Android Basics: Components, project setup

Android Lecture 2

2024
Lukáš Prokop
Simona Kurňavová

About this course

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

Garant: Jan Kofroň

Lecturers: Lukáš Prokop, Simona Kurňavová

Schedule for semestral projects:

- *October 1 – December 1*: Forming project groups (1-3 students) and creating project specifications
- *December 1*: The project specification has to be accepted by a lecturer
- *February 28:* Final version of the project
- *April 15*: Issues identified by lecturers fixed

How to submit project specification:

Via email to Jan Kofroň (jan.kofron@d3s.mff.cuni.cz) with Lukáš Prokop (Lukas.Prokop@gendigital.com) and Simona Kurňavová (Simona.Kurnavova@gendigital.com) as cc. *Email should contain:*

- What is the purpose of the application
- Description of features and functionalities of the application
- Optionally: technical stack and other clarifying information.

How to hand over project:

Ideally using Github/Gitlab/Bitbucket repository link (please make sure it would be accessible to us).

Agenda

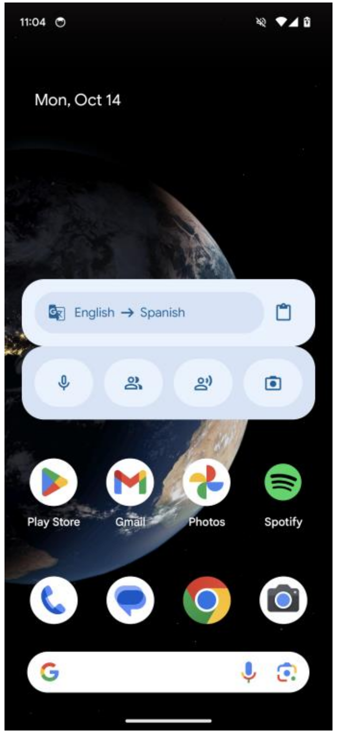
- Android UI Overview
- Project structure
- Android components
- Activity and back stack
- User Interface

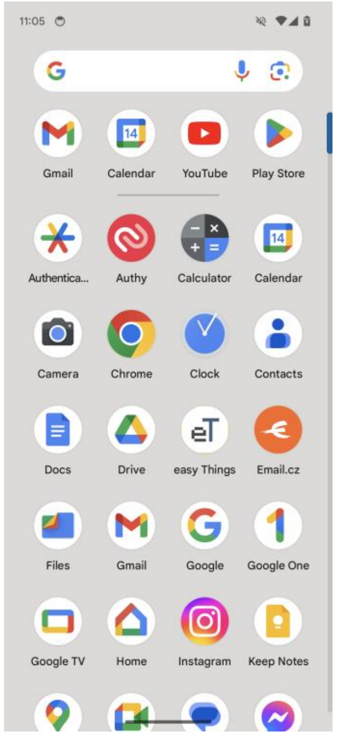
Gen

Android UI Overview

Launcher

- **Acts as home screen and app drawer**
- **Contains apps and widgets**
- **App widget vs. Widget:**
  - Widget is UI item (button, checkbox, etc).
  - App widget is interactive component in launcher/home screen.
- **Stock launcher vs. Third party**

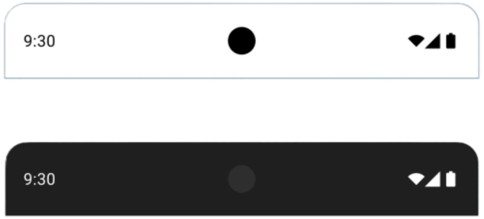


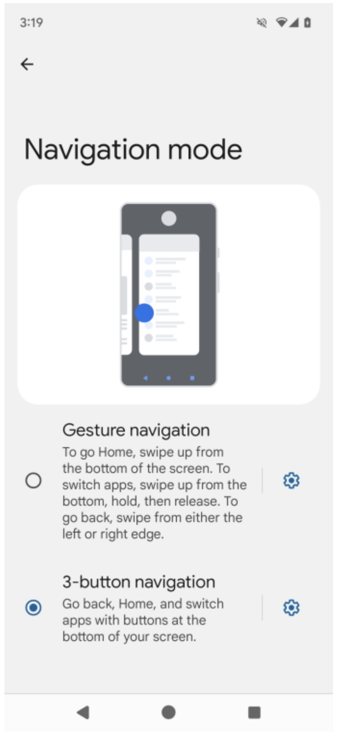


App widgets:

Status bar and navigation

- **Purpose of status bar**: Displays essential information about the device's current state and notifications.
- **Purpose of navigation bar**: Provides system-wide controls to navigate through apps and the Android system.

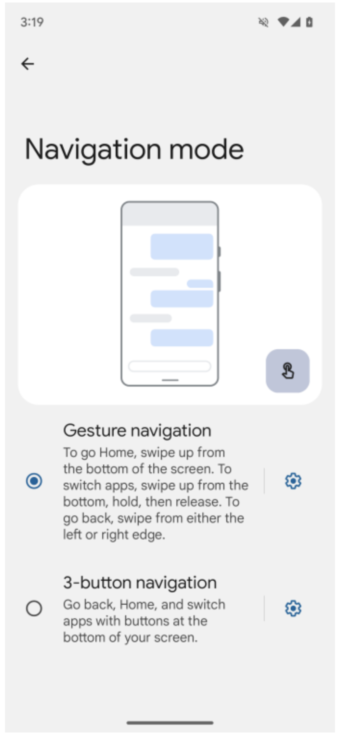



3:19

Navigation mode

Gesture navigation
To go Home, swipe up from the bottom of the screen. To switch apps, swipe up from the bottom, hold, then release. To go back, swipe from either the left or right edge.

[✓ 3-button navigation]
Go back, Home, and switch apps with buttons at the bottom of your screen.


3:19

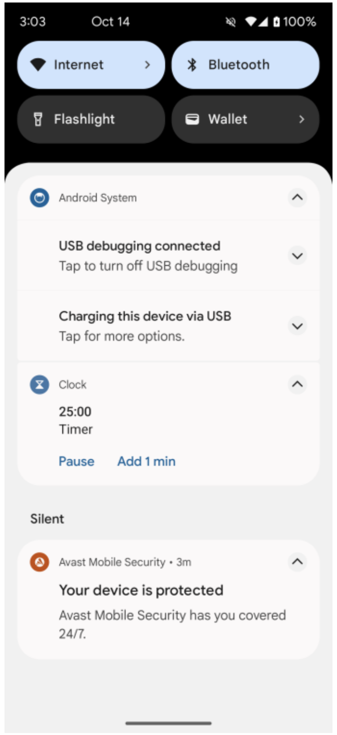
Navigation mode

[✓ Gesture navigation]
To go Home, swipe up from the bottom of the screen. To switch apps, swipe up from the bottom, hold, then release. To go back, swipe from either the left or right edge.

3-button navigation
Go back, Home, and switch apps with buttons at the bottom of your screen.

Notifications

- Alerts that inform users about important events, updates, or actions from apps
- Notification actions (since Android 4.1)
- Can be visible in lock screen (since Android 5.0)
- Notification groups (since Android 7.0)
- Notification channels (since Android 8.0)
- Notification badges (dots) (since Android 8.0)
- *Importance*:
  - Urgent: makes a sound and appears as a heads-up notification.
  - High: makes a sound.
  - Medium: makes no sound.
  - Low: makes no sound and doesn't appear in the status bar.



Android System

USB debugging connected
Tap to turn off USB debugging

Charging this device via USB
Tap for more options.

Clock

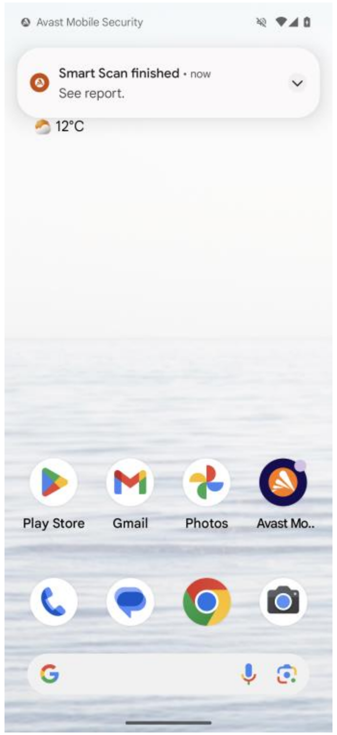
25:00
Timer

Pause Add 1 min

Silent

Avast Mobile Security • 3m

Your device is protected
Avast Mobile Security has you covered 24/7.

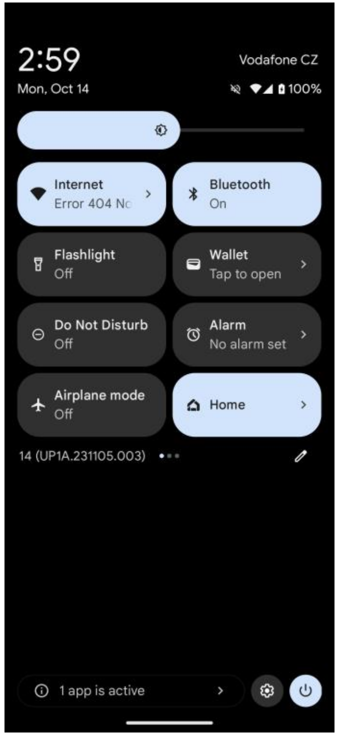


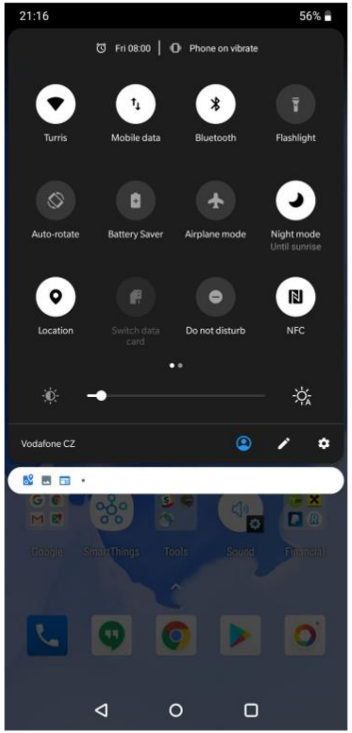
Smart Scan finished • now
See report.

Gen

Quick settings

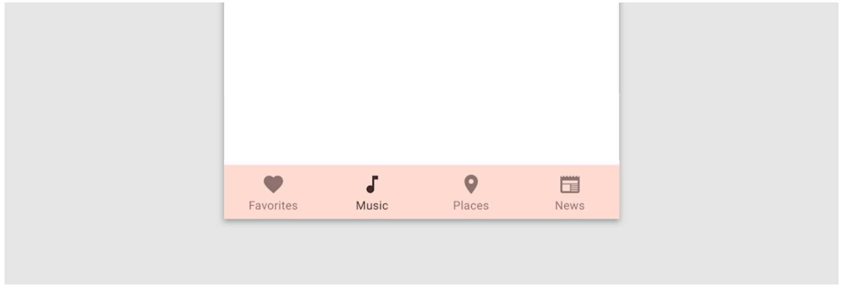
- Since API 16: part of the AOSP
- Since API 24: custom tiles
- Purpose of tile action:
  - Used often
  - Fast access needed
  - Ideally both

