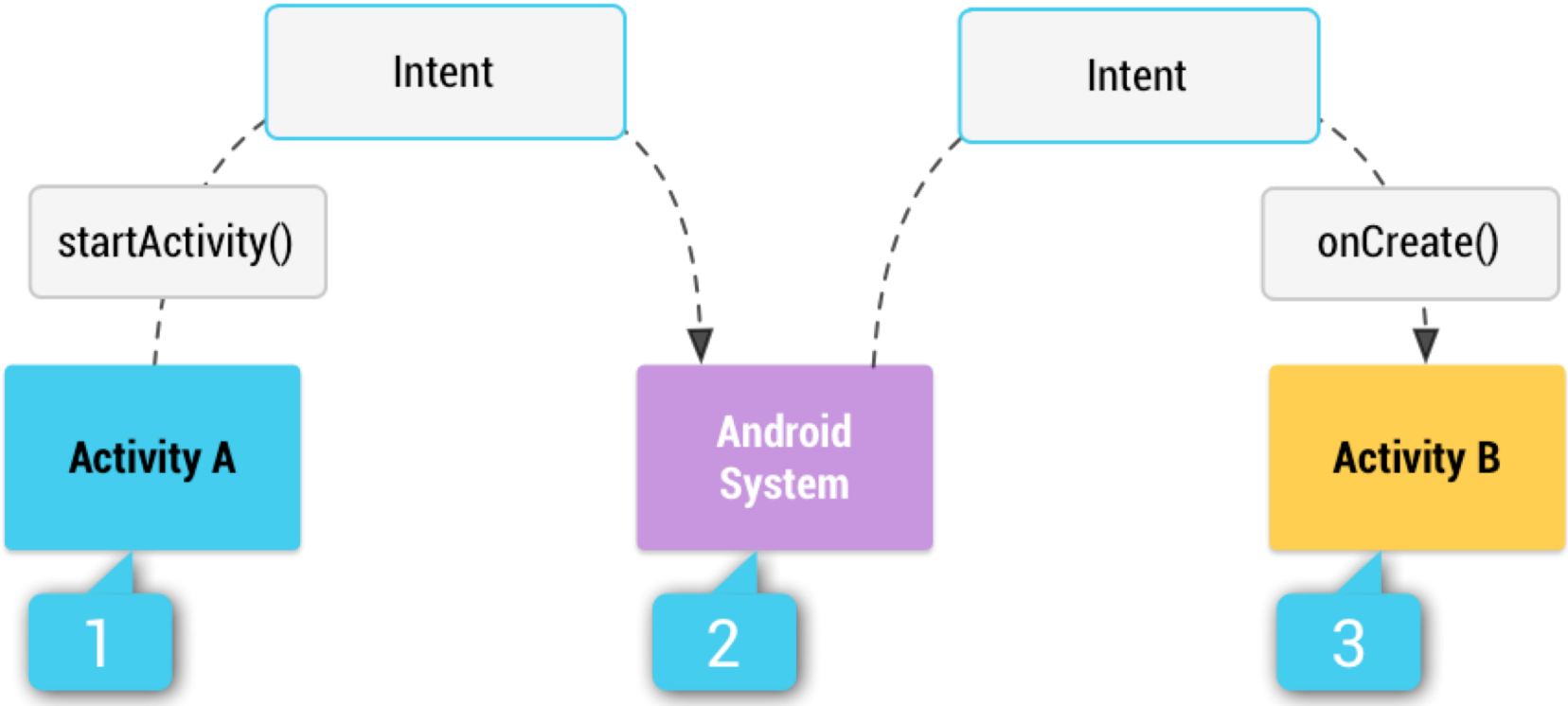
- Entire lifetime
  - Between onCreate() and onDestroy()
  - Setup of global state in onCreate()
  - Release remaining resources in onDestroy()
- Visible lifetime
  - Between onStart() and onStop()
  - Maintain resources that have to be shown to the user
- Foreground lifetime
  - Between onPause() and onResume()

Method	Description	Killable?	Next
`onCreate()`	Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by `onStart()` .	No	`onStart()`
`onRestart()`	Called after your activity has been stopped, prior to it being started again. Always followed by `onStart()`	No	`onStart()`
`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.	No	`onResume()` or `onStop()`
`onResume()`	Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by `onPause()` .	No	`onPause()`
`onPause()`	Called when the system is about to start resuming a previous activity. This is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc. Implementations of this method must be very quick because the next activity will not be resumed until this method returns. Followed by either `onResume()` if the activity returns back to the front, or `onStop()` if it becomes invisible to the user.	**Pre-** `HONEYCOMB`	`onResume()` or `onStop()`
`onStop()`	Called when the activity is no longer visible to the user, because another activity has been resumed and is covering this one. This may happen either because a new activity is being started, an existing one is being brought in front of this one, or this one is being destroyed. Followed by either `onRestart()` if this activity is coming back to interact with the user, or `onDestroy()` if this activity is going away.	**Yes**	`onRestart()` or `onDestroy()`
`onDestroy()`	The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called `finish()` on it, or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the `isFinishing()` method.	**Yes**	*nothing*

