Programming for Android - the basics

Tomáš Kypta

@TomasKypta

Android OS

- based on Linux
- open-source
  - https://source.android.com/
- thousands of devices

Versions of Android

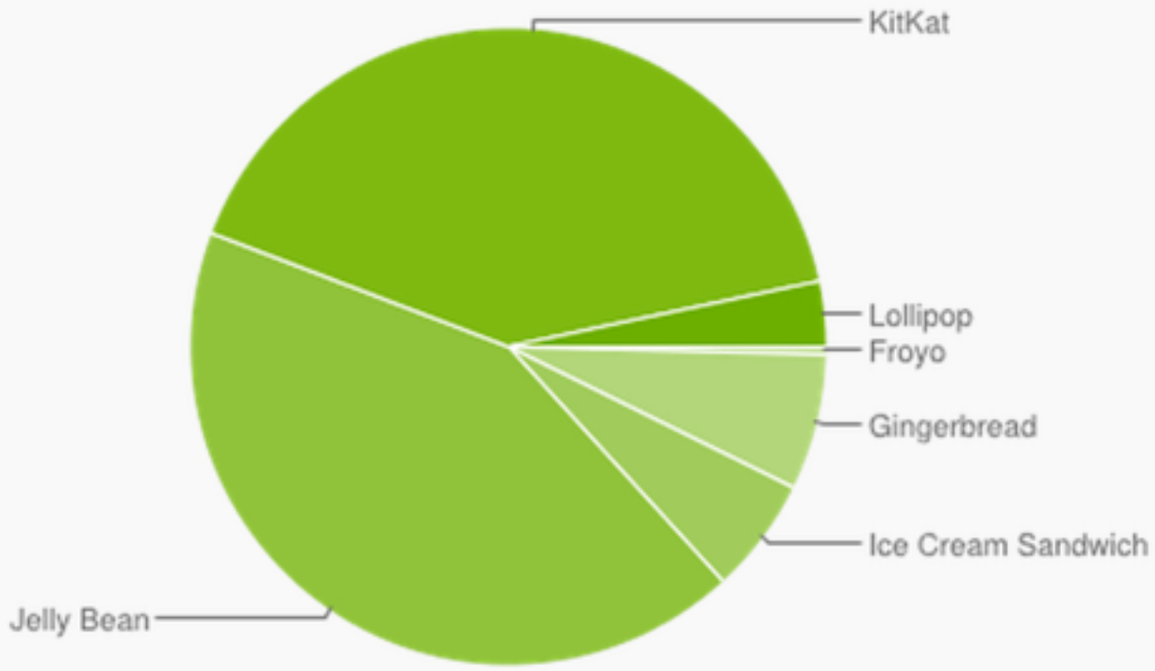
- Sept 2008 Android 1.0
- Dec 2010 Android 2.3 Gingerbread
- Feb 2011 Android 3.0 Honeycomb
- Oct 2011 Android 4.0 Ice Cream Sandwich
- Jul 2012 Android 4.1 Jelly Bean

Versions of Android

- Oct 2013 Android 4.4 KitKat
- Nov 2014 Android 5.0 Lollipop
- Mar 2015 Android 5.1 Lollipop

Versions of Android

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%



Androdi Ecosystem

- Google Play
  - https://play.google.com
  - > 1 400 000 apps
- other stores

Publishing on Google Play

- one-time fee $25
- monetization
    - paid apps
    - in-app billing
        - standard products
        - subscriptions
    - ads

Development for Android

Development for Android

- programming in “Java”
- native apps in C/C++
- Open GL
- Android SDK
- development on Mac OS X, Windows, Linux

Development for Android

- great IDE support
    - Android Studio (preferred)
    - IntelliJ IDEA
    - Eclipse (ADT plugin)
    - other

Development for Android

- no restrictions for using devices for development

Building Android Apps

Hello World

Android Apps

- Java code
- resources
- assets
- AndroidManifest.xml
- included libraries
- build.gradle

Android Building Blocks

- Android apps have 4 component types
  - Activity
  - Service
  - Broadcast Receiver
  - Content Provider

Gradle Build

- build automation tool
- builds upon Ant ant Maven
- DSL based on Groovy = Java + syntactic sugar
- define:
  - project config
  - dependencies
  - build steps
  - …

Android SDK

Android SDK

- build & debug tools
  - android - Android SDK Manager
  - android avd - Android Virtual Device Manager
  - adb - Android Debug Bridge
  - monitor - Android Device Monitor
  - aapt, dx, proguard, zipalign, lint
  - …

Android SDK

- documentation
- samples
- sources
- support libraries

Activity

- a screen of an application

- the only UI component

Activity

- examples
    - list of emails
    - detail of an email
    - email compisition
    - app settings

Service

- no UI
- long term tasks
- services can be started for task execution
- apps can bind to running services

Service

- examples
  - audio player service
  - download service
  - synchronization service

Broadcast Receiver

- broadcasts = system wide messages
- receiver responds to broadcasts
- can be registered:
    A. statically - in AndroidManifest.xml
    B. dynamically - in code
- system or custom messages

Broadcast Receiver

- examples
  - incoming SMS
  - incoming call
  - low battery
  - removed SD card
  - BT device available
  - …

Content Provider

- manages and shares application data
- can use any data storage (db, web, filesystem, …)
- apps can query and modify data via content providers
- r/w permissions can be defined

Content Provider

- examples - system DBs
  - contacts
  - messages
  - call log
  - …

AndroidManifest.xml

- defines parts of the apps
- defines exposed endpoints
- permissions
- declares required features and configurations
- …

Intent

- asynchronous message
- binds components together
  - except Content Provider
- starting activities
- starting services and binding to services
- sending broadcasts

Fragment

- piece of app UI
- not a component
- contained in activities
- introduced to support one app with multiple UIs - for phone and tablet

Fragment

Activity

Activity

- extends **`android.app.Activity`**
- the most common component
- stored in a stack within an app

Transitions between Activities

- via Intent

- we can start an explicit activity

```
Intent intent = new Intent(MainActivity.this, OtherActivity.class);
context.startActivity(intent);
```