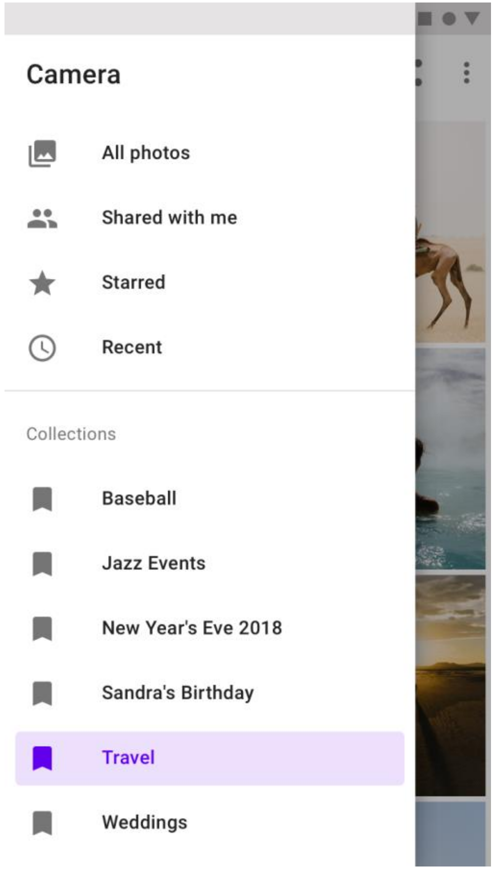




Navigation inside of the app

- Bottom navigation
- Drawer



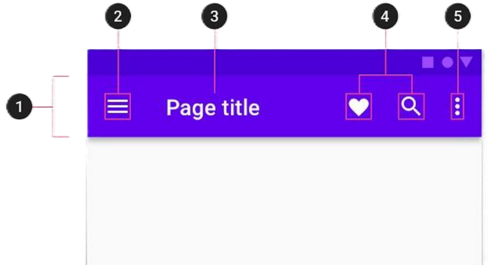


Source1, Source2

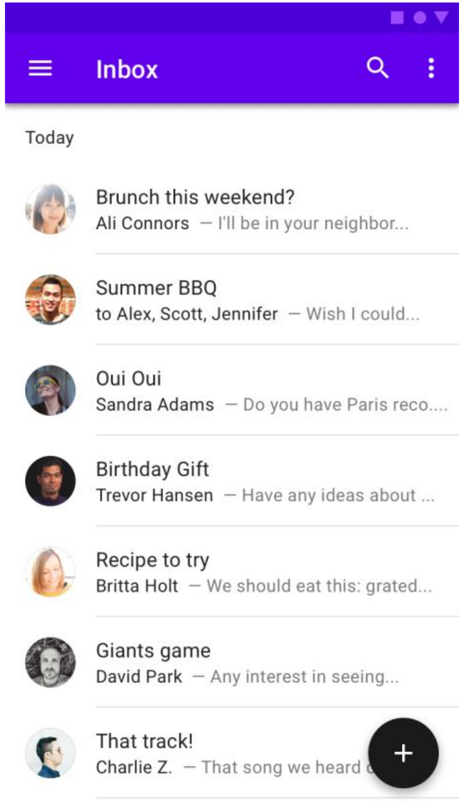
Gen

Navigation inside of the app

- Toolbar (or app bar)
- Floating action button



1. Container
2. Navigation icon
3. Title
4. Action items
5. Overflow menu



Today

Brunch this weekend?
Ali Connors — I'll be in your neighbor...

Summer BBQ
to Alex, Scott, Jennifer — Wish I could...

Oui Oui
Sandra Adams — Do you have Paris reco....

Birthday Gift
Trevor Hansen — Have any ideas about ...

Recipe to try
Britta Holt — We should eat this: grated...

Giants game
David Park — Any interest in seeing...

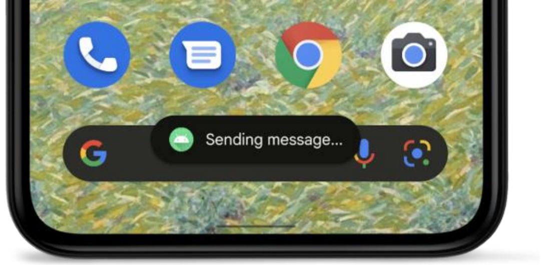
That track!
Charlie Z. — That song we heard

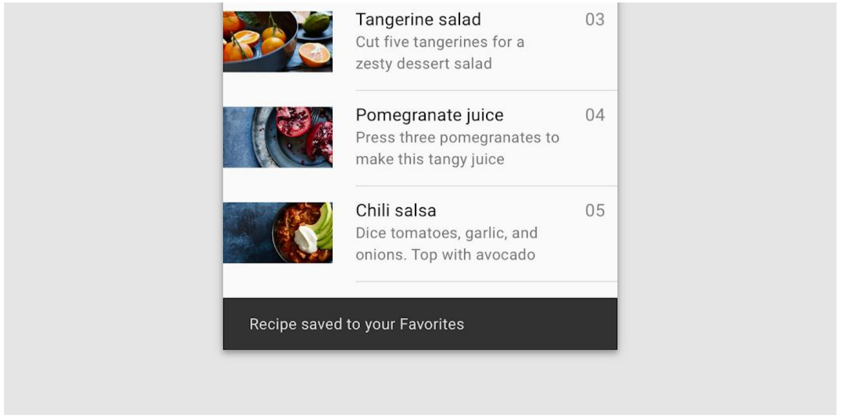
Source1, Source2

Gen

In-app messaging

- Dialogs
- Toasts
- Snackbars

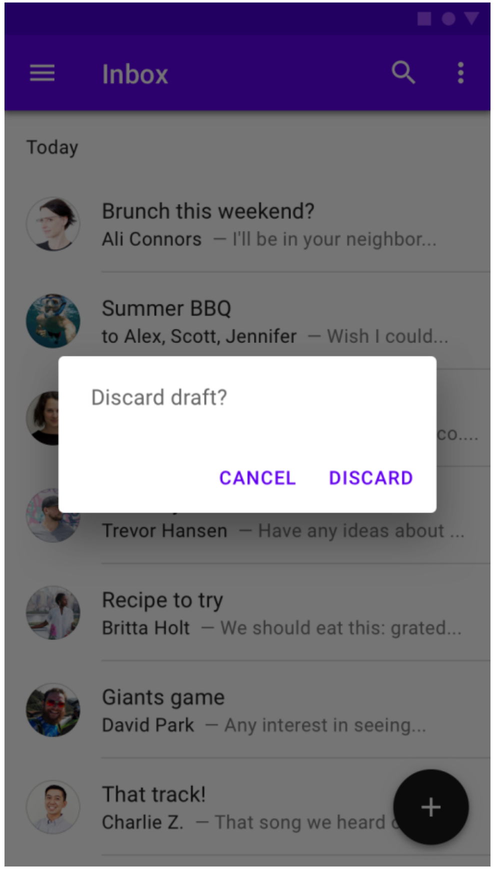



Tangerine salad 03
Cut five tangerines for a zesty dessert salad

Pomegranate juice 04
Press three pomegranates to make this tangy juice

Chili salsa 05
Dice tomatoes, garlic, and onions. Top with avocado

Recipe saved to your Favorites


Inbox

Today

Brunch this weekend?
Ali Connors — I'll be in your neighbor...

Summer BBQ
to Alex, Scott, Jennifer — Wish I could...

Discard draft?

CANCEL DISCARD

Trevor Hansen — Have any ideas about ...

Recipe to try
Britta Holt — We should eat this: grated...

Giants game
David Park — Any interest in seeing...

That track!
Charlie Z. — That song we heard

Source1, Source2

Gen

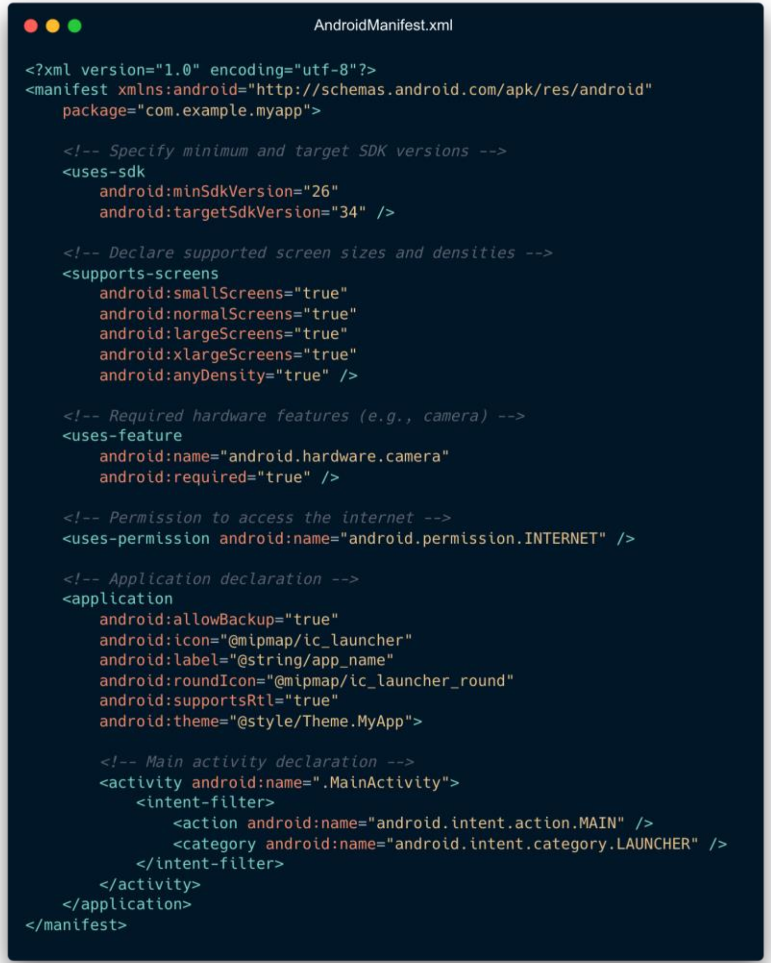
Material design

- 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/
- https://www.materialpalette.com/
- https://m3.material.io/styles

Android Components

AndroidManifest.xml

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



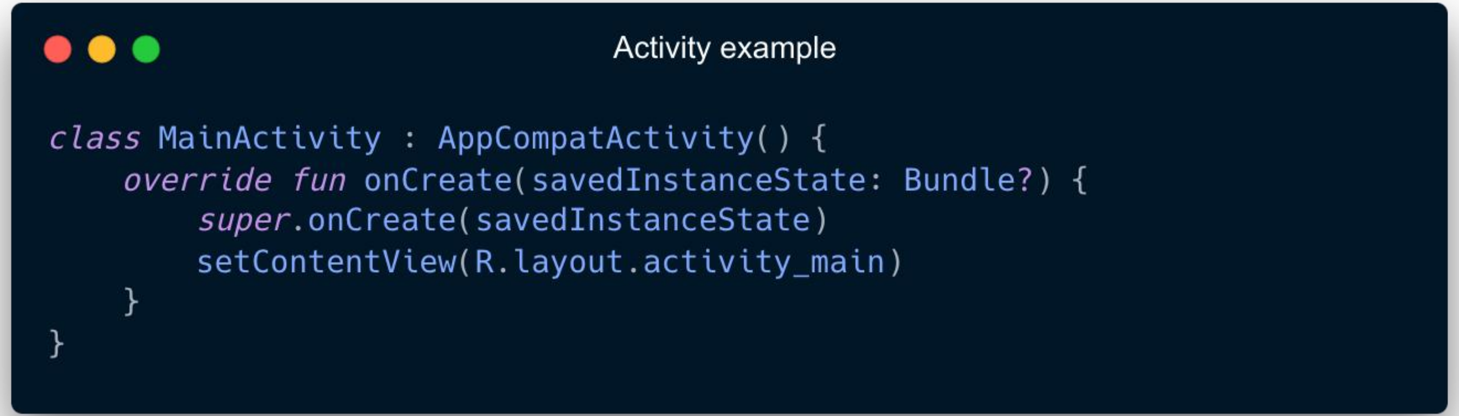
AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">

    <!-- Specify minimum and target SDK versions -->
    <uses-sdk
        android:minSdkVersion="26"
        android:targetSdkVersion="34" />

    <!-- Declare supported screen sizes and densities -->
    <supports-screens
        android:smallScreens="true"
        android:normalScreens="true"
        android:largeScreens="true"
        android:xlargeScreens="true"
        android:anyDensity="true" />

    <!-- Required hardware features (e.g., camera) -->
    <uses-feature
        android:name="android.hardware.camera"
        android:required="true" />

    <!-- Permission to access the internet -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- Application declaration -->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApp">

        <!-- Main activity declaration -->
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Gen

Activity

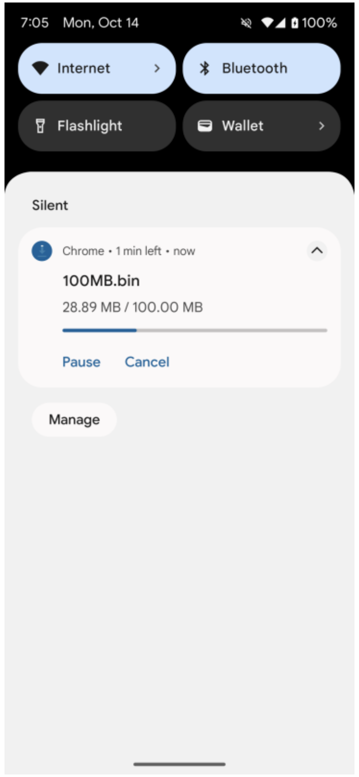
- Screen with UI
- Activity stack
- Lifecycle
- Single activity application is recommended by Google
- Can contain: Fragments, Views, Composables (when used with Jetpack Compose)


