Basics, Activity

Android Lecture 2

2023

Lukáš Prokop
Simona Kurňavová

Finishing subject

https://d3s.mff.cuni.cz/teaching/nprg056/

Agenda

- Android UI – overview
- Project structure
  - Gradle
  - Source code
  - Resources
- Android components
  - Android Manifest
  - Service
  - Content provider
  - Broadcast receiver
  - Intent
- Activity

Android UI - overview

Launcher

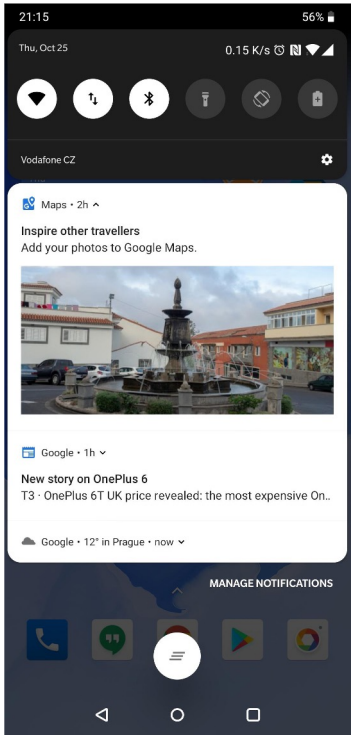
Status bar

App widget

App widget vs. widget
Widget is UI item (button, checkbox, etc)

System bar (or gesture navigation)

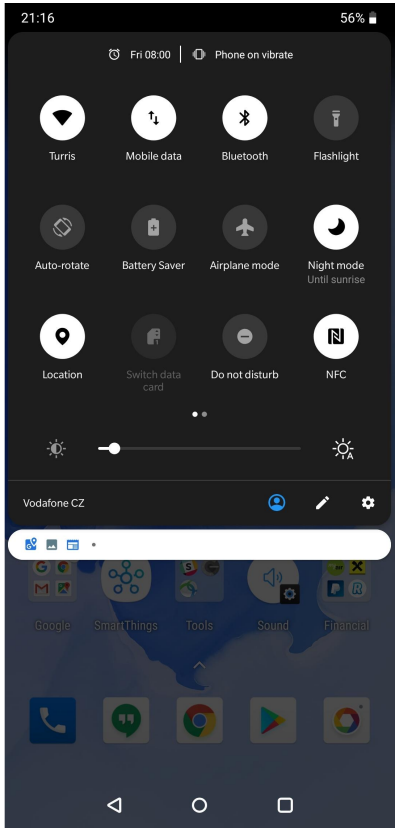
Notifications



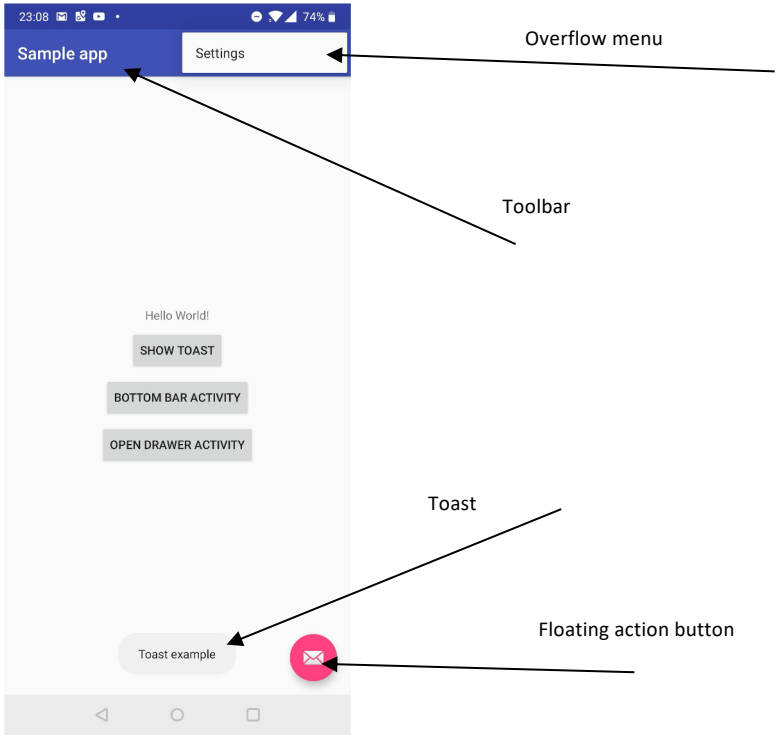
- Inform user (email, progress, etc)
- Actions in notifications since API-16
- Can be visible on lock screen
- Optional sound, vibration, LED light

Quick settings

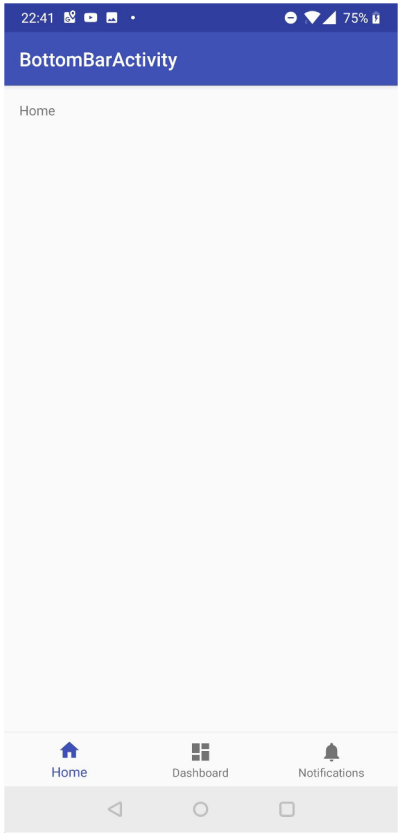
- Since API-16 part of AOSP
- Since API-24 custom actions