- we can start an implicit activity

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://…"));
context.startActivity(intent);
```

Transitions between Activities

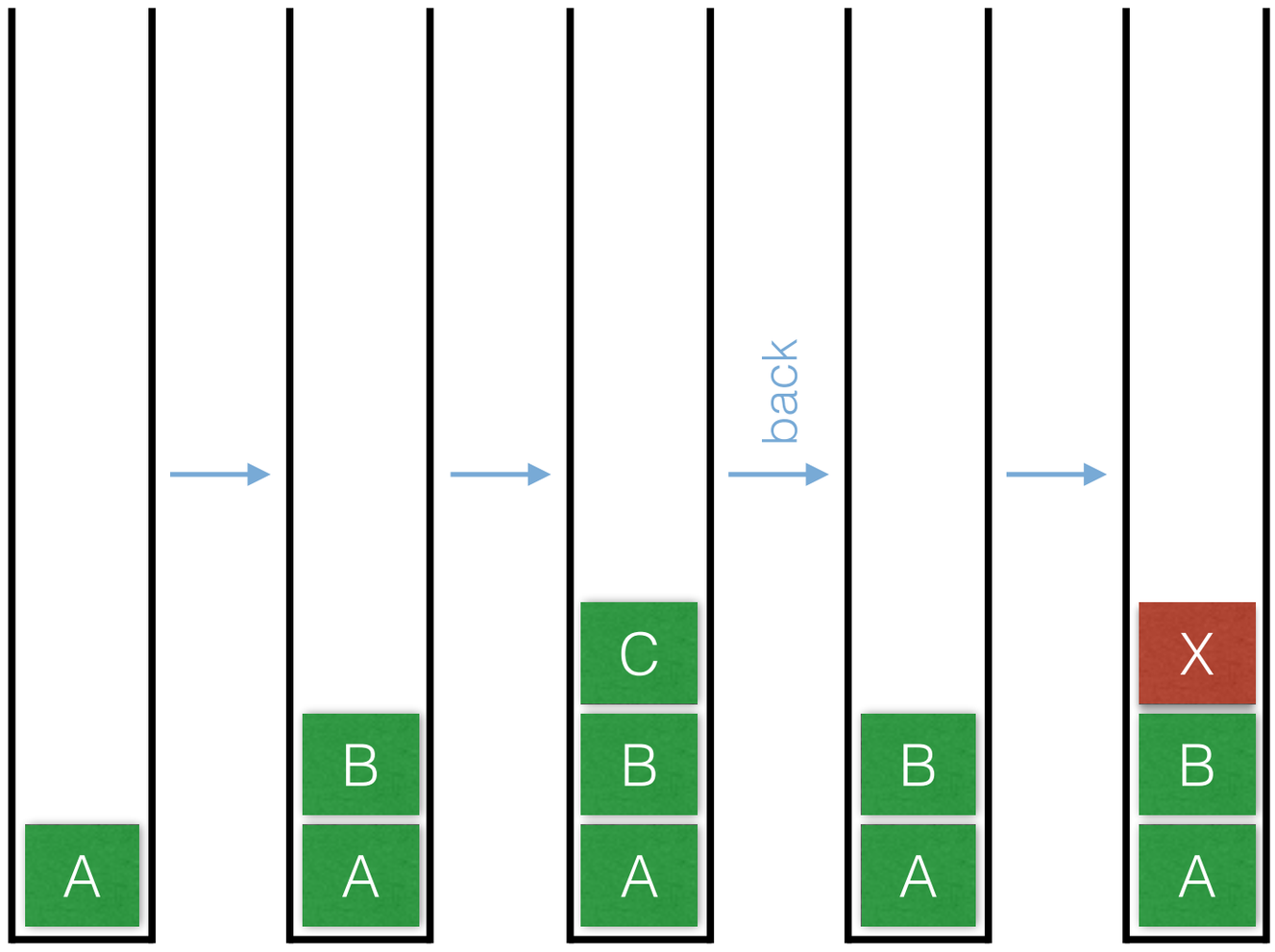
- we can start an activity for a result