Activity example

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Service

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



Chrome • 1 min left • now

100MB.bin

28.89 MB / 100.00 MB

Gen

Content provider

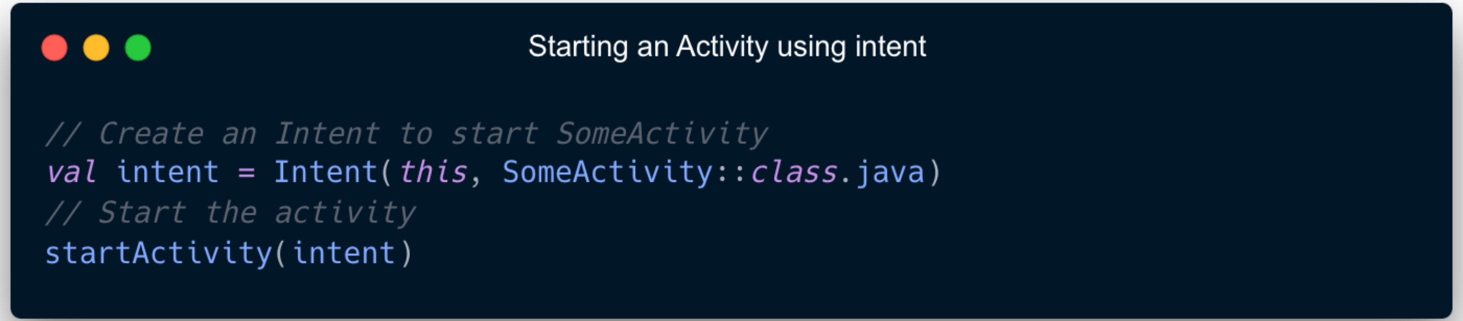
- Manage and share application data
- Doesn't specify storage implementation (db, file, web)
- Query or modify data
- Optional permissions
    - Custom permissions who can access data
- Used by system for
    - SMS
    - Contacts
    - Call log
- Initialized before Application
    - Used by AndroidX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

Broadcast receiver

- Listens for actions invoked by system or other application
- Static or dynamic registration
- System-wide
- Limited since Android 8.0
    - Implicit broadcasts
- Examples:
    - Incoming SMS
    - Low battery, battery percentage changed
    - Connectivity change
    - Headphones connected/disconnected
    - ...

Intent

- Asynchronous message between component
- Starts activities
- Starts or binds services
- Sends broadcast



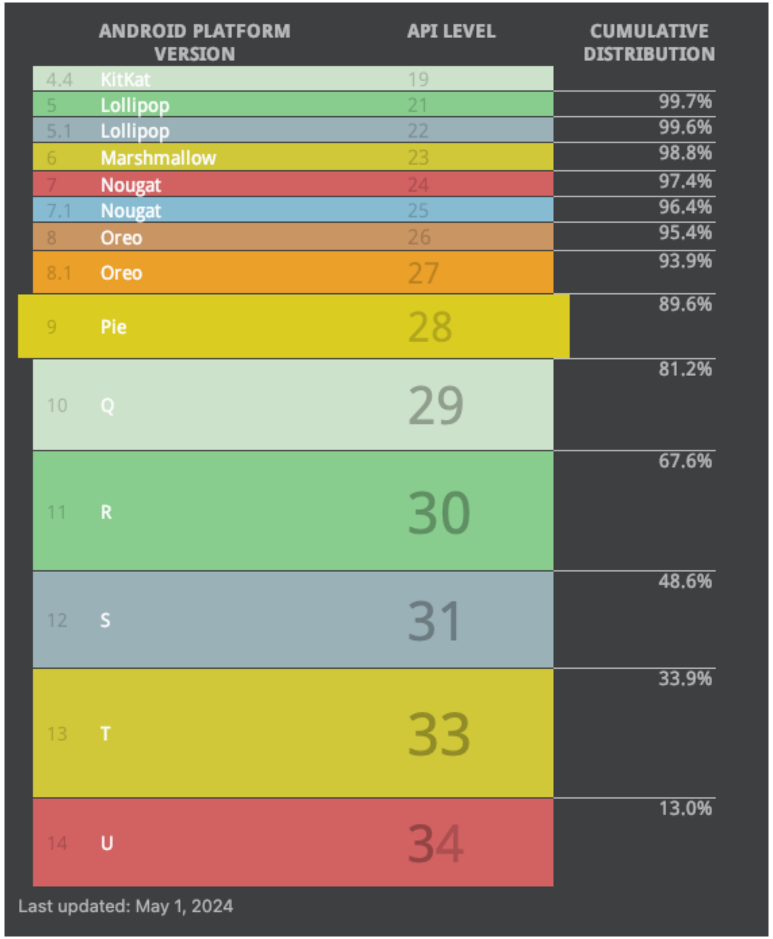
Starting an Activity using intent

```
// Create an Intent to start SomeActivity
val intent = Intent(this, SomeActivity::class.java)
// Start the activity
startActivity(intent)
```

Project setup

Consider

- Target audience
- Compatibility



ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.4	KitKat	19	
5	Lollipop	21	99.7%
5.1	Lollipop	22	99.6%
6	Marshmallow	23	98.8%
7	Nougat	24	97.4%
7.1	Nougat	25	96.4%
8	Oreo	26	95.4%
8.1	Oreo	27	93.9%
9	Pie	28	89.6%
10	Q	29	81.2%
11	R	30	67.6%
12	S	31	48.6%
13	T	33	33.9%
14	U	34	13.0%

Last updated: May 1, 2024

Source: Android Studio

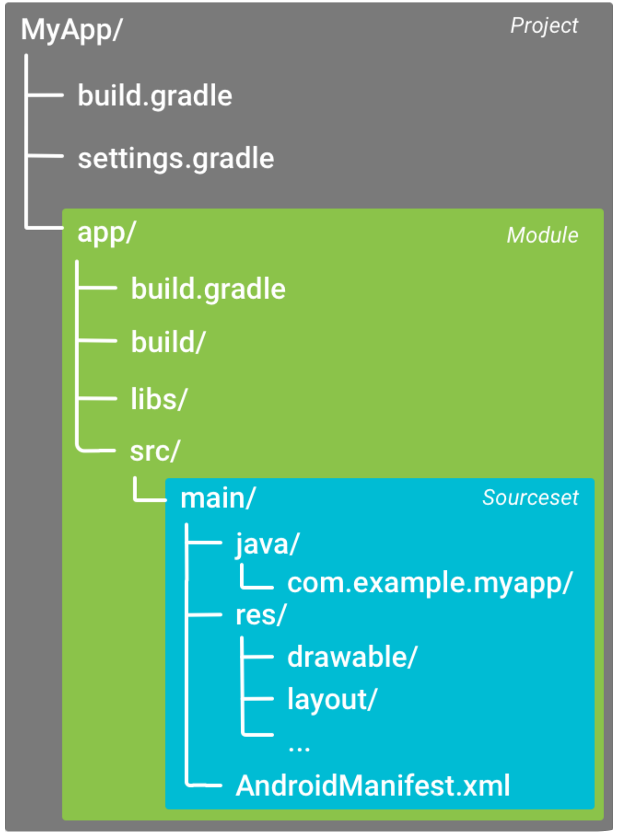
minSdk, compileSDK, targetSDK

- **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
- **compileSDK**: Always compile with the latest SDK !
  - Select newest available API at compile time
  - Deprecations
  - Lint checks
- **targetSDK**: Way how system provide forward compatibility
  - Change behavior of the app
  - Runtime permissions handling
  - Menu button deprecation handling

Project structure

Project structure: Project

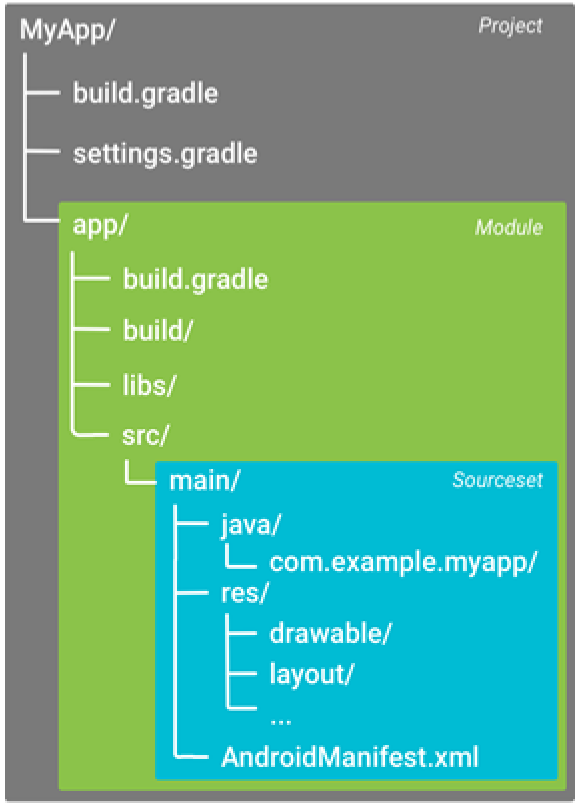
- Common configuration for modules
  - Common dependencies versions
  - 3rd party plugins configuration



```
MyApp/                                  Project
├── build.gradle
├── settings.gradle
└── app/                                Module
    ├── build.gradle
    ├── build/
    ├── libs/
    └── src/
        └── main/                       Sourceset
            ├── java/
            │   └── com.example.myapp/
            ├── res/
            │   ├── drawable/
            │   ├── layout/
            │   └── ...
            └── AndroidManifest.xml
```

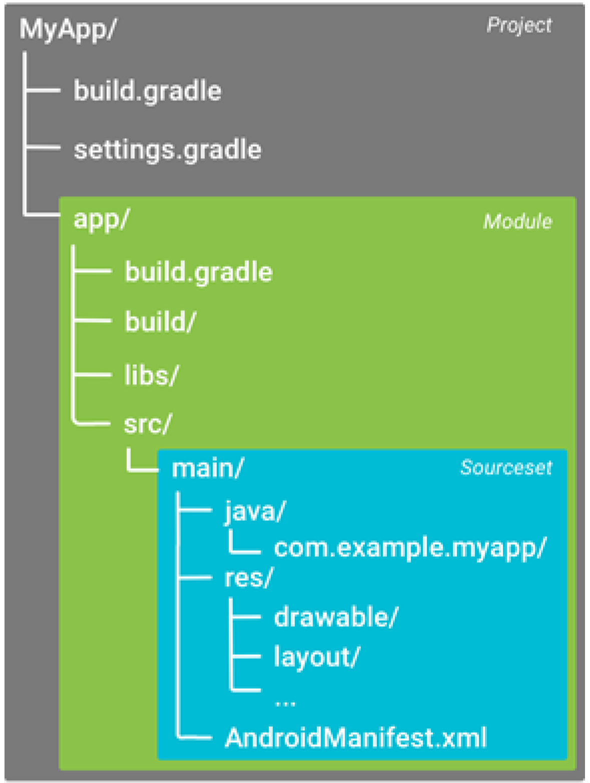
Project structure: Module

- 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

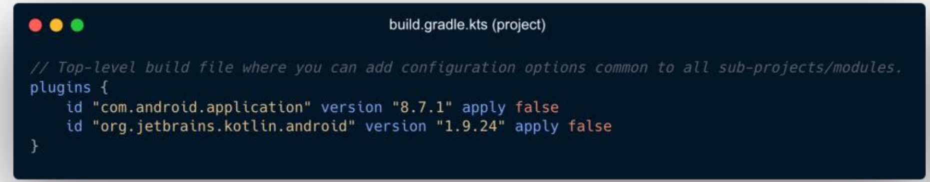
- Source code and resources
- Source code from main source set available everywhere
- Resources can be overridden in different source set



```
MyApp/                              Project
├─ build.gradle
├─ settings.gradle
└─ app/                             Module
   ├─ build.gradle
   ├─ build/
   ├─ libs/
   └─ src/
      └─ main/                      Sourceset
         ├─ java/
         │  └─ com.example.myapp/
         ├─ res/
         │  ├─ drawable/
         │  ├─ layout/
         │  └─ ...
         └─ AndroidManifest.xml
```