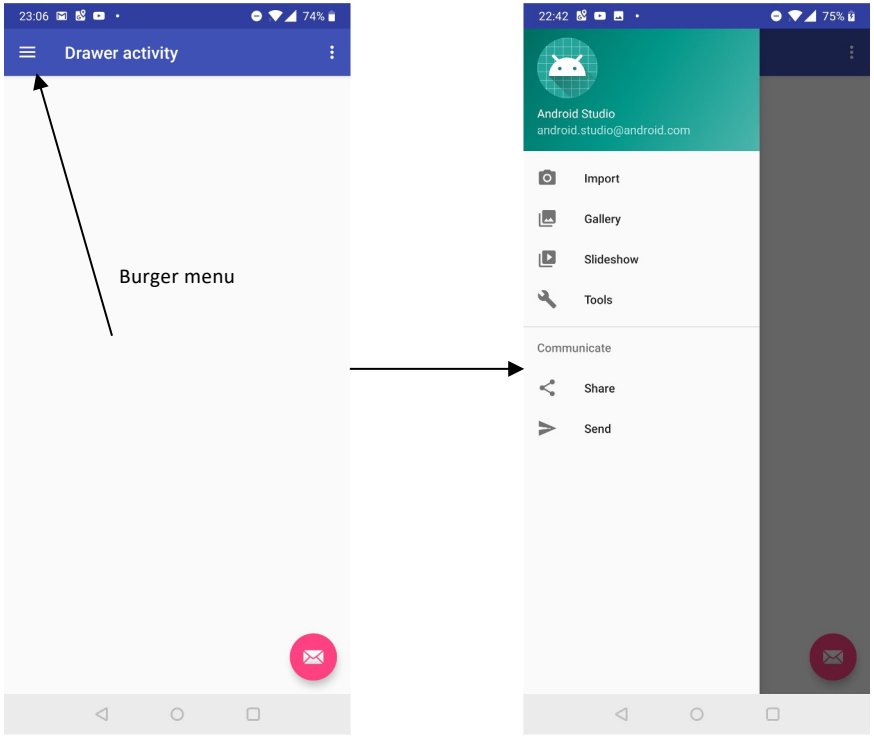
Navigation in app



Navigation in app - bottom bar



Navigation in app - drawer



Material design

- Design language from Google
- Material is the metaphor
- Inspired by physical world (reflecting light, cast shadow)
- Cross-platform (Android, iOS, Flutter, web)
- Material components available as library
- https://www.material.io/

Components

AndroidManifest.xml

- **Essential information about app for OS**
- **Package name (application unique id)**
- **Describe components**
- **Permissions**
- **Min required API level**
- **Supported/required screens, features**
- **Target SDK**
  - Compatibility modes
- **Used for filtering in app store**

Activity

- Screen with UI
- Single activity application is recommended by Google
- Activity stack
- Lifecycle
- Contains fragments and/or views

Service

- **No UI**
- **Optional notification (mandatory for foreground services)**
- **Operations that are not tight to activity lifecycle**
- **Long running tasks**
  - Download service
  - Music playback

Content provider

- **Manage and share application data**
- **Doesn’t specify storage implementation (db, file, web)**
- **Query or modify data**
- **Optional permissions**
  - Custom permissions who can access data
- **Used by system for**
  - SMS
  - Contacts
  - Call log
- **Initialized before Application**

Broadcast receiver

- **Listens for actions invoked by system or other application**
- **Static or dynamic registration**
- **System-wide**
- **Limited since API-26**
  - Implicit broadcasts
- **Examples**
  - Incoming SMS
  - Low battery, battery percentage changed
  - Connectivity change
  - Headphones connected/disconnected

Intent

- Asynchronous message between component
- Start activities
- Start or bind services
- Send broadcast

Project setup

Consider

- Target audience
- Compatibility

MinSdk

- **Lowest supported SDK**
  - Installation on older devices is not possible
- **New features are not available on older APIs**
- **Supporting old SDK can take a lot of resources to maintain compatibility**
  - API levels checks
  - Testing
  - Value of these users

Compile Sdk

Always compile with the latest SDK

- **Select newest available API at compile time**
- **Deprecations**
- **Lint checks**

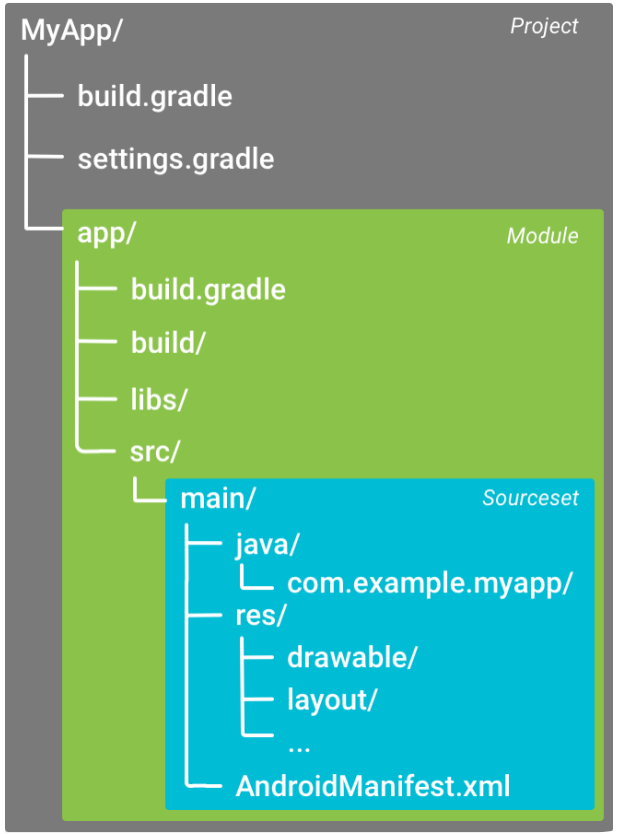
Target sdk

- **Way how system provide forward compatibility**
- **Change behavior of the app**
  - Runtime permissions handling
  - Menu button deprecation handling

Project structure

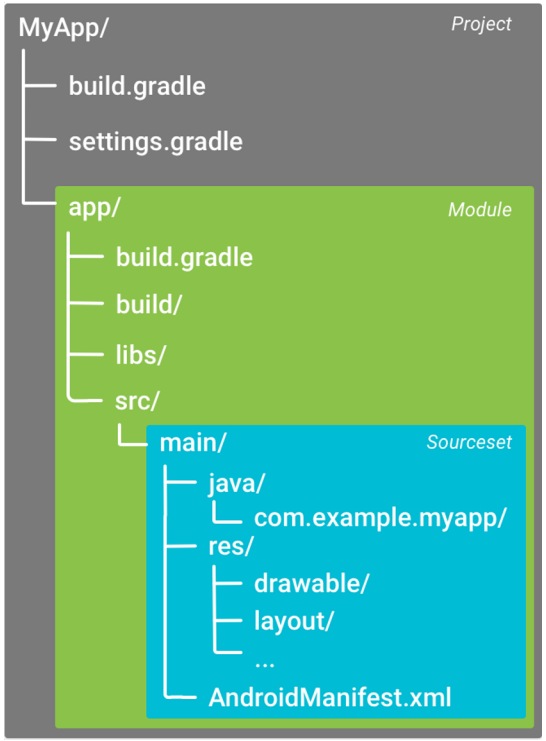
Project structure - project

- **Common configuration for modules**
  - Common dependencies versions
  - 3rd party plugins configuration



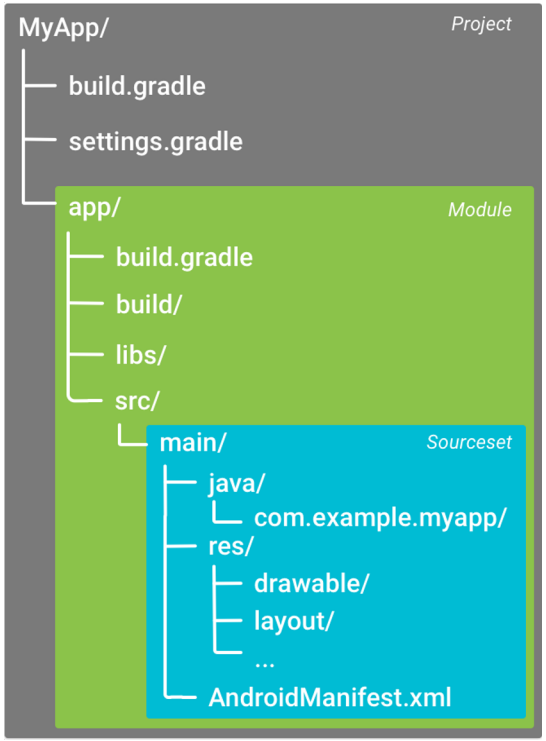
Project structure - module

- **Application or library**
- **Different module for phone/watch/tv app**
- **Multiple source sets (optional)**
  - Different version of same app (paid vs. free)