```
Intent intent = new Intent(…);
startActivityForResult(intent);
```

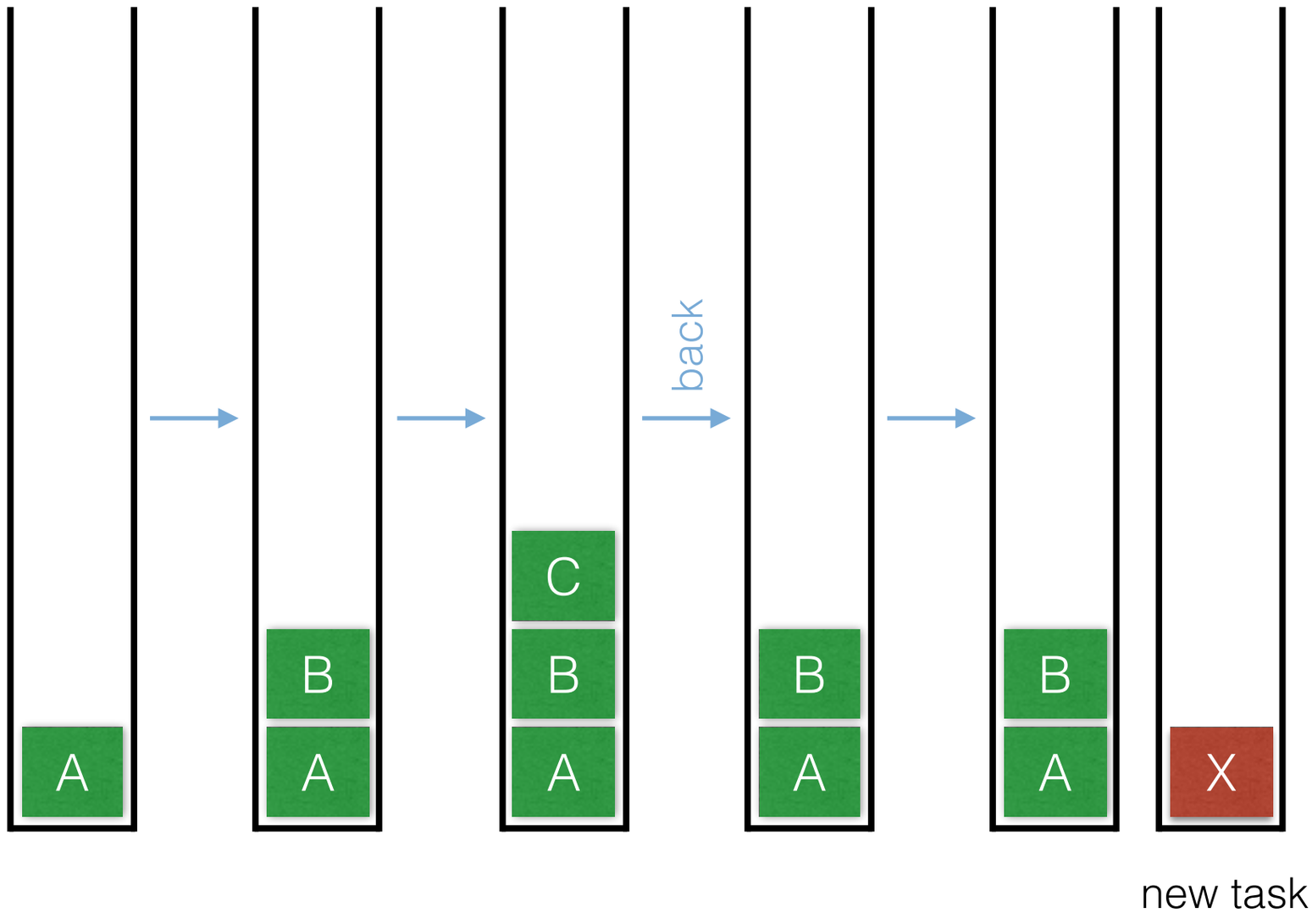
Tasks & Back Stacks

- running application = task = stack of activities
- starting new app starts new task
- transition between app activities adds activities to the stack
- other apps' activities can be added of my task

Tasks & Back Stacks

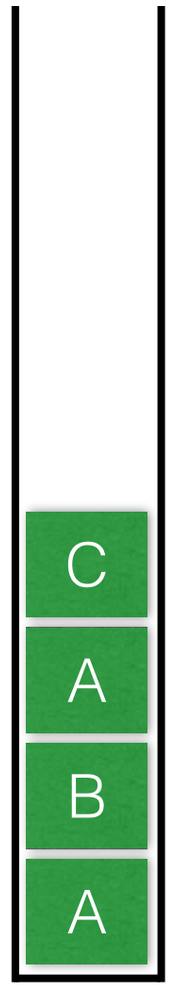


Tasks & Back Stacks



Tasks & Back Stacks

- stack can contain multiple instances of an activity



Activity Lifecycle

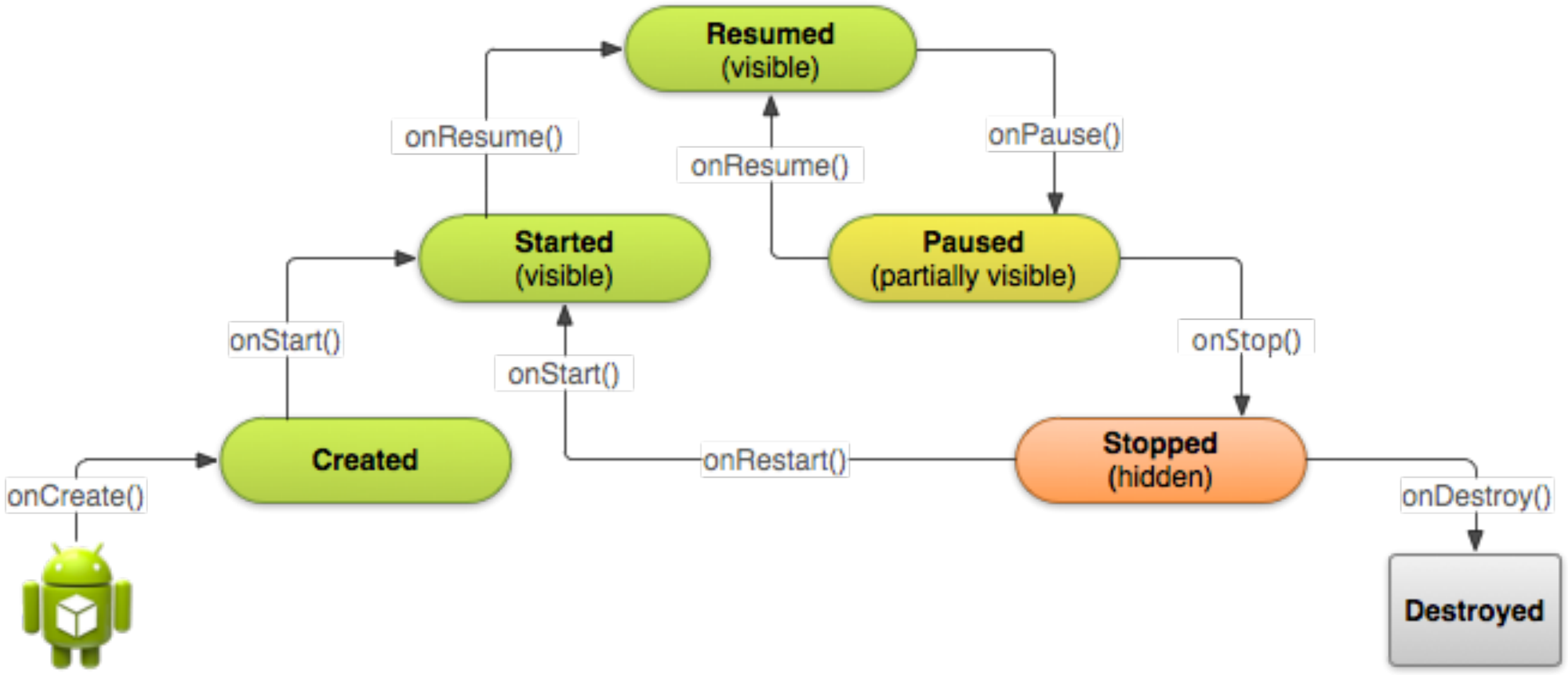
Activity Lifecycle

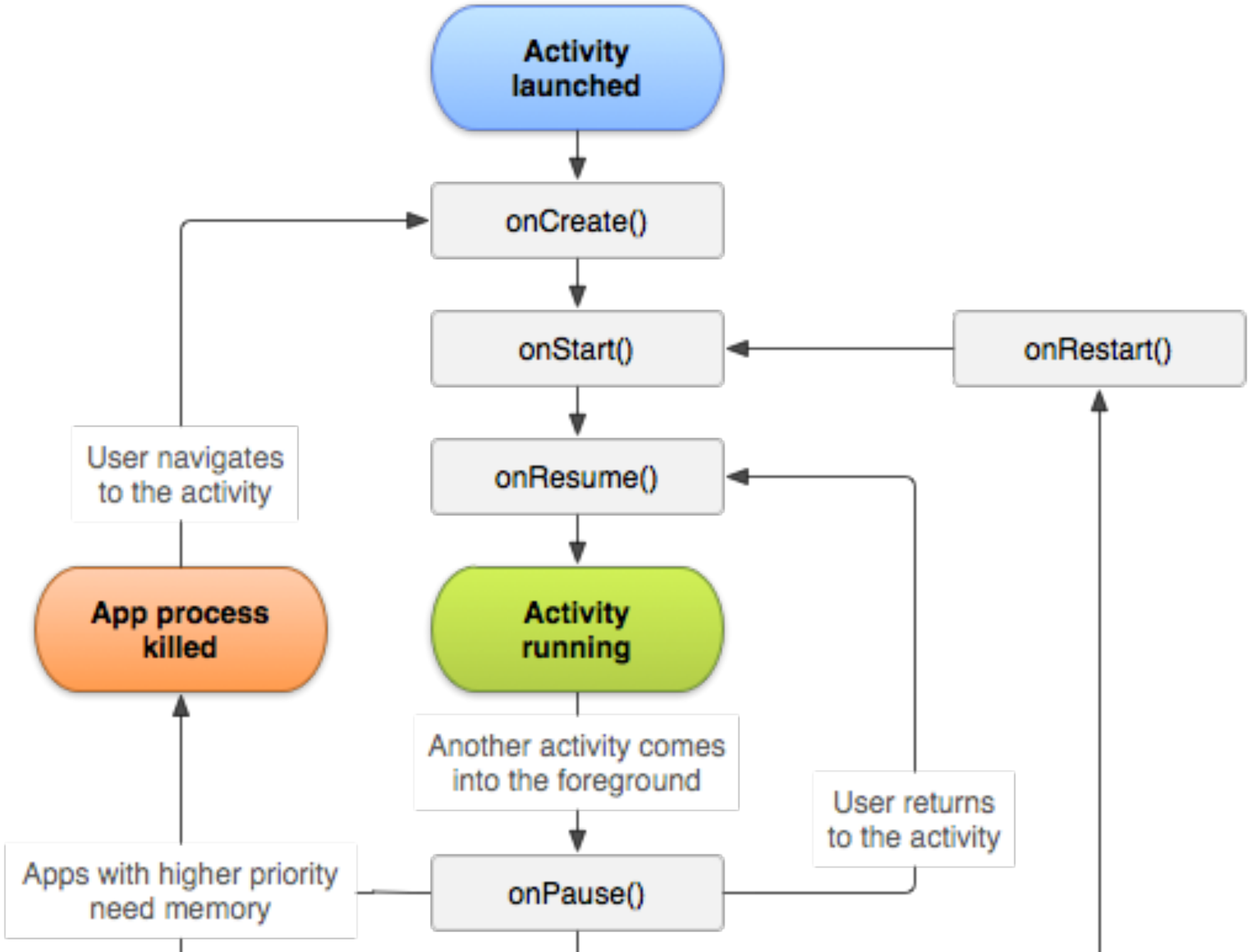
- activity can be in one of the following states
  - foreground
  - visible
  - stopped
  - killed

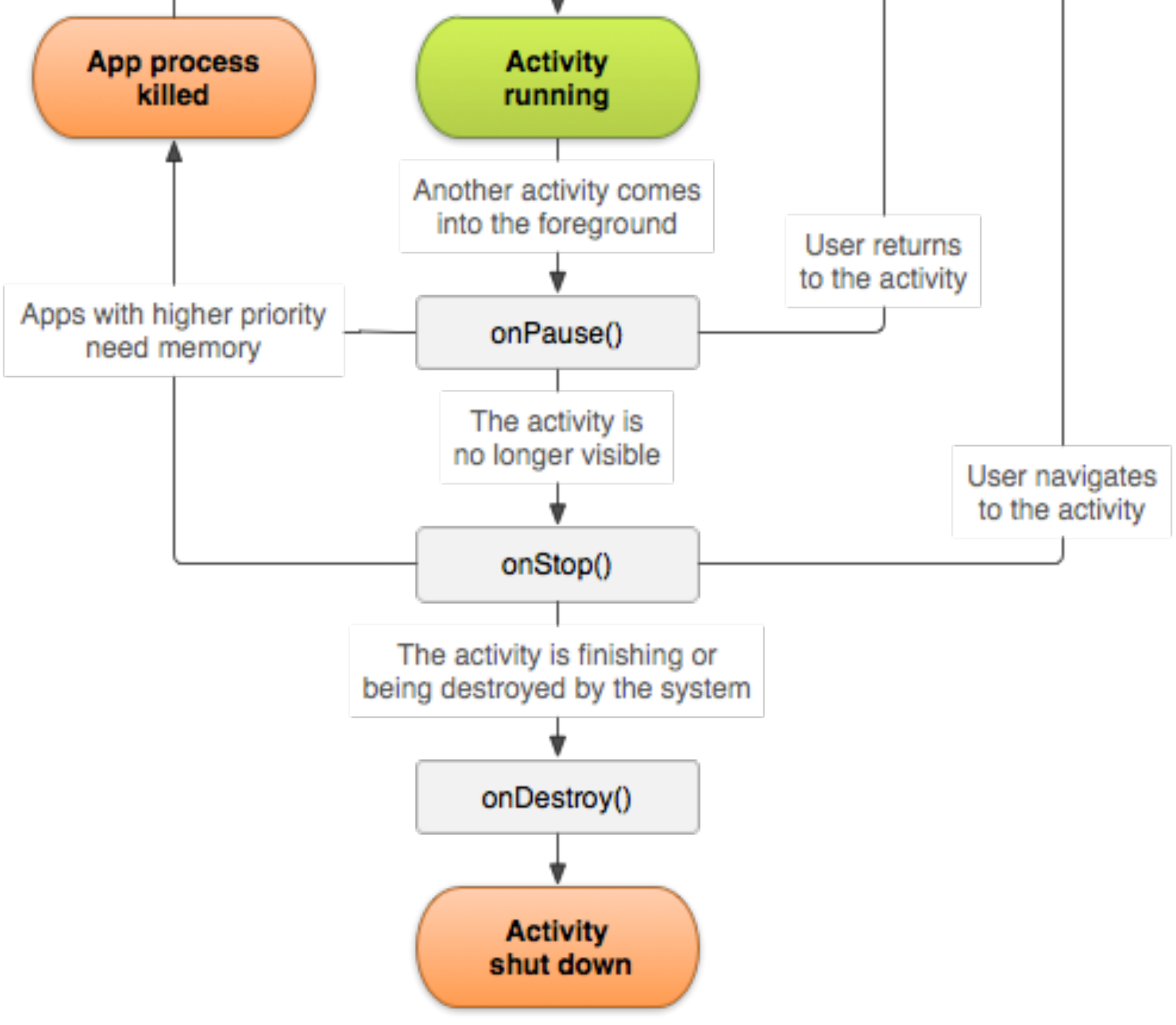
Lifecycle Callbacks

- callbacks are called when activity state changes

Lifecycle Callbacks




Activity launched
onCreate()
onStart()
onRestart()
onResume()
Activity running