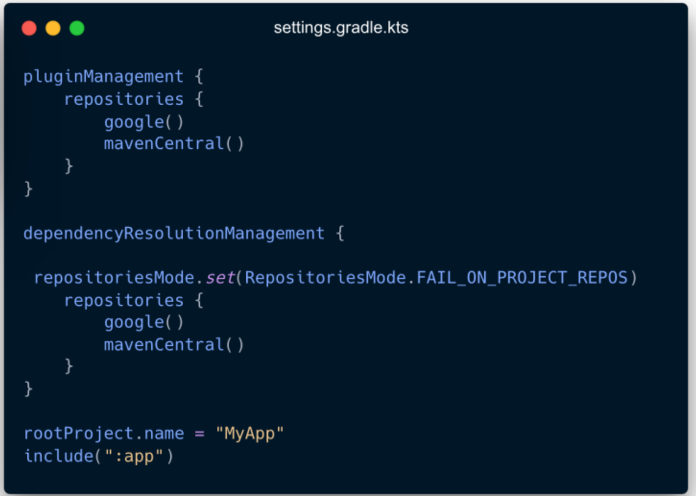
Gen

Project level files: build.gradle and settings.gradle



build.gradle.kts (project)

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    id "com.android.application" version "8.7.1" apply false
    id "org.jetbrains.kotlin.android" version "1.9.24" apply false
}
```



settings.gradle.kts

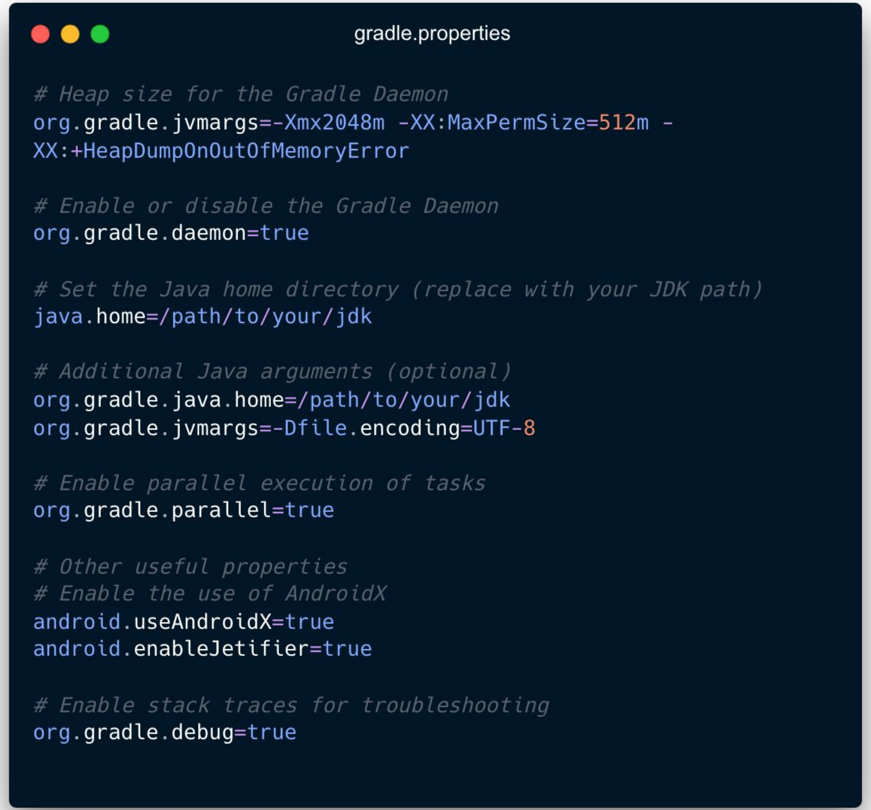
```
pluginManagement {
    repositories {
        google()
        mavenCentral()
    }
}

dependencyResolutionManagement {

 repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
    }
}

rootProject.name = "MyApp"
include(":app")
```

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

- List of modules to build

Gen

Project level files: gradle.properties



gradle.properties

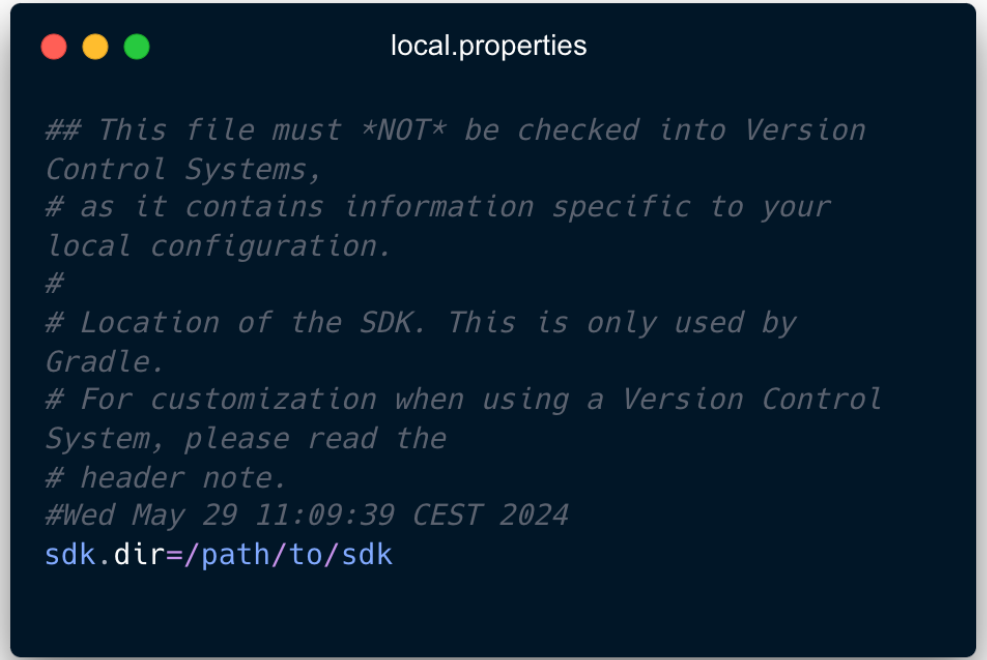
```
# Heap size for the Gradle Daemon
org.gradle.jvmargs=-Xmx2048m -XX:MaxPermSize=512m -
XX:+HeapDumpOnOutOfMemoryError

# Enable or disable the Gradle Daemon
org.gradle.daemon=true

# Set the Java home directory (replace with your JDK path)
java.home=/path/to/your/jdk

# Additional Java arguments (optional)
org.gradle.java.home=/path/to/your/jdk
org.gradle.jvmargs=-Dfile.encoding=UTF-8

# Enable parallel execution of tasks
org.gradle.parallel=true

# Other useful properties
# Enable the use of AndroidX
android.useAndroidX=true
android.enableJetifier=true

# Enable stack traces for troubleshooting
org.gradle.debug=true
```

- **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

Gen

Project level files: local.properties

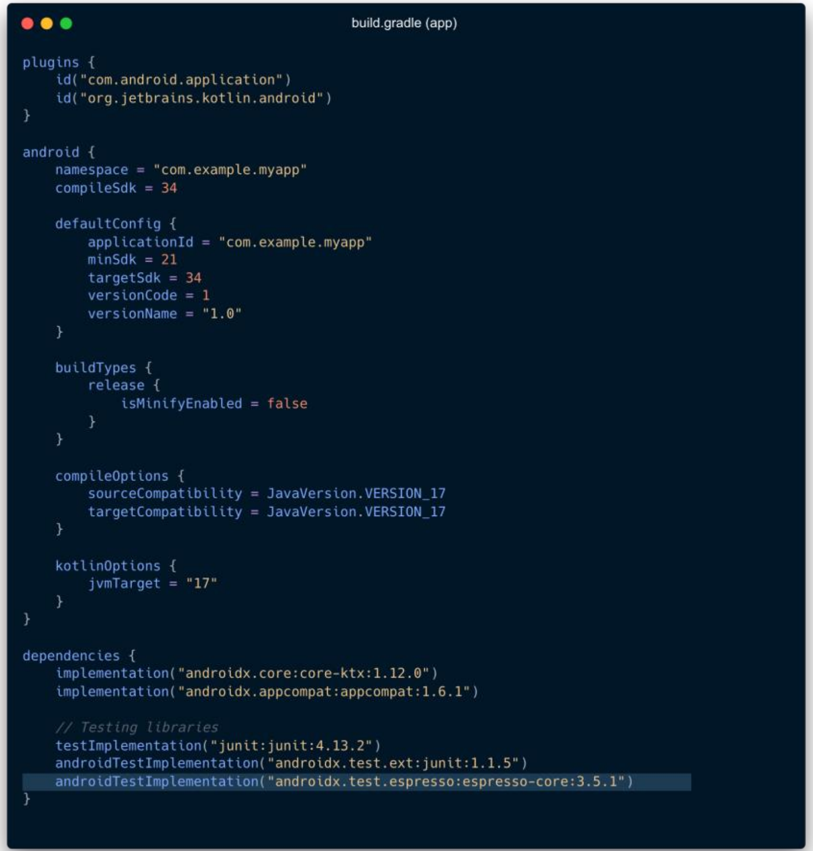
- Contains paths to SDK and NDK
- Can't be shared between developers
- Generated during build, do not modify it manually
- Do not include this file in submitted projects
- .gitignore


local.properties

```
## This file must *NOT* be checked into Version Control Systems,
# as it contains information specific to your local configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please read the
# header note.
#Wed May 29 11:09:39 CEST 2024
sdk.dir=/path/to/sdk
```

Module level 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
- Documentation



```
plugins {
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
}

android {
    namespace = "com.example.myapp"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.myapp"
        minSdk = 21
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_17
        targetCompatibility = JavaVersion.VERSION_17
    }

    kotlinOptions {
        jvmTarget = "17"
    }
}

dependencies {
    implementation("androidx.core:core-ktx:1.12.0")
    implementation("androidx.appcompat:appcompat:1.6.1")

    // Testing libraries
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

Module level files: libs/

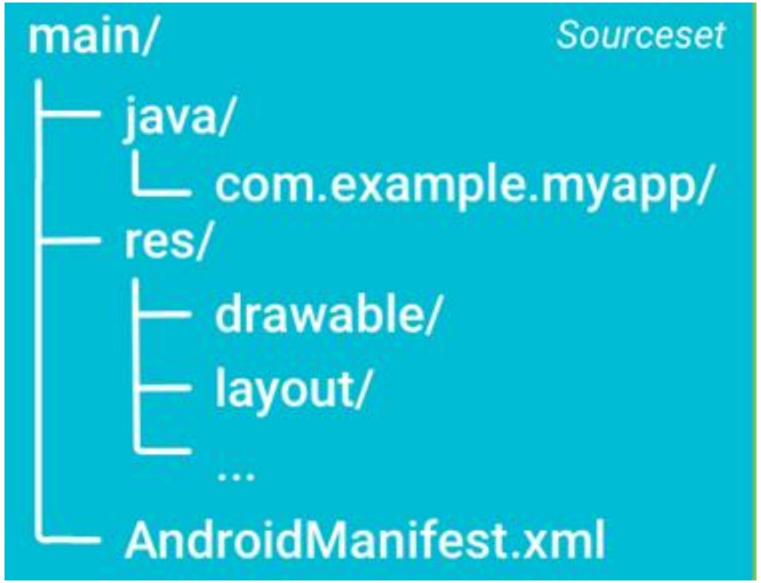
- *.jar libraries
- If it is possible use library as gradle dependency