Project structure - sourceset

- Source code and resources
- Source code from main source set available everywhere
- Resources can be overridden in different source set



Project files

build.gradle(.kts)

- Configuration that applies to all modules
- Defines android build plugin version
- List of repositories where to download dependencies and gradle build plugin

settings.gradle(.kts)

- List of modules to build

Project files - gradle.properties

- **Project wide gradle fields**
- **Customization of how it will run**
  - Heap size
  - Daemon or not
  - Java_home and java arguments
  - Parallel run
  - Proxy
  - And much more

Project files - local.properties

- Contains paths to sdk and ndk
- Can’t be shared between developers
- Generated during build, do not modify it manually

- Do not include this file in submitted projects
- .gitignore

Module files - build.gradle

- Configure build setting for specific module
- Defines build variants and their source sets
- applicationId
- Min and target SDK version
- compileSdkVersion and buildToolsVersion
- Dependencies
- https://developer.android.com/reference/tools/gradle-api

Module files - libs/

- *.jar libraries
- If it is possible use library as gradle dependency

Module files - src/

- Source code
- Resources
- Assets
- Main - default sourceset for all build variants
- Recommended to split code into packages

Source set

- **java/**
  - Source codes
- **res/**
  - Resources
  - Drawables
  - Layouts
  - Values
  - …
- **assets/**
- **AndroidManifest.xml**

Resources

- Layout
- Strings
- Menu
- Animations
- Icons
- Dimensions
- Drawables
- Mipmap

Resource qualifiers

- Resources in different variants
- Drawable, drawable-mdpi…
- Values, values-cs, values-de
- Layout, layout-sw600dp

Resources - drawables

- **Bitmaps**
- **9-patch png**
- **State lists**
- **Vector drawables**
  - Since API-21
  - Backward compatibility with support library

Always prefer using vector drawables

Resources - layout

- Definition of UI
- Used for Activity or Fragment
- XML

Resources units

- **Dp - density independent pixel**
  - On 160dpi screen 1dp = 1px
- **Sp - scale independent pixel (fonts)**
  - Similar to dp, but scaled by the user’s font size preference
- **Never use px**

Binding between resources and java

- **XML elements has id generated in R.java**
- **R.id.txt_headline**
- **R.layout.activity_main**

- **Binding**
  - Manual
  - View binding – preferred way
    - https://developer.android.com/topic/libraries/view-binding
  - Data binding
  - Kotlin synthetics (deprecated)

Layouts

- **Extends ViewGroup**
- **Defined in XML or programmatically**
- **Folder res/layout**
- **Options**
  - FrameLayout
  - LinearLayout
  - RelativeLayout
  - TableLayout
  - GridLayout
  - ConstraintLayout
    - Google IO 2016
    - Available as support library

Layout - FrameLayout

- Places all items in top left corner
- Usage as placeholder for other view/fragment
- Fast

Layout - LinearLayout

- Places childs vertically or horizontally
- Possible to use weight to size item in some ratio
- Usually leads to layout nesting

Layout - Constraint layout

- **Available as dependency**

```
implementation "androidx.constraintlayout:constraintlayout:2.1.4"
```

- **“Extended relative layout”**
- **Constraint is connection or alignment to another view/parent/guideline**
- **Recommended today**
- **https://developer.android.com/develop/ui/views/layout/constraint-layout**

Jetpack Compose

- Modern toolkit for building native UI
- Intuitive Kotlin API
- UI in Kotlin instead of XML
- Reusability of the UI components

https://developer.android.com/jetpack/compose

Widgets – UI elements

- **Extends View**
- **width and height needs to be set**
  - Can be replaced by weight
  - match_parent
    Fills the whole width/height of parent
  - wrap_content
    Wraps around the content
  - dimension

Widgets

- **Button**
- **TextView**
- **EditText**
- **ImageView**
- **CheckBox**
- **RadioButton**
- **WebView**
- **AdapterView**
  - ListView
  - Spinner
- **RecyclerView**

Hello World

Gen™

Activity & Back stack

norton Avast LifeLock by norton Avira AVG REPUTATION DEFENDER by norton CCleaner

Activity

- Presentation layer of application
- Only UI component
- Contains Views or Fragment
- Every activity defined in manifest
- Runs on UI thread
- All components run in one process by default
- Lifecycle
- Activity back stack

Starting activity

- Intent describes which activity to start
- Can contain data for new activity
- Flags - manipulation with activity stack

```
val intent = Intent(activity, SecondActivity::class.java)
intent.putExtra("key", "value")
intent.putExtra("keyInt", 5)
startActivity(intent)
```