Another activity comes into the foreground
User returns to the activity
onPause()
Apps with higher priority need memory
App process killed
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

Lifecycle Callbacks

- callbacks are paired (mostly)

onCreate() - onDestroy()

onRestart() - onStart() - onStop()

onResume() - onPause()

Lifecycle Callbacks

- the most important are

`onCreate()`

- create activity

`onPause()`

- user leaving activity

Configuration Changes

- when configuration changes, activities are destroyed and recreated
    - default behaviour that can be changed
- it’s important to handle config changes properly
    - save activity state

Saving Activity State

- save

  `onSaveInstanceState(Bundle)`

- restore

  `onCreate(Bundle)`

  `onRestoreInstanceState(Bundle)`

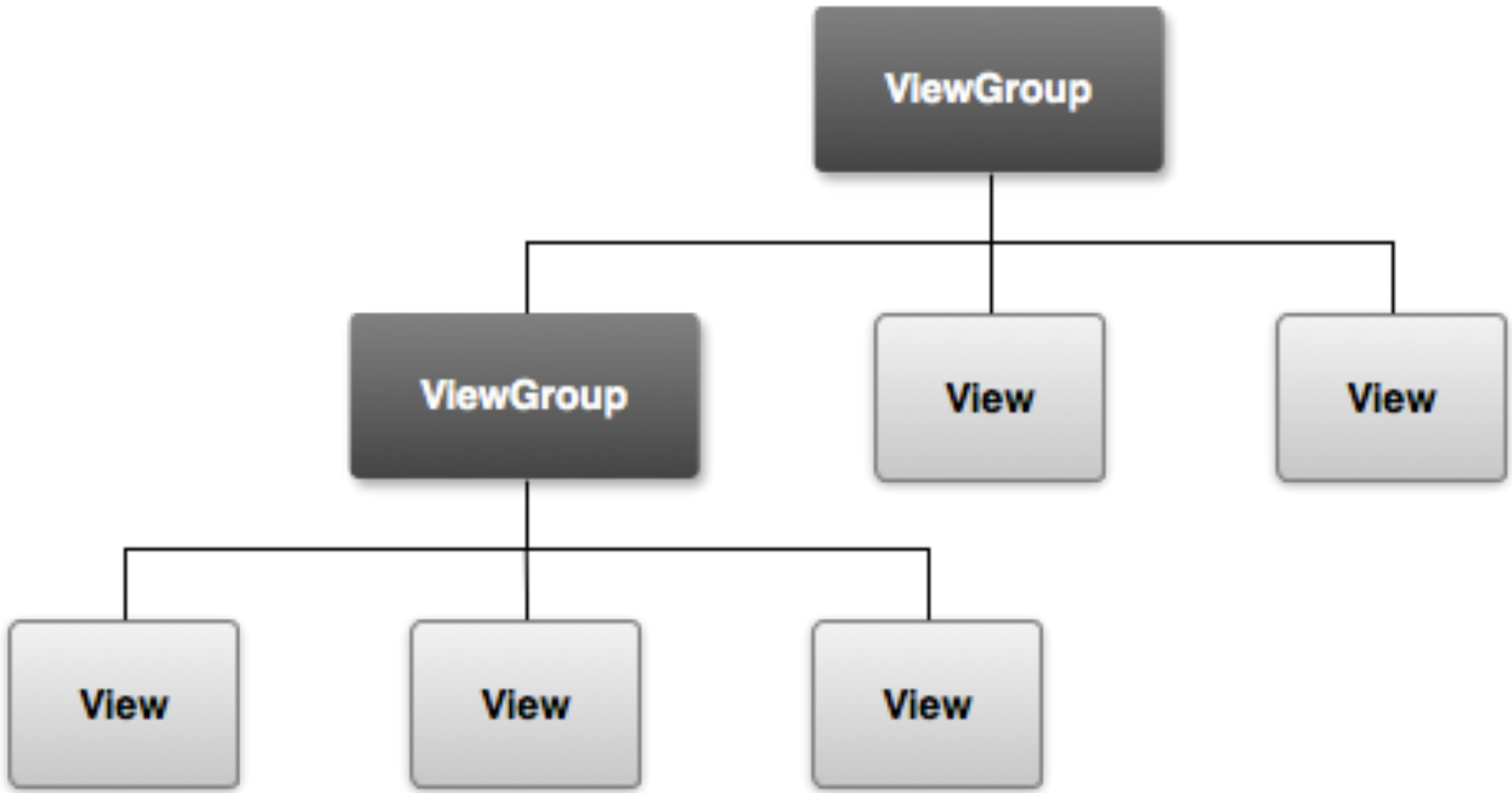
onPause() vs. onSaveInstanceState()

- **`onSaveInstanceState()`**
  - record per-instance state of the activity