Module level 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**



```
main/                          Sourceset
├── java/
│   └── com.example.myapp/
├── res/
│   ├── drawable/
│   ├── layout/
│   └── ...
└── 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 vectors over bitmaps**

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**

Demo: Hello World

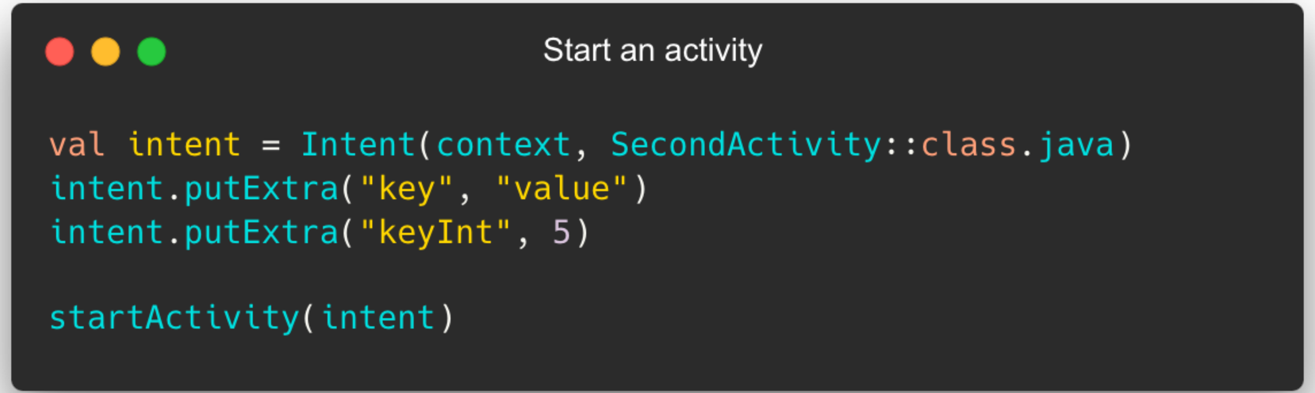
Activity & Back stack

Activity

- Presentation layer of application
- Only UI component
- Contains Views, Fragments, Composables
- Every activity must be defined in manifest
- Runs on UI (Main) 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


Start an activity

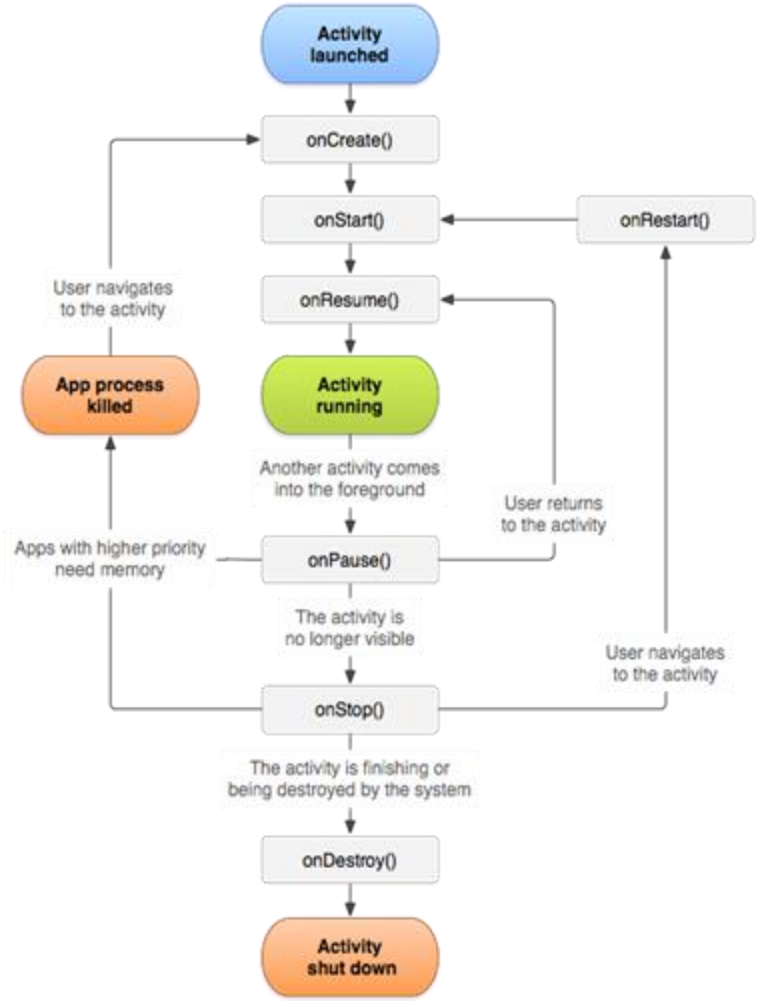
```
val intent = Intent(context, SecondActivity::class.java)
intent.putExtra("key", "value")
intent.putExtra("keyInt", 5)

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

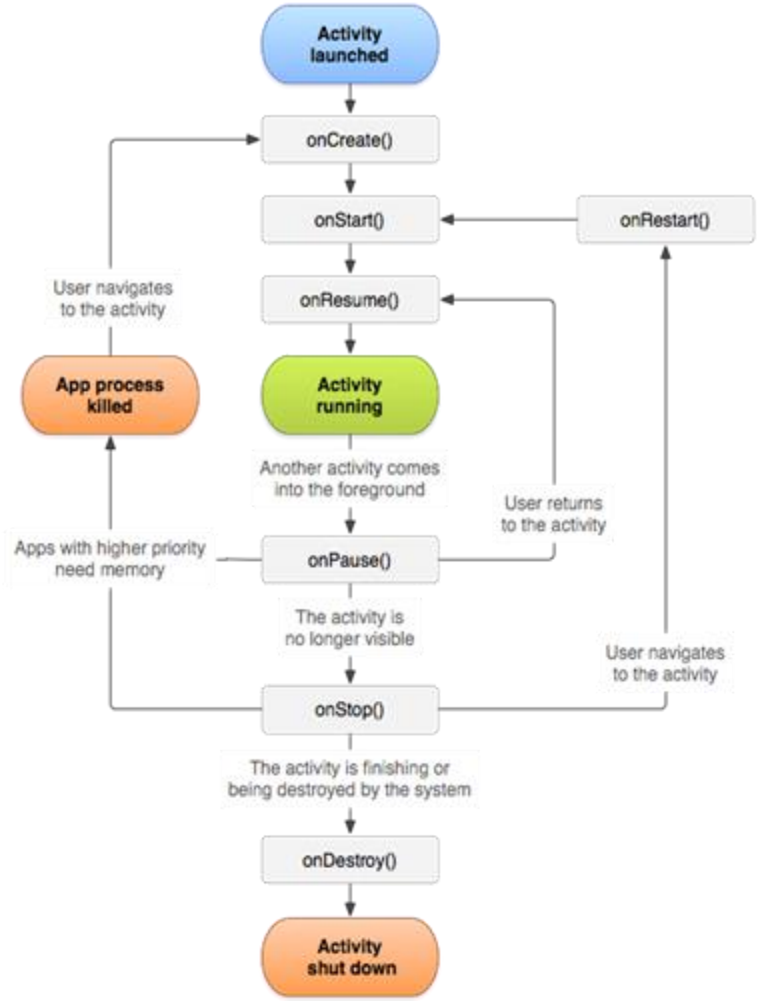
Activity - states

- Documentation
- **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


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

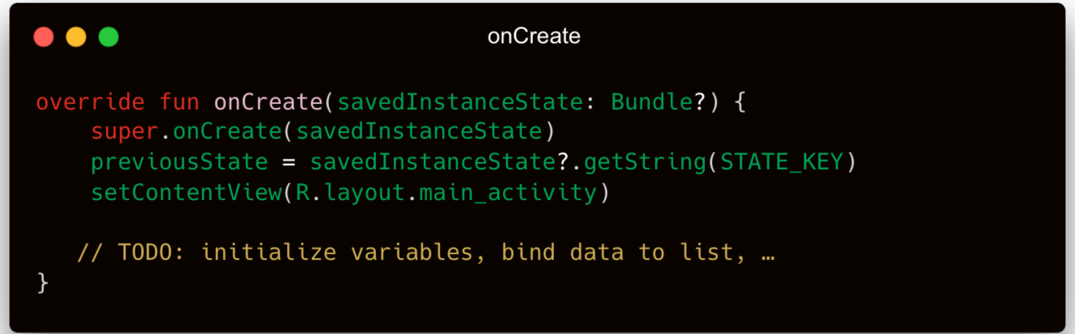
Activity - states

- **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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

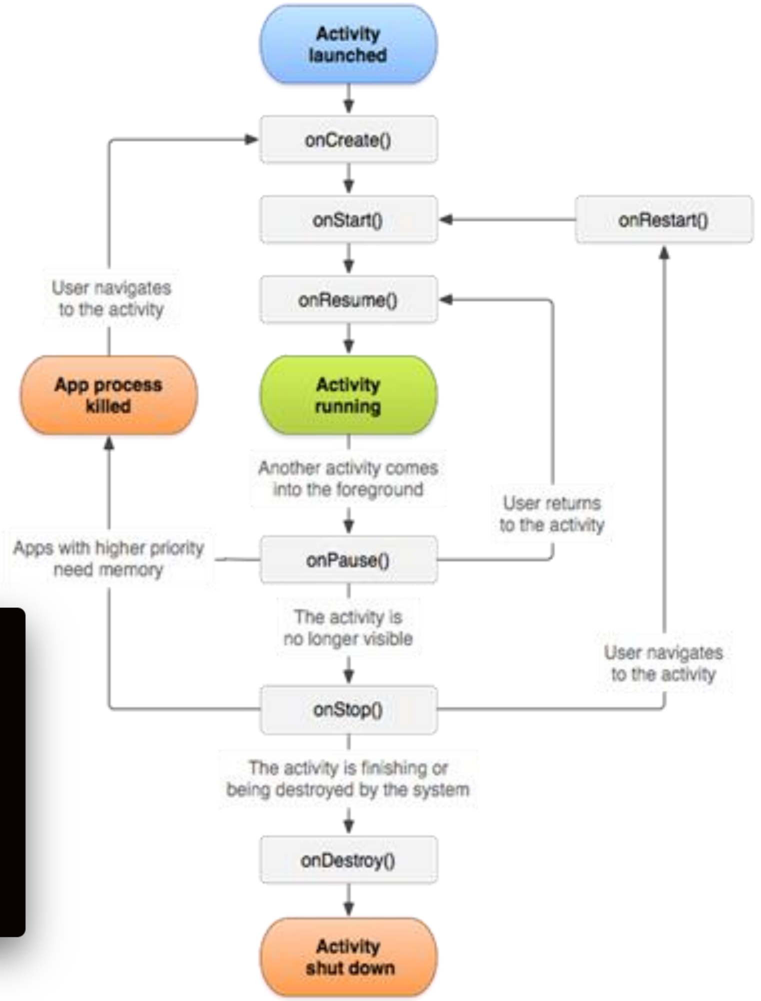
Activity#onCreate(Bundle)

- **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()