Intents

- Allow for late binding between components
  - Activities, Services, Broadcast receivers
- Indirect communication
- Reuse existing and installed applications
- Two possible types
  - Explicit: The target receiver is specified through the Component Name
  - Implicit: The target is specified by data type/names. The system chooses the receiver that matches the request

Simple example



FLAG_ACTIVITY_CLEAR_TOP

- If the activity being launched is already running in the current task
  - Instead of launching a new instance of that activity
  - All of the other activities on top of it will be closed
  - The intent will be delivered to the (now on top) old activity as a new Intent
- Consider a task consisting of activities A, B, C, D
  - If D calls startActivity() with an Intent that resolves to activity B
    - C and D will be finished and B receives the given Intent
    - The stack now is: A, B
- Other flags are available

Constraining instances

```
protected void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    instances++;
    setContentView(R.layout.activity_a2);
    tf = (TextView) findViewById(R.id.instanceCount2);

    final Button button = (Button) findViewById(R.id.button2);
    final Intent intent = new Intent(this, A1.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

    button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            startActivity(intent);
        }
    });
}
```

Intent definition

- Component Name (optional): the receiver of the intent
- Action name: the action to be performed
  - Predefined actions exist, the programmer can define new ones
- Data: information exchanged between caller and callee
- Category: the kind of component that should handle the Intent
- Extras: additional information in the form of key-value pairs
- Flags: additional information to instruct Android how to launch an activity, and how to treat it after execution

Actions

Action Name	Description
ACTION_CALL	Perform a call to someone specified by the data
ACTION_EDIT	Provide explicit editable access to the given data
ACTION_MAIN	Start as a main entry point, does not expect to receive data
ACTION_PICK	Pick an item from the data, returning what was selected
ACTION_VIEW	Display the data to the user
ACTION_SEARCH	Perform a search

Examples of action/data pairs

- ACTION_VIEW content://contacts/people/1
  - Display information about the person whose identifier is "1"
- ACTION_DIAL content://contacts/people/1
  - Display the phone dialer with the person filled in
- ACTION_VIEW tel:123
  - Display the phone dialer with the given number filled in
- ACTION_DIAL tel:123
  - Display the phone dialer with the given number filled in.
- ACTION_EDIT content://contacts/people/1
  - Edit information about the person whose identifier is "1".
- ACTION_VIEW content://contacts/people/
  - Display a list of people, which the user can browse through

Explicit intents

- startActivity(Intent x) starts a new activity, and places it on top of the stack
  - The Intent parameter describes the activity we want to execute
  - Often they do not include any other information
    - It is a way for an application to launch internal activities
- new Intent(Context c, Class c);
  - Context is a wrapper for global information about an application environment
  - Activity subclasses Context

Explicit intents

```
Intent intent=new Intent(this, SndAct.class);
startActivity(intent);
```

```
Intent intent=new Intent();
ComponentName component=new ComponentName(this,SndAct.class);
intent.setComponent(component);
startActivity(intent);
```

Intents with results

- Activities can return results
  - startActivityForResult(Intent int, int reqCode);
  - onActivityResult(int reqCode, int resCode, Intent data)

```
static final int PICK_CONTACT_REQUEST = 1;
// The request code

Intent pickContactIntent = new Intent(Intent.ACTION_PICK,
                                      Uri.parse("content://contacts"));
pickContactIntent.setType(Phone.CONTENT_TYPE);
// Show user only contacts with phone numbers
startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
```

Moreover

```
@Override
protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which
            // contact was selected.

            // Do something with the contact here
        }
    }
}
```

setResult()

- void setResult(int resultCode, Intent data)
- The result is delivered to the caller component only after invoking the finish() method!

Implicit intents

- Target component is not named
  - component name is left blank
- When Intent is launched, Android tries to find an activity that can answer it
  - If at least one is found, then that activity is started!
  - If more than one matching activity is found, the user is prompted to make a choice

Example

- Implicit intents are very useful to re-use code and to launch external applications
  - More than a Component can match the Intent request

```
Intent i = new Intent(android.content.Intent.ACTION_VIEW,
                                    Uri.parse("http://www.polimi.it"));
startActivity(i);
```