- **`onPause()`**
  - store global persistent data

User Interface

User Interface

- defined by a hierarchy of views



User Interface

- ViewGroup
  - invisible containers

LinearLayout

RelativeLayout

FrameLayout

GridLayout

…

User Interface

- View
  - UI widget

Button

TextView

EditText

WebView

…

User Interface

- lists
  - special ViewGroups
  - display data inserted by **`Adapter`**

    `ListView`

    `GridView`

    `Spinner`

    …

    `RecyclerView`

Resources

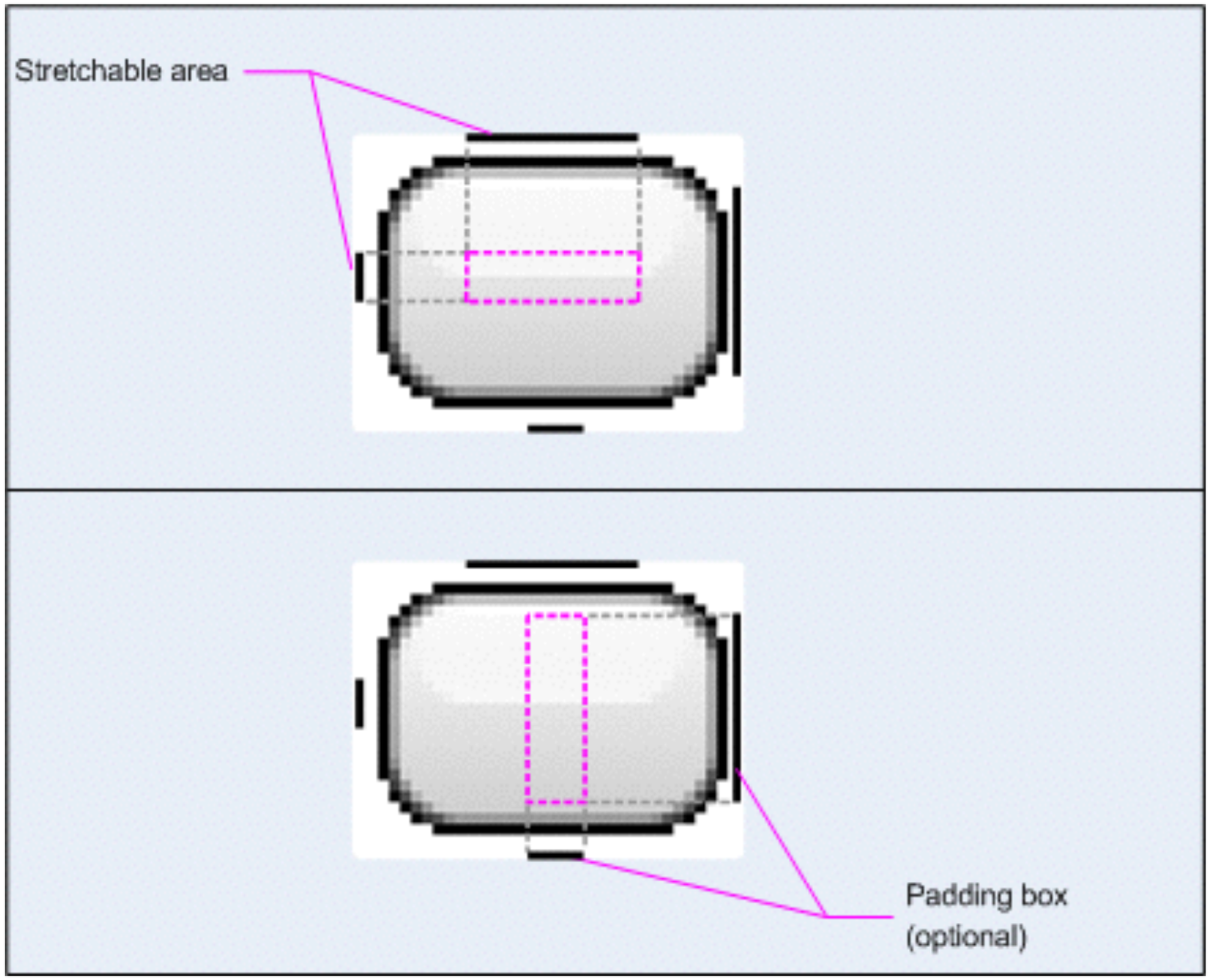
Resources

- many types
  - drawable
  - layout
  - string
  - menu
  - color
  - dimension
  - animation
  - arrays
  - ids
  - xml
  - raw
  - …

Drawable Resources

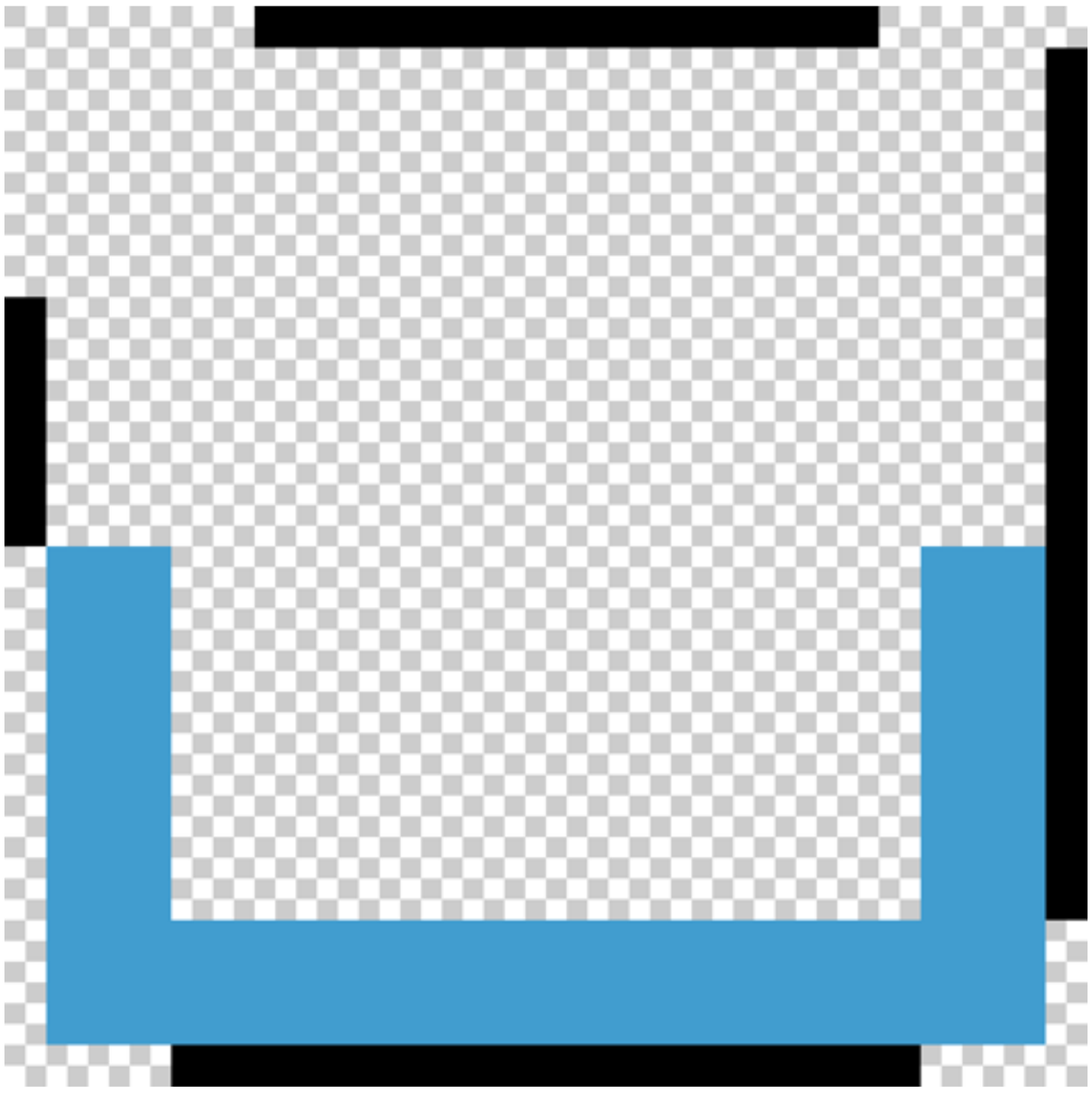
- bitmap
- 9-patch png
- state list
- layer list
- shape
- vector (since Lollipop)

9-patch



9-patch

9-patch



Resources

- in code accessed via generated R class
- each resource identified by a generated integer