onCreate

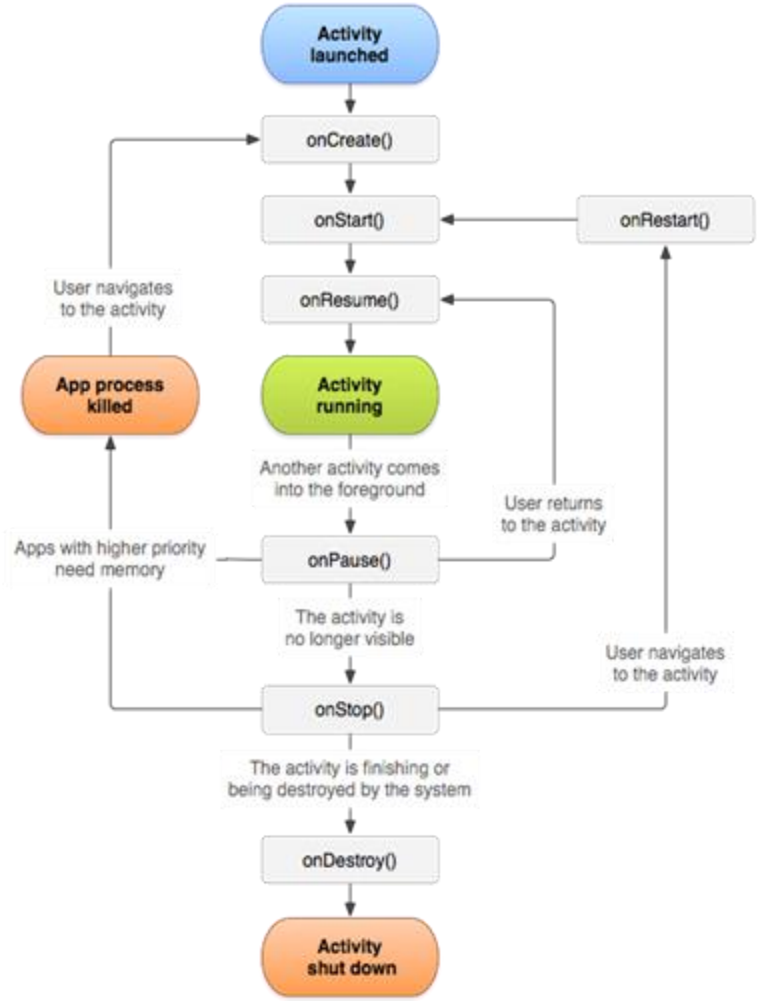
```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    previousState = savedInstanceState?.getString(STATE_KEY)
    setContentView(R.layout.main_activity)

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



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

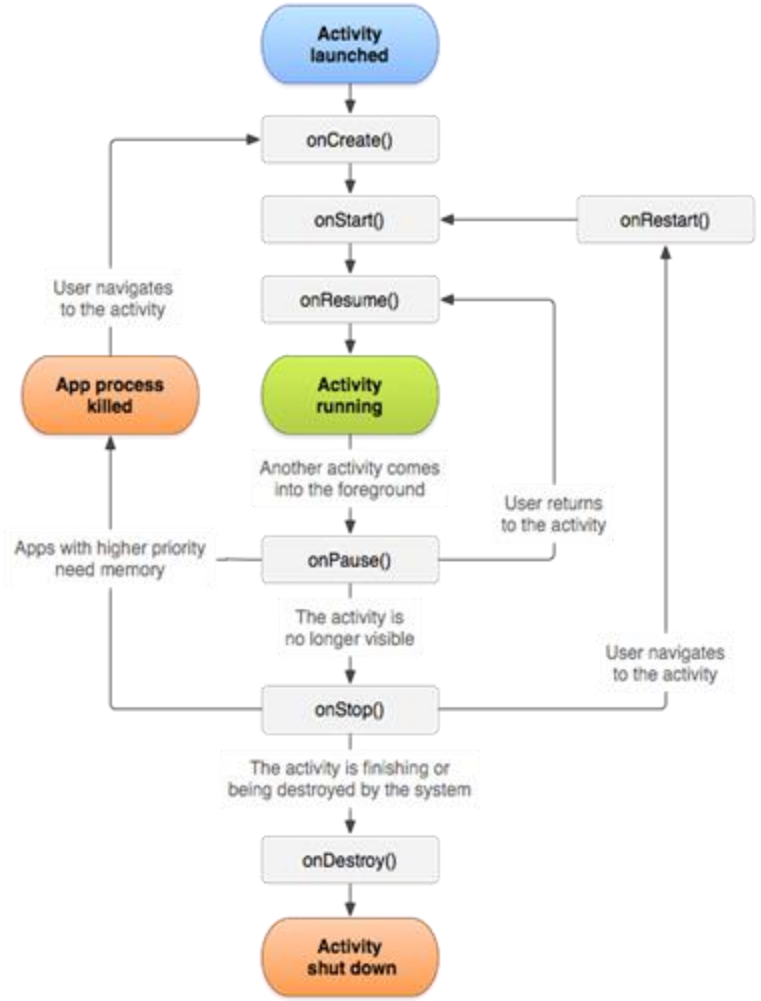
Activity#onStart()

- **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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

Activity#onResume()

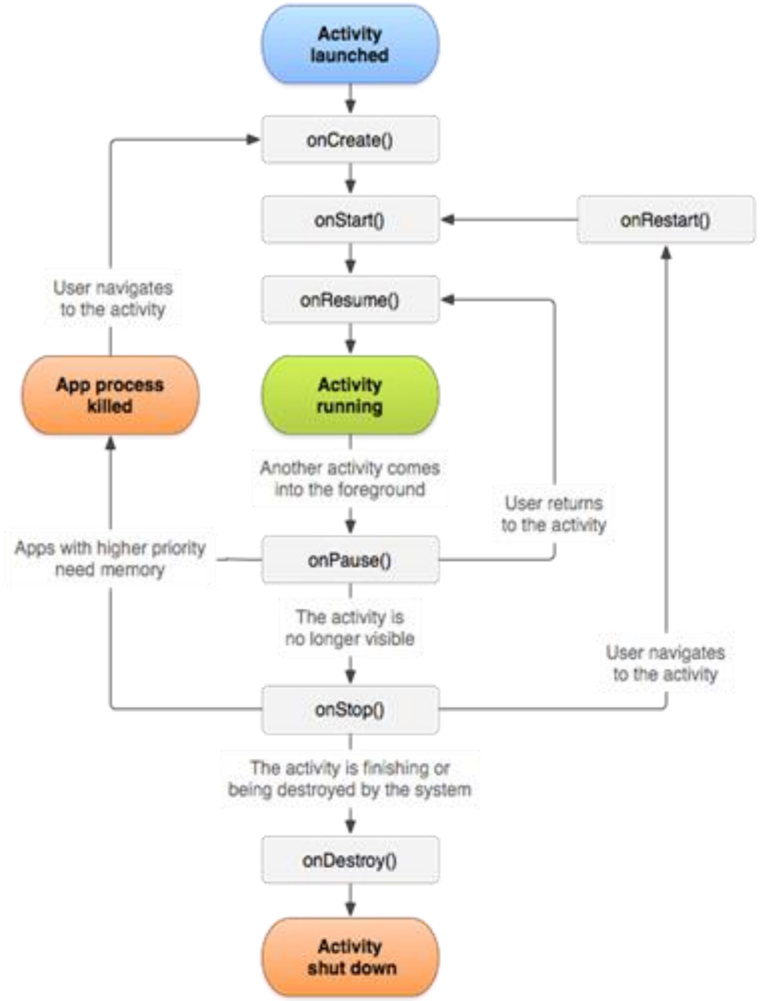
- Called just before activity start interacts with user
- Activity is on top of activity stack
- Run stuff for user
- Always followed by onPause()


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

Gen

Activity#onPause()

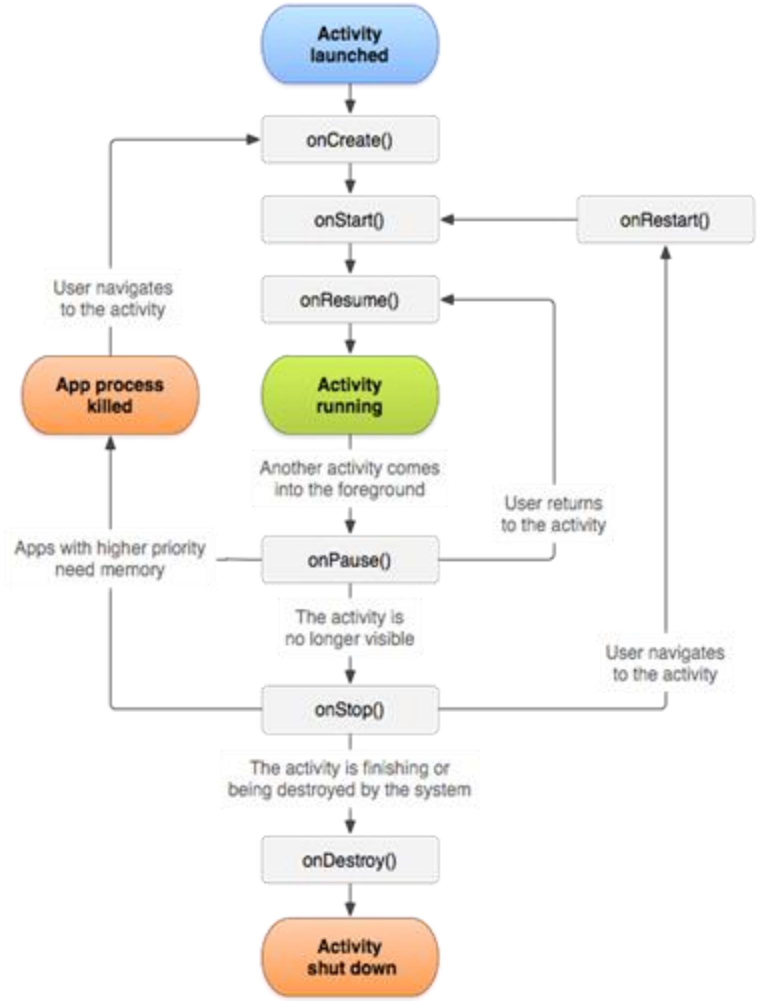
- System is about to resume another activity
- Stop animations and CPU intensive stuff
- Should be very fast, because another activity onResume() waits until this finishes
- Followed by
    - onResume() if the activity returns back to the front
    - onStop() if became invisible to the user
- Activity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


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

Gen

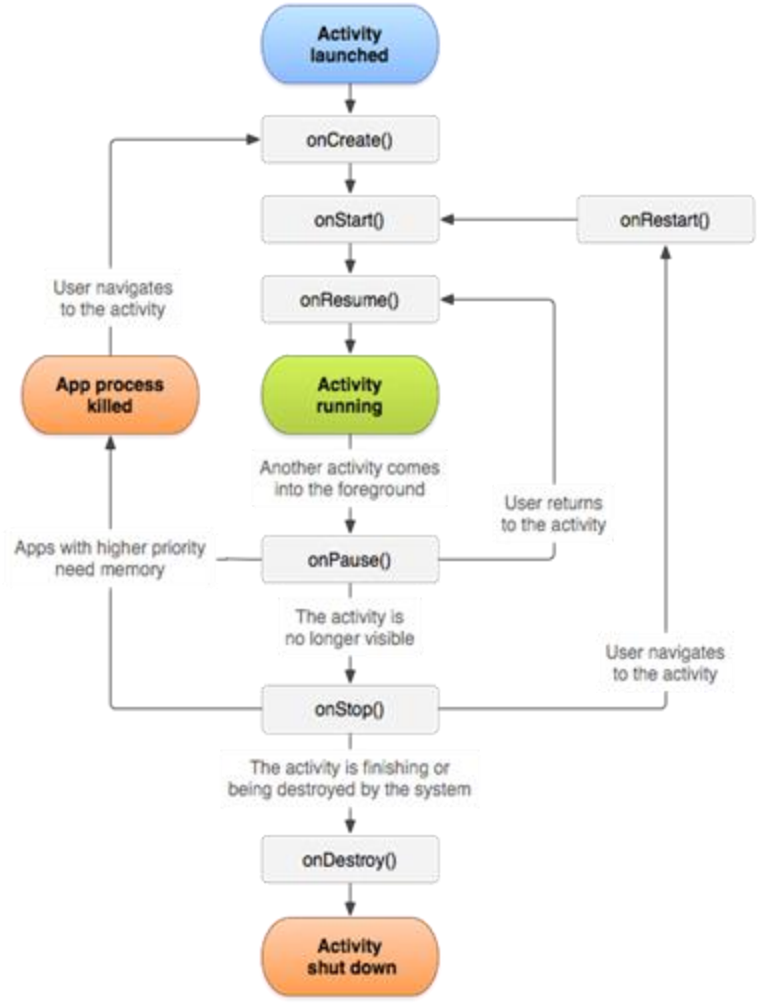
Activity#onStop()

- 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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

Activity#onDestroy()

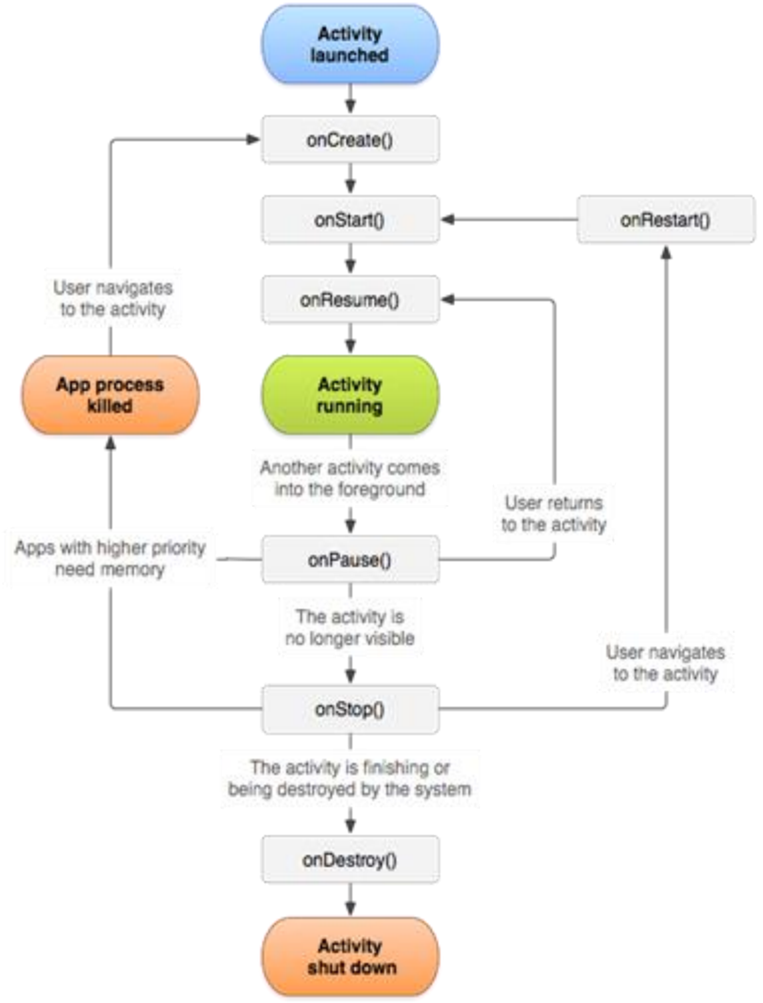
- Called before activity is destroyed
- Activity is finished by #finish() method
- System needs more resources (RAM)



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

Activity#onRestart()

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


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

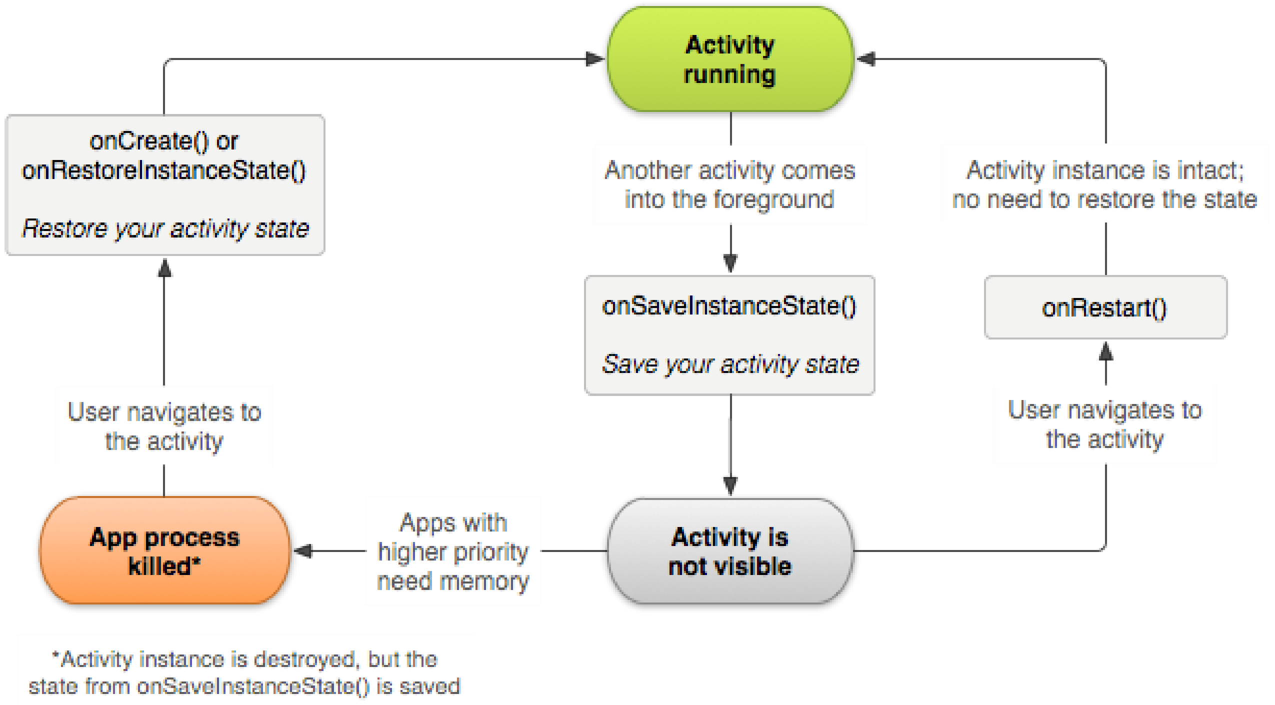
Bundle

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

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