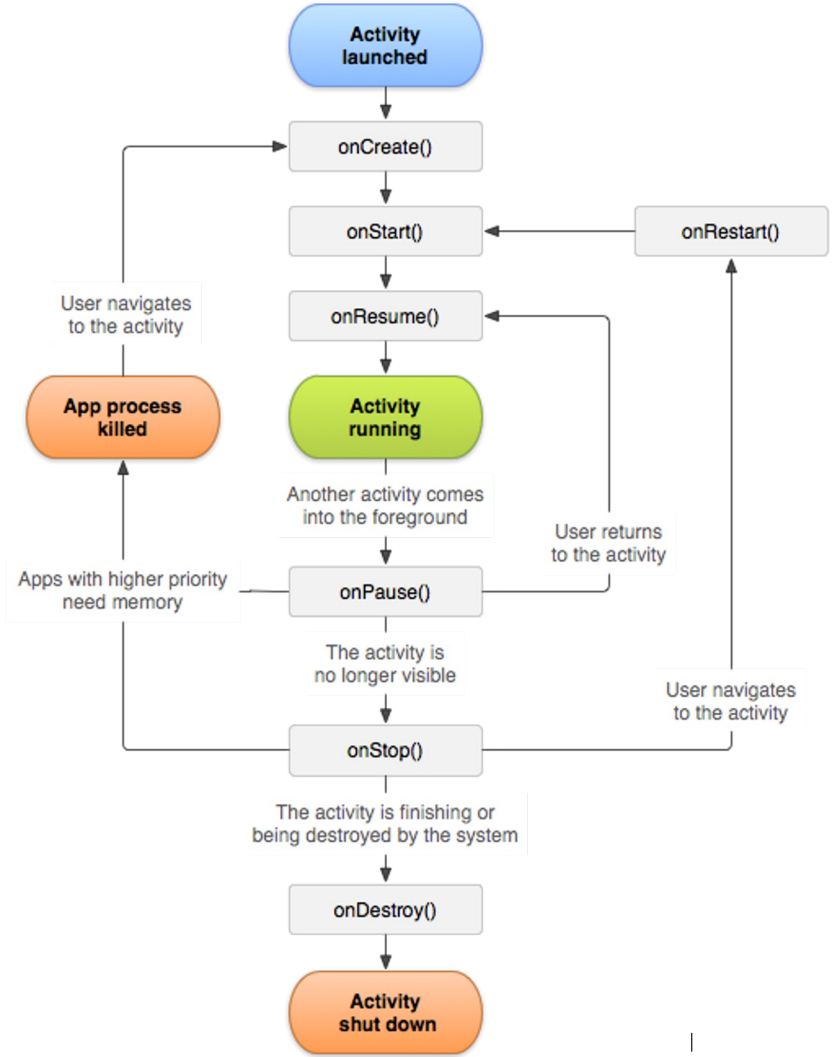
Explicit vs. implicit intent

- **Explicit intent**
  - Specify component by fully qualified class name
  - Typically component in our application
- **Implicit intent**
  - Just declare general action to perform
  - Enables multiple apps to handle that action
  - Examples
    - Send email - ACTION_SEND
    - Open browser - ACTION_VIEW
  - If multiple apps are capable to handle intent, system shows picker
  - Intent filters defined in manifest

Activity - states

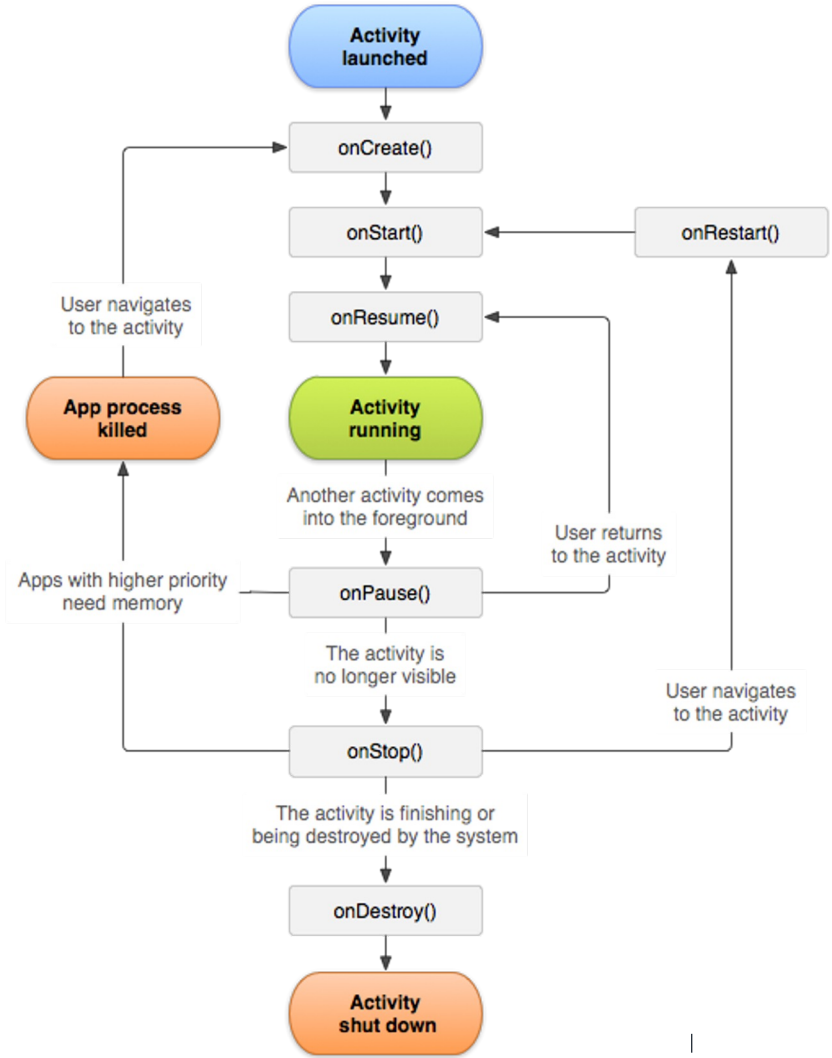
- **Activity changes state based on the user or OS actions:**
  - User navigates to activity
  - User switches to different app and returns
  - User presses back button
  - Screen is automatically locked
  - Phone starts ringing
  - ...
- **Lifecycle callbacks:**
  - Methods called by OS when state of activity changes
  - Allows programmer to react to these changes

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



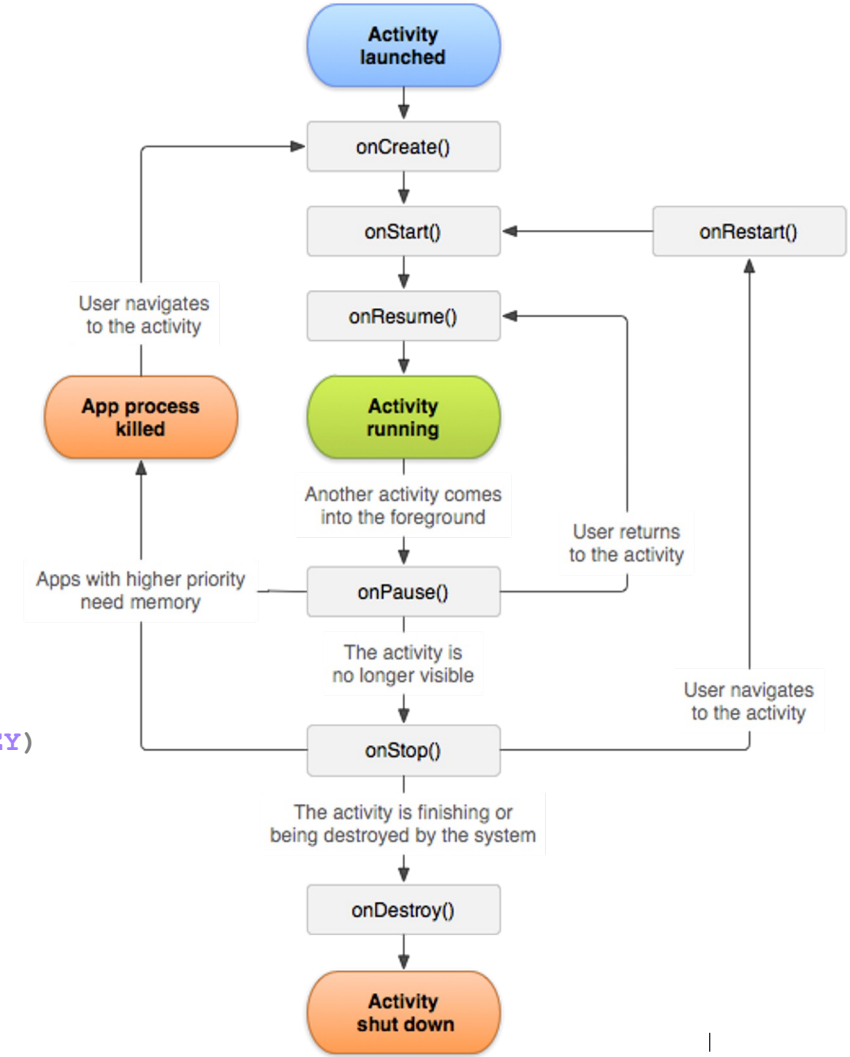
Activity - states

- **Created**
  - Activity is being created
- **Started**
  - Activity is about to be visible
- **Resumed**
  - Running, is visible, user can interact
- **Paused**
  - Partially visible, remains in memory
- **Stopped**
  - Different activity is on top
  - Moved to background
  - Still alive, remains in memory
  - Hosting process can be killed
- **Destroyed**