```
view.findViewById(R.id.txt_name)

txtName.setText(R.string.txt_name_label)
```

Resources

- one resource can be accessed in another (xml based) resource

```
android:layout_below="@id/txt_name"
android:text="@string/txt_name_label"
```

String Resources

- parametrized strings

```
<string name="welcome_messages">Hello, %1$s! You have %2$d
new messages.</string>
```

- string arrays

```
<string-array name="days">
    <item>Monday</item>
    <item>Tuesday</item>
    <item>Wednesday</item>
</string-array>
```

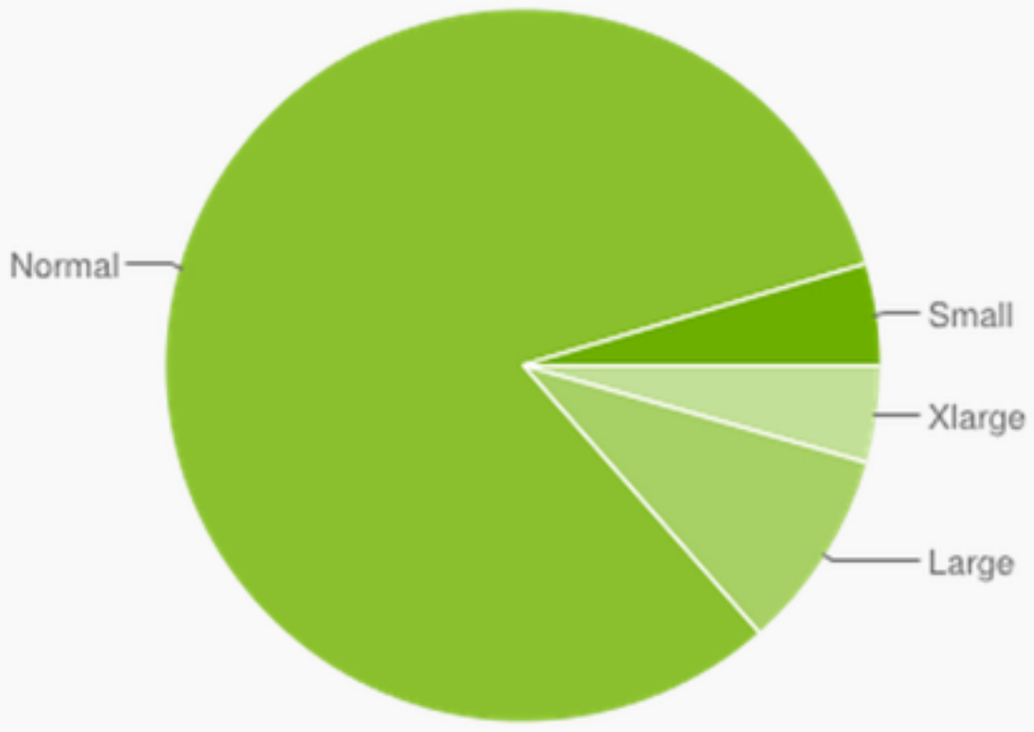
String Resources

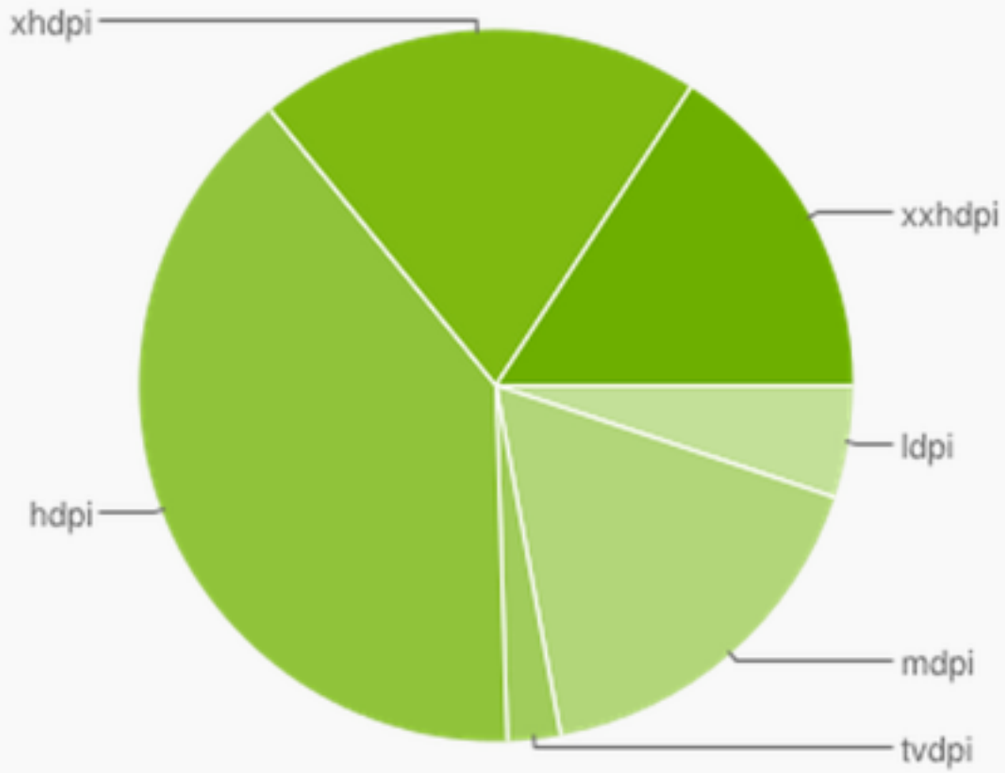
- quantity strings