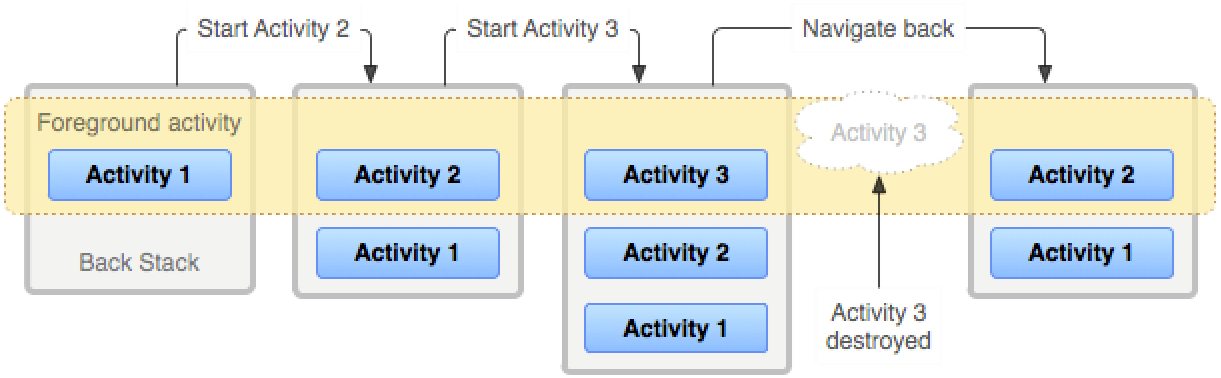

Activity running
onCreate() or onRestoreInstanceState()
Restore your activity state
Another activity comes into the foreground
Activity instance is intact; no need to restore the state
onSaveInstanceState()
Save your activity state
onRestart()
User navigates to the activity
User navigates to the activity
App process killed*
Apps with higher priority need memory
Activity is not visible
*Activity instance is destroyed, but the state from onSaveInstanceState() is saved

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)

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

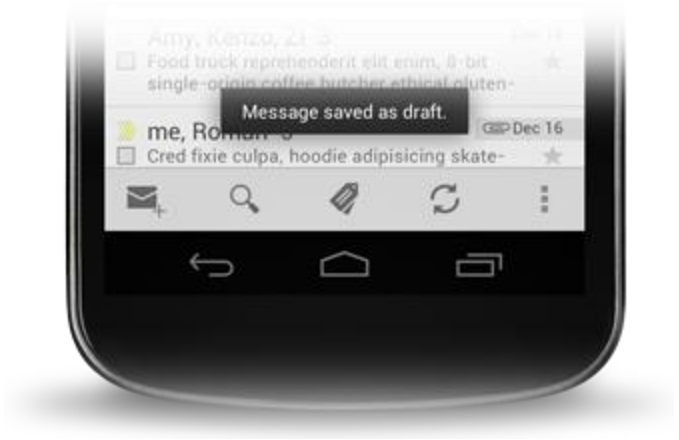
- **Sometimes is necessary to change behavior of back stack**
- **Manifest attributes**
  - `taskAffinity`
  - `launchMode`
  - `allowTaskReparenting`
  - `clearTaskOnLaunch`
  - `alwaysRetainTaskState`
  - `finishOnTaskLaunch`
- **Intent flags**
  - `FLAG_ACTIVITY_NEW_TASK`
    - Start activity in new 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`


Amy, Kenzo, 21 3
Food truck reprehenderit elit enim, 8-bit
single-origin coffee butcher ethical gluten-
Message saved as draft.
me, Roman 3
Dec 16
Cred fixie culpa, hoodie adipisicing skate-

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

User Interface

Approaches to writing UI

- XML
- Jetpack Compose

XML: Layouts

- Definition of UI
- Used for Activity or Fragment
- Extends ViewGroup
- Defined in XML or programmatically
- Folder res/layout
- *Options*: FrameLayout, LinearLayout, RelativeLayout , TableLayout, GridLayout, ConstraintLayout (Google IO 2016, Available as support library)

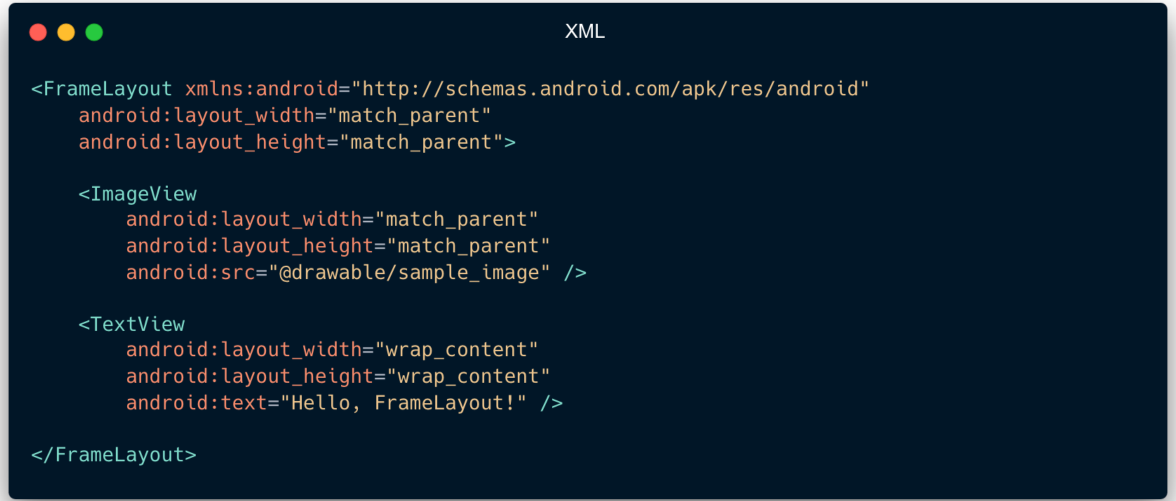
XML: Binding between layouts and java

- 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)

Layout - FrameLayout

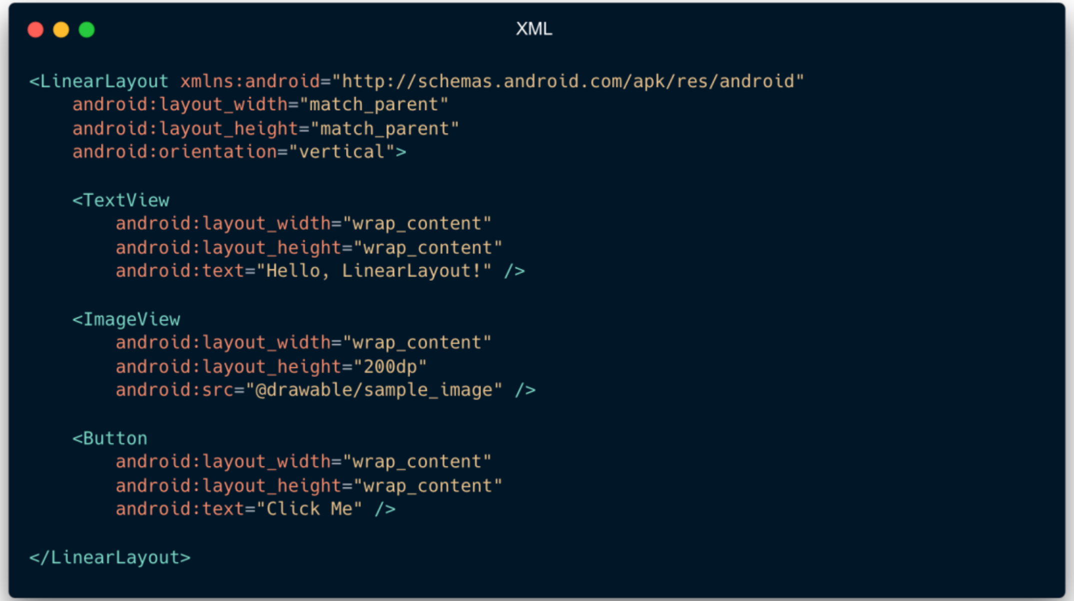
- Places all items in top left corner
- Usage as placeholder for other view/fragment
- Fast