Activity#onCreate(Bundle)

- **Activity is being created**
- **One-time event, called only once per instance**
- Create views
- Passed Bundle object contains activity previous state
- Read data from starting intent
- Always followed by #onStart()

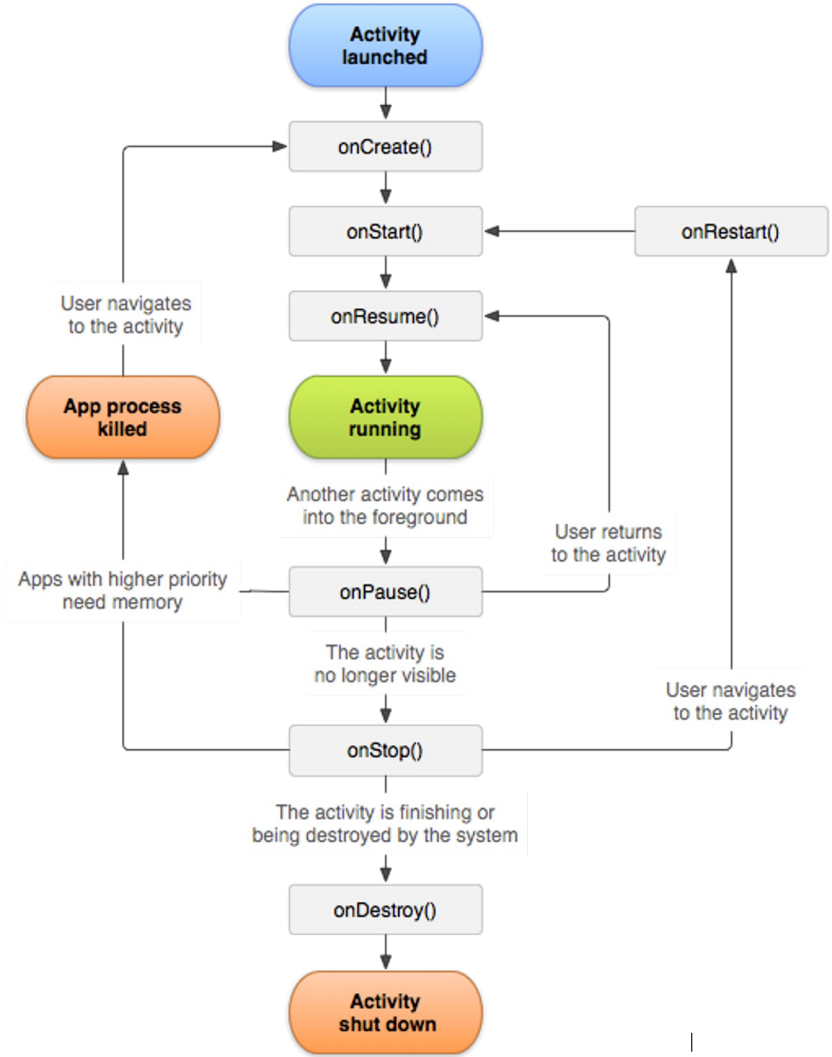



```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    previousState = savedInstanceState?.getString(STATE_KEY)
    setContentView(R.layout.main_activity)

    // TODO: initialize variables, bind data to list, …
}
```

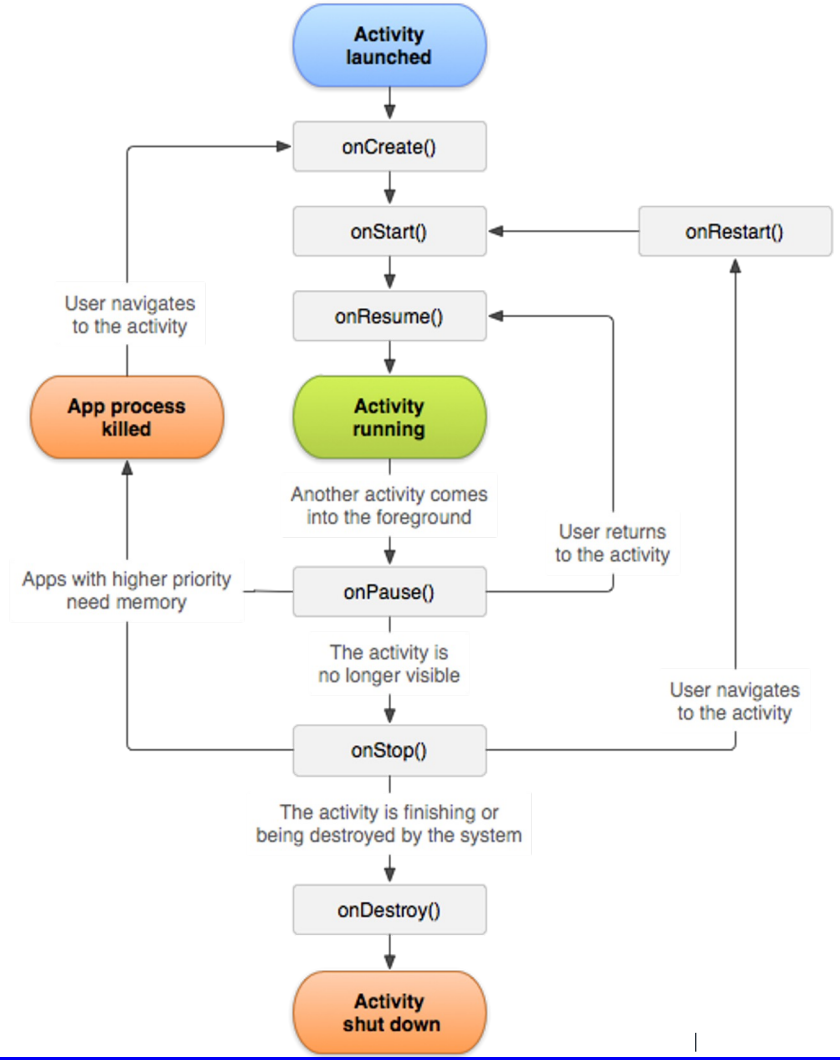
Activity#onStart()

- **Called before the activity become visible to the user**
- **Can be called multiple times**
- **Followed by**
  - onResume() if come to the foreground
  - onStop() if becomes hidden
- **Activity is partially visible, register listeners for changing UI**
- **Register broadcast receivers**