```
<plurals name="selected">
    <item quantity="one">%s vybraný</item>
    <item quantity="two">%s vybrané</item>
    <item quantity="few">%s vybrané</item>
    <item quantity="many">%s vybraných</item>
    <item quantity="other">%s vybraných</item>
</plurals>
```

Screen Sizes and Densities

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	**Total**
Small	4.6%						4.6%
Normal		8.4%	0.1%	38.7%	18.9%	15.8%	81.9%
Large	0.5%	5.1%	2.3%	0.6%	0.6%		9.1%
Xlarge		3.5%		0.3%	0.6%		4.4%
Total	5.1%	17.0%	2.4%	39.6%	20.1%	15.8%	





Screen Densities

- ldpi (low) ~120dpi
- mdpi (medium) ~160dpi
- hdpi (high) ~240dpi
- xhdpi (extra-high) ~320dpi
- xxhdpi (extra-extra-high) ~480dpi
- xxxhdpi (extra-extra-extra-high) ~640dpi

Handling Screen Sizes and Densities

- several versions of resources
  - best version selected by current configuration in runtime
- independent resource units
  - dp - density-independent pixel
  - sp - scale-independent pixel (for fonts only)

Resource Qualifiers

- way to distinguish versions of resources

- suffixes for resource folders

Resource Qualifiers

- density
  - -mdpi, -hdpi, -xhdpi, …
- language
  - -cs, -en, -en-rGB
- screen orientation
  - -port, -land
- screen size
  - -small, -normal, -large

Resource Qualifiers

- version
  - -v11, -v14
- screen size (since Android 3.2)
  - -w600dp, -h720dp, -sw640dp
- many others
- combinations
  - -hdpi-v11

Localization

- via string resources

- never hardcode strings!!!

Adapter

Adapter

- provides data for the list
- is responsible for creation of item views

Adapter

- most critical is the item creation
- always use convert views to recycle the views
- use “view holder” design pattern or custom view to avoid inefficiencies with **`findViewById()`**

RecyclerView

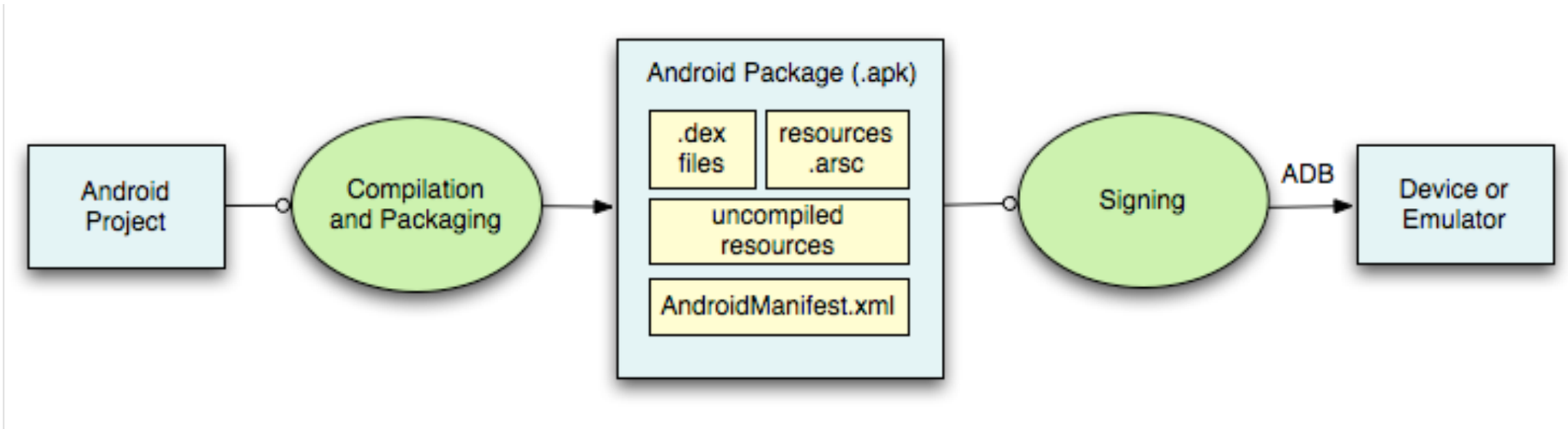
- new flexible view for a limited window into a large data set

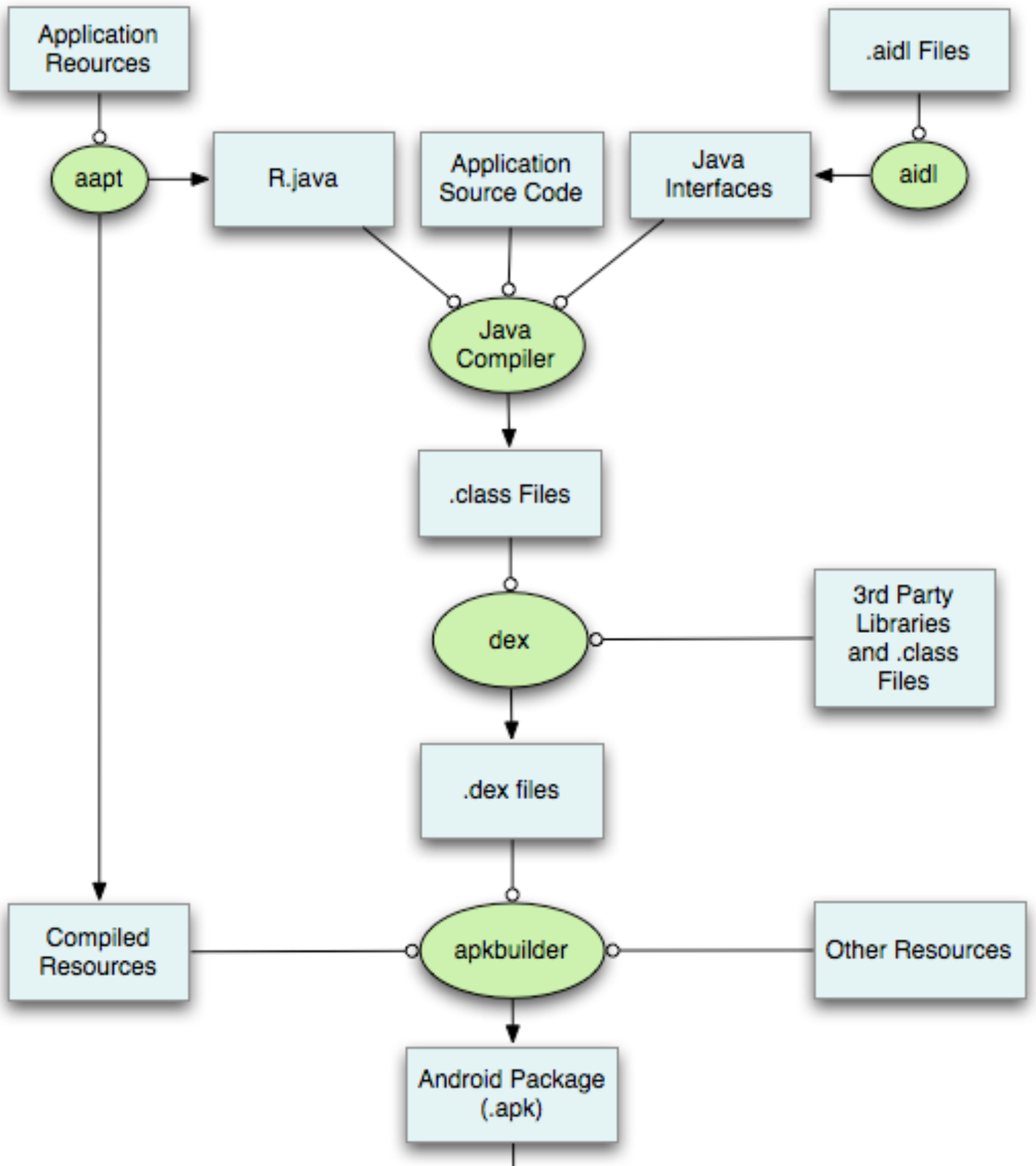
- **`RecyclerView`**

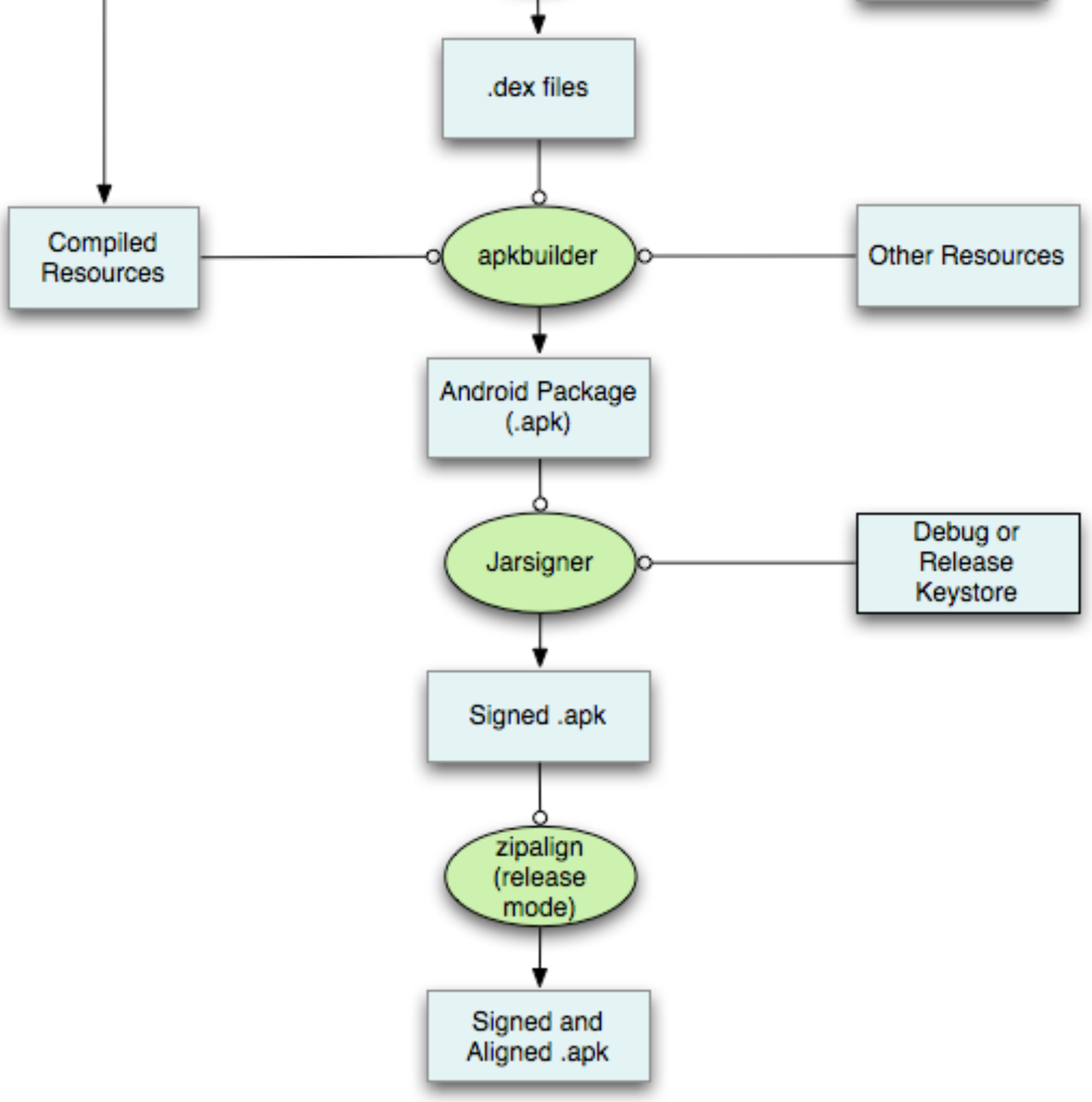
- uses **`RecyclerView.Adapter`**

Build System

Build


Android Project
Compilation and Packaging
Android Package (.apk)
.dex files
resources .arsc
uncompiled resources
AndroidManifest.xml
Signing
ADB
Device or Emulator


Application Reources
.aidl Files
aapt
R.java
Application Source Code
Java Interfaces
aidl
Java Compiler
.class Files
dex
3rd Party Libraries and .class Files
.dex files
Compiled Resources
apkbuilder
Other Resources
Android Package (.apk)


.dex files
Compiled Resources
apkbuilder
Other Resources
Android Package (.apk)
Jarsigner
Debug or Release Keystore
Signed .apk
zipalign (release mode)
Signed and Aligned .apk

build.gradle

build.gradle

- huge flexibility of build customization
  - manifest entries
  - build types & product flavors
  - signing configuration
  - dependencies
  - multi-project setup
  - testing

build.gradle