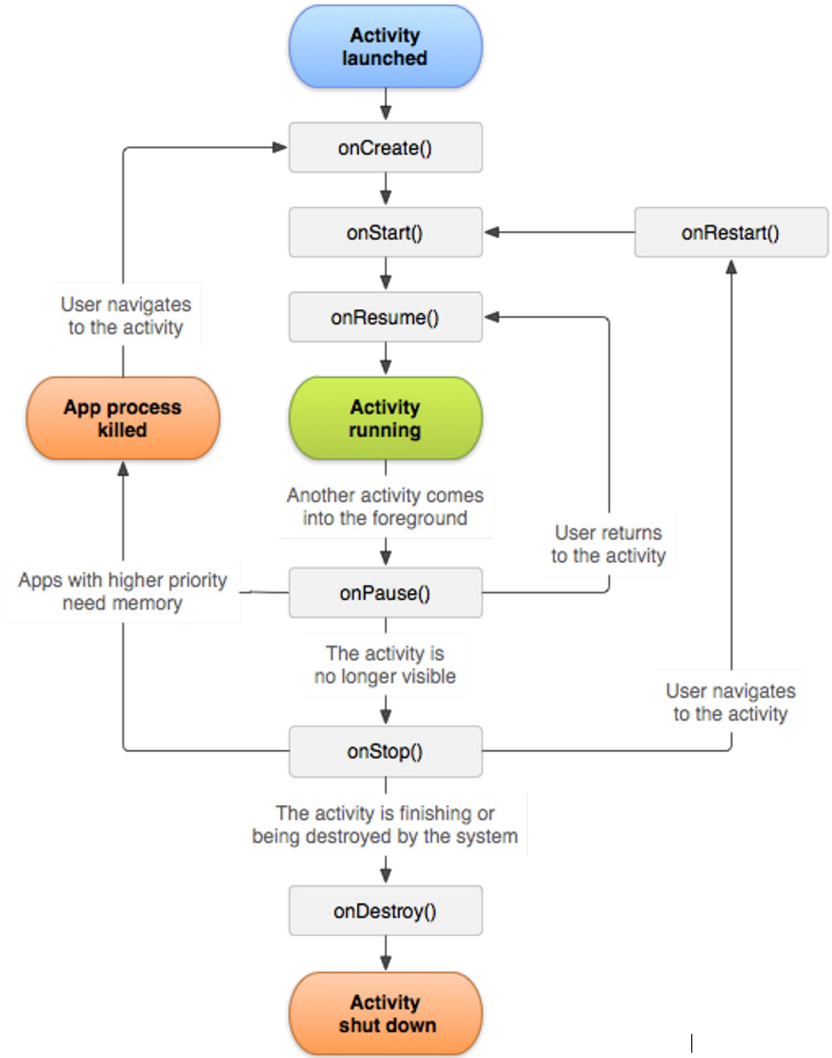
Activity#onResume()

- Called just before activity start interacts with user
- Activity is on top of activity stack
- Run stuff for user
- Always followed by onPause()



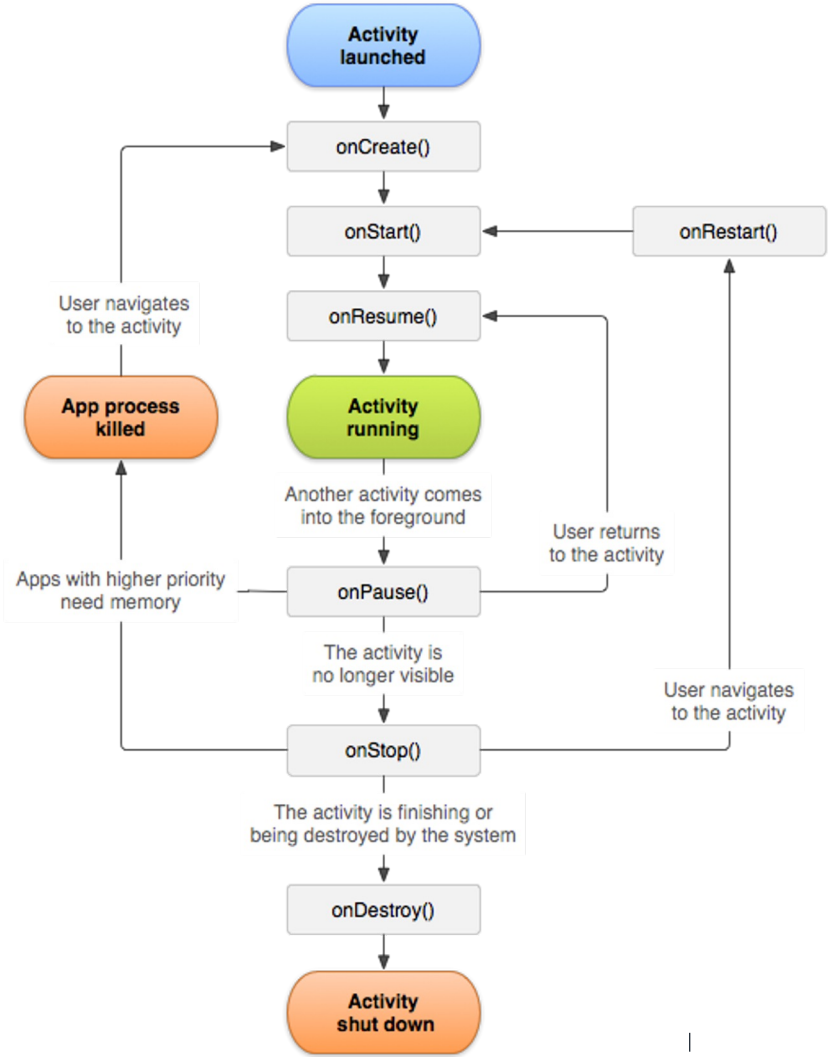
Activity#onPause()

- **System is about to resume another activity**
- **Stop animations and CPU intensive stuff**
- **Should be very fast, because another activity onResume() waits until this finishes**
- **Followed by**
  - onResume() if the activity returns back to the front
  - onStop() if became invisible to the user
- **Activity can be killed by system**
- **Counterpart to onResume()**



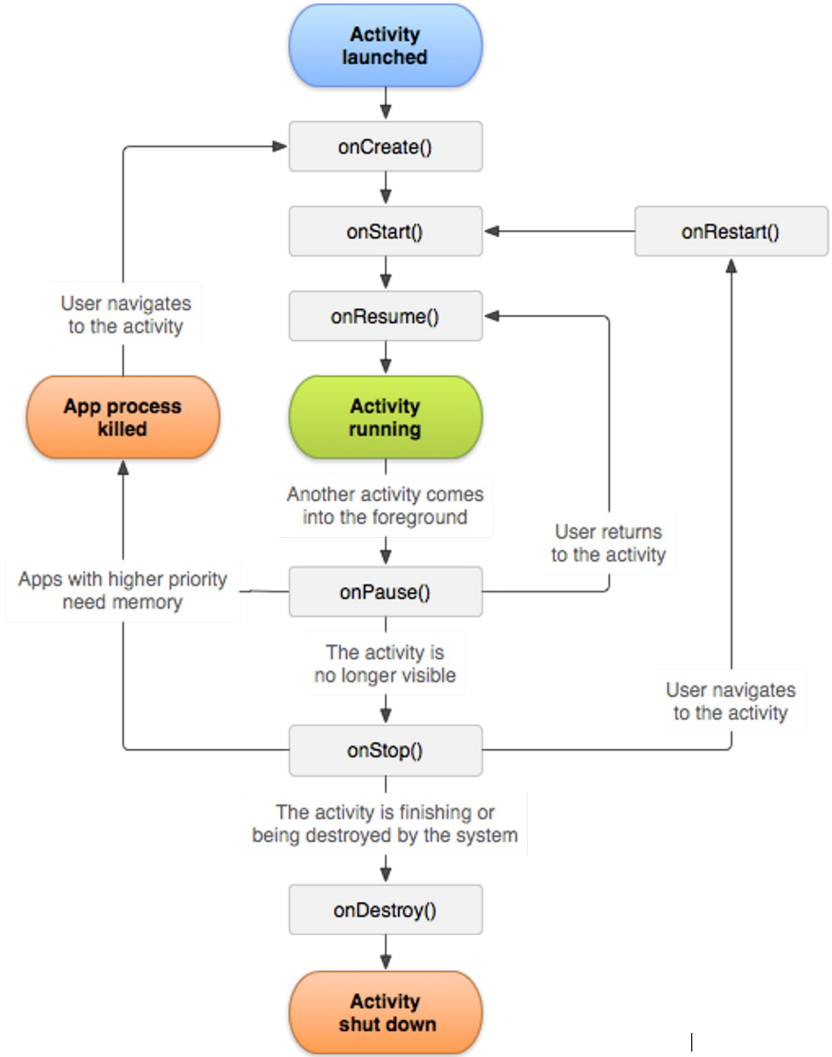
Activity#onStop()

- **Called when it is no longer visible to the user**
- **It is being destroyed or another activity has been resumed and covering it.**
- **Finish stuff started in #onStart()**
- **Followed by**
  - onRestart() - coming back to interact with user
  - onDestroy() - activity is going away
- **Called when being minimized, navigate to another screen**



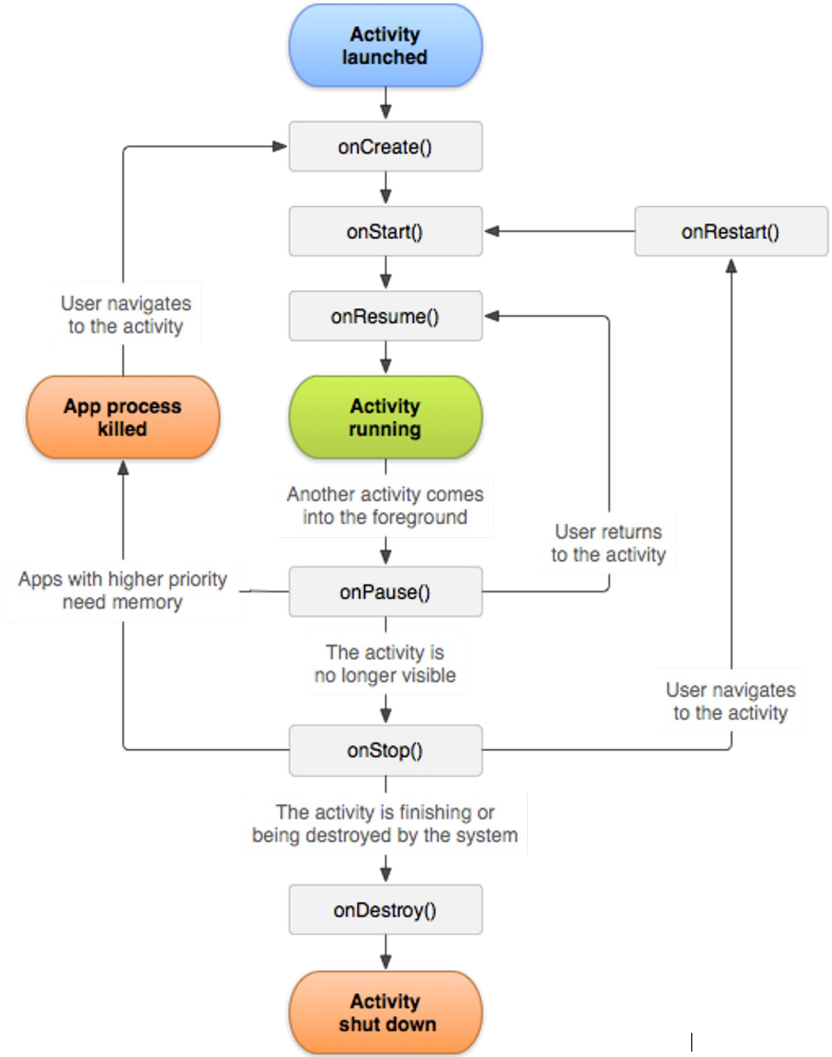
Activity#onDestroy()

- Called before activity is destroyed
- Activity is finished by #finish() method
- System needs more resources (RAM)



Activity#onRestart()

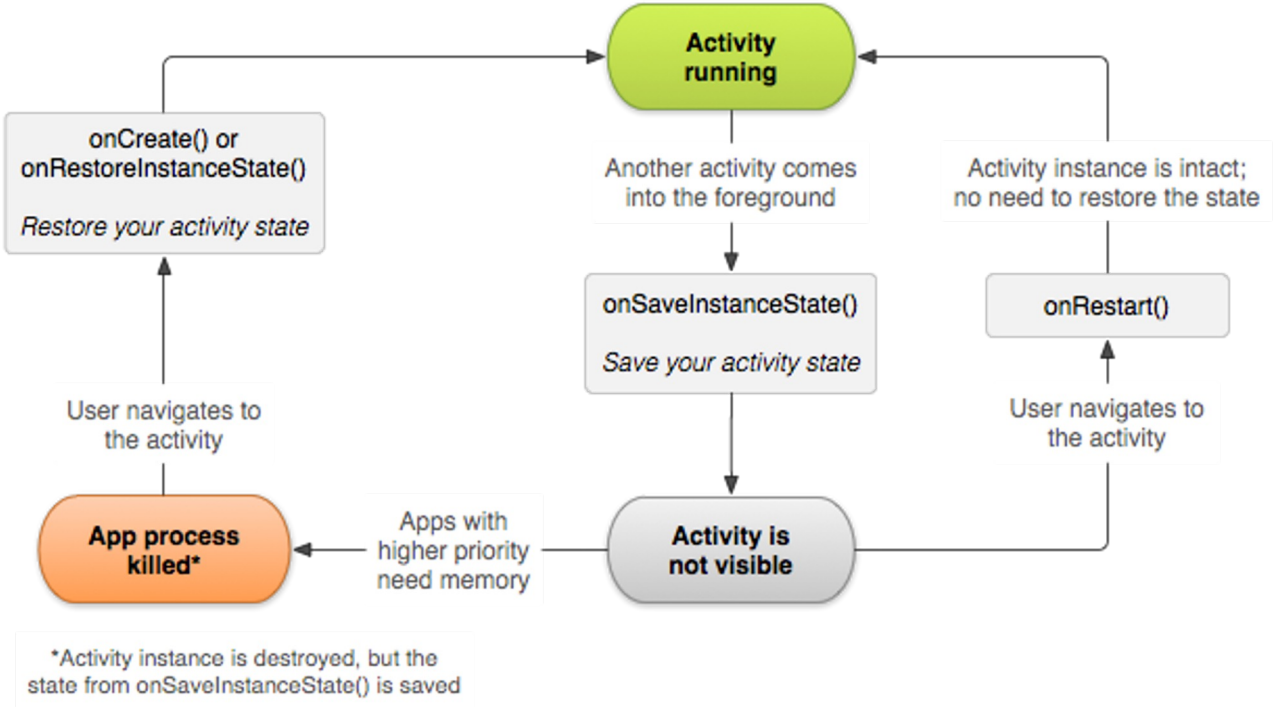
- Called after activity has been stopped, and before is started again