```
dependencies {
    compile files('libs/foo.jar')
    compile 'com.google.guava:guava:18.0'
}
```

Fragmentation

UI Fragmentation

- done via resources and fragments

Version Fragmentation

- compile sdk version
  - version the build tools are using to compile & build
- target sdk version
  - version our app is created for
  - might affect styling in runtime
- min sdk version
  - min version our app supports and can be installed on

Version Fragmentation

- handling versions in code

```
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.GINGERBREAD) {
    // code for Android < 2.3
}
```

Version Fragmentation

```
private boolean functionalitySupported = false;
static {
    try {
        checkFunctionalitySupported();
    } catch (NoClassDefFoundError e) {
        functionalitySupported = false;
    }
}

private static void checkFunctionalitySupported()
        throws NoClassDefFoundError {
    functionalitySupported = android.app.Fragment.class != null;
}
```

Threads

Threads

- main thread = UI thread
- never block the UI thread!!!
- use worker threads for time consuming operations
  - networking, db, filesystem, …
- UI toolkit is not thread safe
  - never manipulate UI from a worker thread!!!

Threads

- use

  `java.lang.Thread`

  `AsyncTask<Params, Progress, Result>`

  - parallel/serial behaviour changed in some versions of Android

  - workaround with running on specific `Executor`

`task.executeOnExecutor(Executor, Params…)`

Threads, Leaks & Crashes

- Android development far more constrained than backend/desktop development
- Android components are often recreated
- take special care when working with threads and AsyncTasks
  - leaking activities
  - accessing invalid activities

Logging

java.util.logging.Logger

android.util.Log

Toast

- simple non-modal information
- displayed for a short period of time
- doesn’t have user focus

Preferences

Preferences

- stored in a file

```
SharedPreferences prefs = PreferenceManager
    .getDefaultSharedPreferences(context);
SharedPreferences prefs =
    config.getSharedPreferences(PREFS_FILE_NAME,
    Activity.MODE_PRIVATE);
```

Preferences

- get

```
int storedValue = prefs.getInt(SOME_KEY, defaultValue);
```

- set

```
SharedPreferences.Editor editor = prefs.edit();
editor.putInt(SOME_KEY, storedValue);
editor.commit();
```

- set asynchronously

```
SharedPreferences.Editor editor = prefs.edit();
editor.putInt(SOME_KEY, storedValue);
editor.apply();
```

Sources

- http://developer.android.com
- http://android-developers.blogspot.com
- http://source.android.com
- http://stackoverflow.com
- http://youtube.com/androiddevelopers

Questions?