Configuration changes

- **Activity is destroyed and recreated**
  - Screen rotation
  - Language change
  - HW keyboard opens
  - Projector is connected
- **Needs to be handled properly**
  - Activity#onSaveInstanceState
  - Activity#onCreate(savedInstanceState: Bundle?)
  - Activity#onRestoreInstanceState(savedInstanceState: Bundle)

Save activity state



Saving activity state

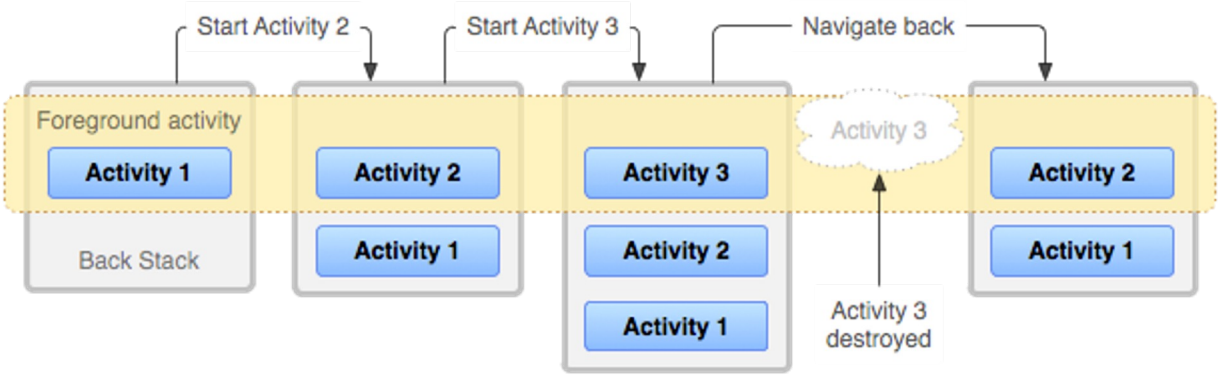
- **System can kill background activity to free up resources => state of the activity is lost**
- **Implement #onSaveInstanceState**
  - Called before activity is vulnerable to destruction
  - Passed Bundle is for remembering its state
  - Bundle with the stored state is passed into #onCreate and #onRestoreInstanceState (called before #onStart())
  - Default implementation takes care of widget with unique id (user input), but doesn't store state (enabled/disabled)

Bundle

- Mapping parcelable and serializable objects
- String keys
- #putString, #putInt
- #getString, #getInt
- Other java primitives

Tasks and back stack

- **Task is collection of activities, to perform certain job**
  - Activity in task can be from different application (send email)
- **Activities arranged in a stack, in order in which there were opened**
- **Task has its own back stack**



Tasks and back stack

- **Sometimes is necessary to change behavior of back stack**
- **Manifest attributes**
  - `taskAffinity`
  - `launchMode`
  - `allowTaskReparenting`
  - `clearTaskOnLaunch`
  - `alwaysRetainTaskState`
  - `finishOnTaskLaunch`
- **Intent flags**
  - `FLAG_ACTIVITY_NEW_TASK`
    - Start activity in new task, or bring task with that activity
  - `FLAG_ACTIVITY_CLEAR_TOP`
    - If the activity is in stack, pick them and destroy all other activities on top
  - `FLAG_ACTIVITY_SINGLE_TOP`
    - Do not start new instance of activity, if is already on top of stack

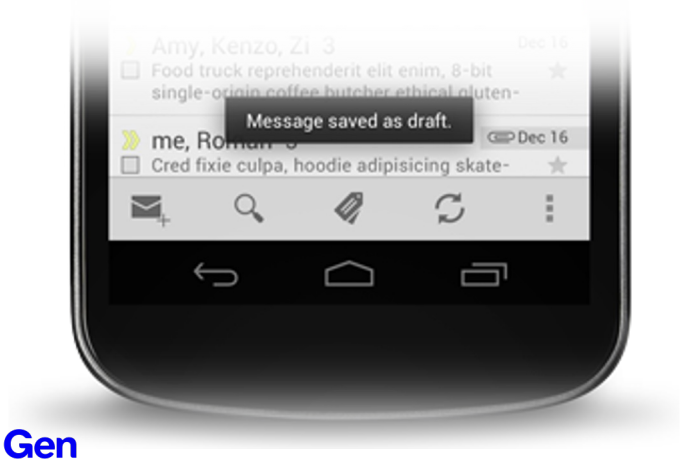
Task affinity

- If you need that flag `FLAG_ACTIVITY_NEW_TASK` open activity in new task you need to set different affinity for that activity
- It needs to be set for independent apps in one APK, we use it for debug tools (separate app which allows us to (re)set some values in main app)

Toast

- Simple non modal information
- Displayed for short period of time
- Doesn’t have user focus
- `android.widget.Toast`

```
Toast.makeText(context, "Toast example", Toast.LENGTH_LONG).show()
```



Log messages

- Static method in Log class
- `android.util.Log`
- `Log.{v,d,i,w,e,wtf}(tag: String, message: String, e: Throwable)`

- `Verbose`
- `Debug`
- `Info`
- `Warning`
- `Error`
- `What a terrible failure`

Context

Context

- Abstract class implemented by components
- `android.content.Context`

- Resources access
- Register/unregister BroadcastReceivers
- Run Activity, Services
- Binds Services

Context

- **Application**
  - Single instance
  - Extends Context
- **Activity/Service**
  - Multiple instances
  - Extends Context
  - Can be easily leaked
- **BroadcastReceiver**
  - Receive instance of Context in `BroadcastReceiver#onReceive()`
  - `registerReceiver()` and `bindService()` doesn't work
- **ContentProvider**
  - Not instance of Context
  - `getContext()` returns Context of application which called the receiver

Thank you

Lukáš Prokop
Simona Kurňavová

Lukas.Prokop@gendigital.com
Simona.Kurnavova@gendigital.com

Gen