```xml
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/sample_image" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, FrameLayout!" />

</FrameLayout>
```

Layout - LinearLayout

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

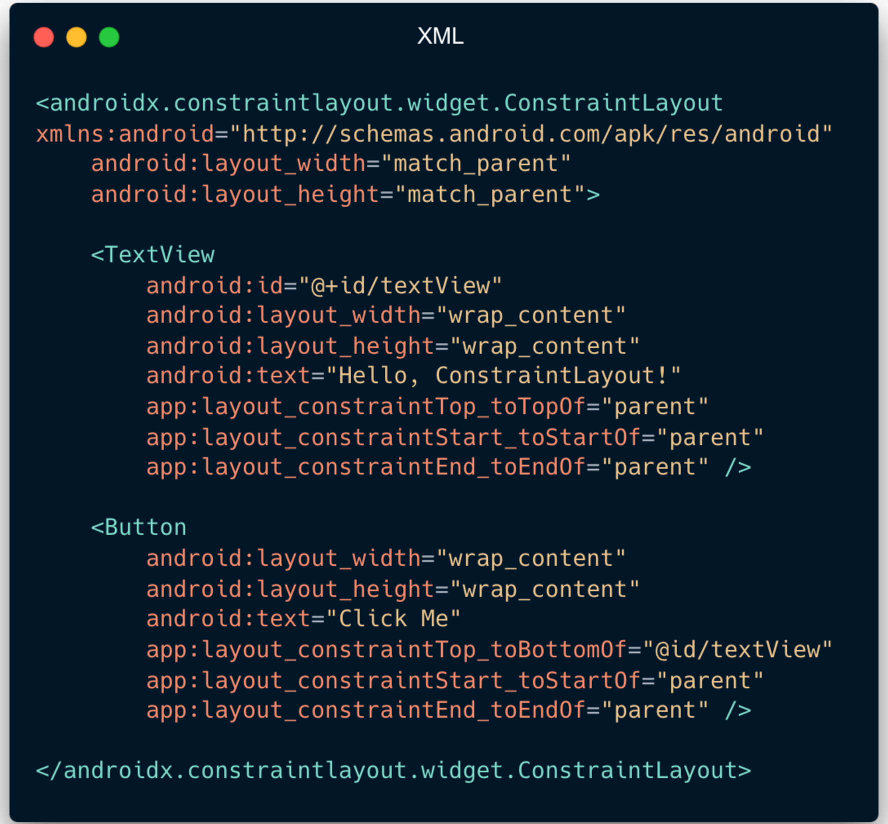


```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, LinearLayout!" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="200dp"
        android:src="@drawable/sample_image" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me" />

</LinearLayout>
```

Layout - Constraint layout

- “Extended relative layout”
- Constraint is connection or alignment to another view/parent/guideline
- Recommended today
- Documentation
- Available as dependency:

`"androidx.constraintlayout:constraintlayout"`



```xml
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, ConstraintLayout!"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me"
        app:layout_constraintTop_toBottomOf="@id/textView"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Widgets – UI elements

- Extend 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
  - Specific dimension

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

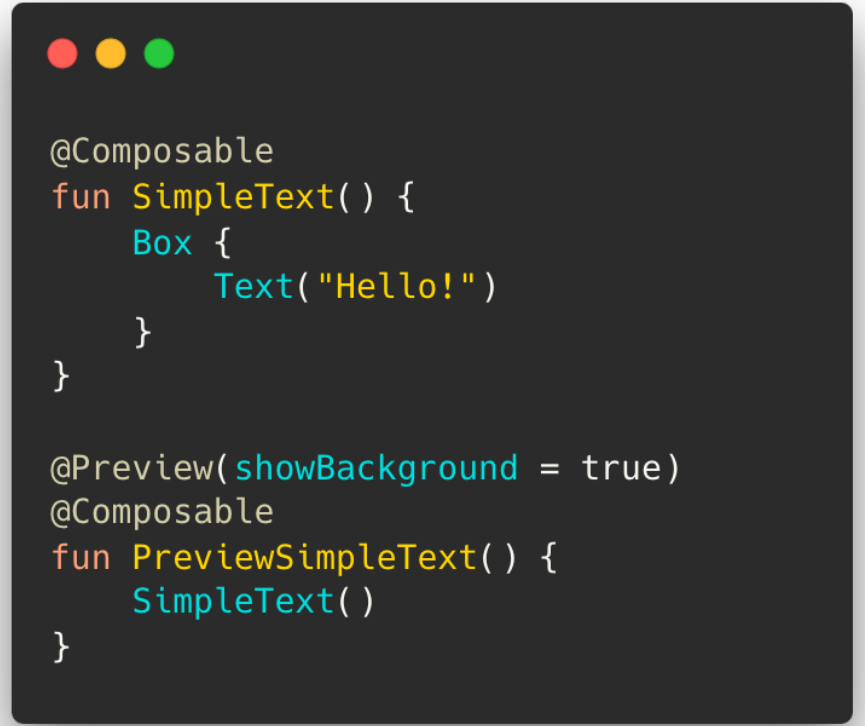
Navigation

- *Navigation* = how user moves between "screens"
- *Options*:
  - Between activities: Intents
  - Between fragments: fragment transactions, navigation component

Demo: Simple UI in XML

Jetpack Compose

- Modern toolkit for building native UI
- Intuitive Kotlin API
- UI in Kotlin instead of XML
- Reusability and modularity of the UI components
- Emphasizes "what" over "how"
- Previews
- **Any UI element in Compose is a Composable function**
- Documentation

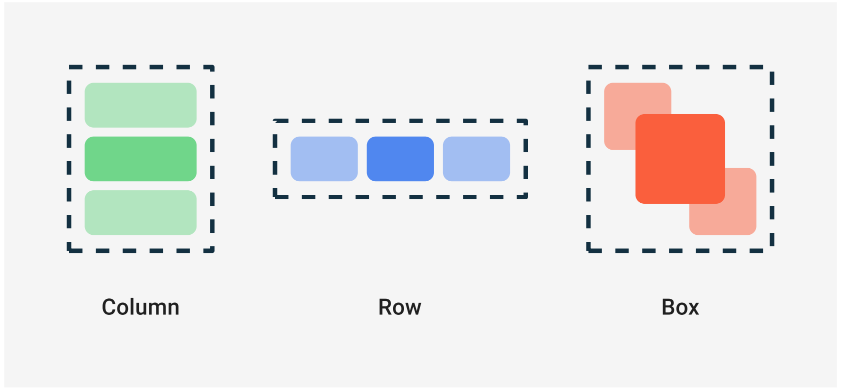


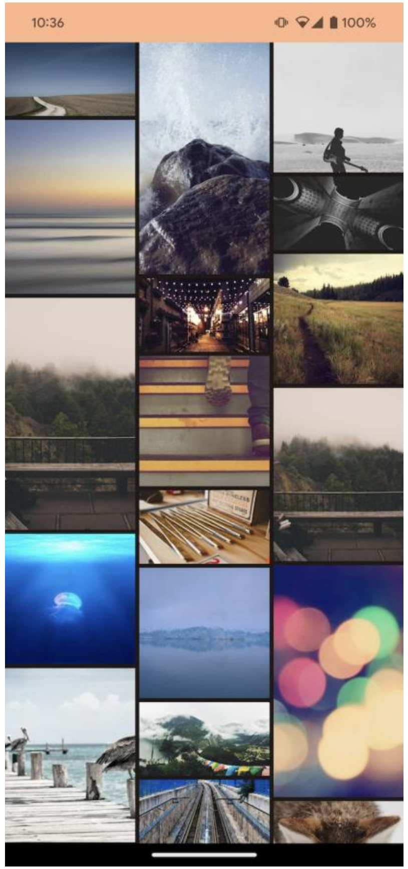
```
@Composable
fun SimpleText() {
    Box {
        Text("Hello!")
    }
}

@Preview(showBackground = true)
@Composable
fun PreviewSimpleText() {
    SimpleText()
}
```

Gen

Jetpack Compose: Layouts

- Box
- Column, Row
- LazyColumn, LazyRow
- LazyVerticalGrid, LazyHorizontalGrid
- LazyVerticalStaggeredGrid, LazyHorizontalStaggeredGrid (experimental)
- Scaffold

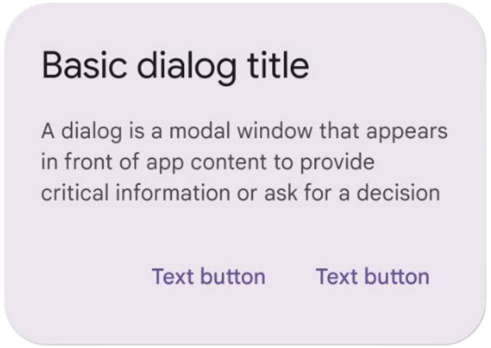




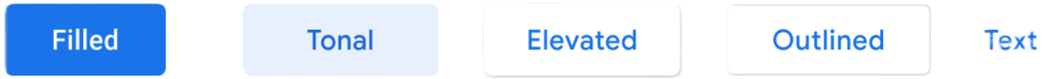
LazyVerticalStaggeredGrid

Jetpack Compose: Basic components

- Text
- ClickableText
- Image
- Button
- FloatingActionButton
- Spacer, Divider
- CircularProgressIndicator, LinearProgressIndicator
- AlertDialog, Popup
- Snackbar
- https://m3.material.io/components

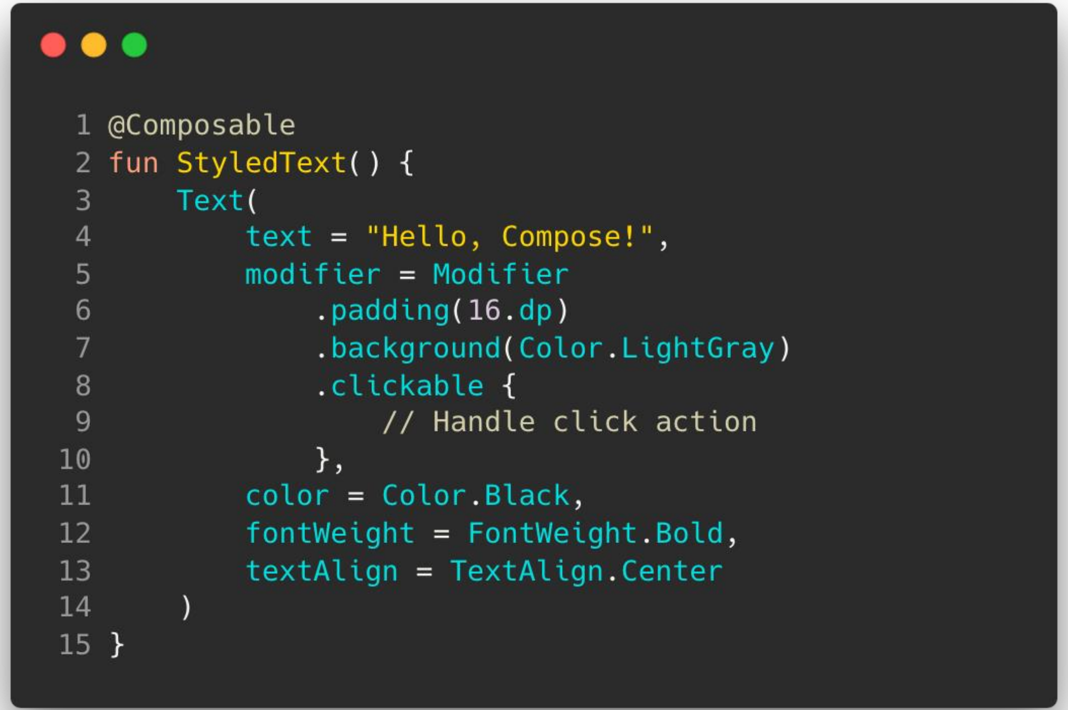

Basic dialog title

A dialog is a modal window that appears in front of app content to provide critical information or ask for a decision



Jetpack Compose: Modifiers

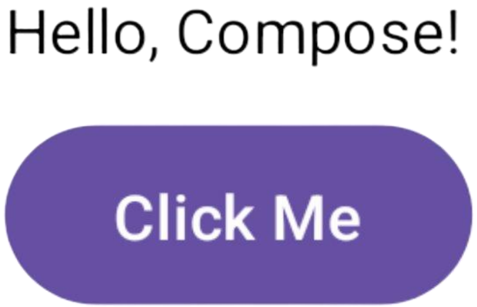
- Layout, styling, interactivity
- Chainable
- Order matters!
- Examples:
  - padding()
  - background()
  - align()
  - clickable()
  - scrollable()
  - ...

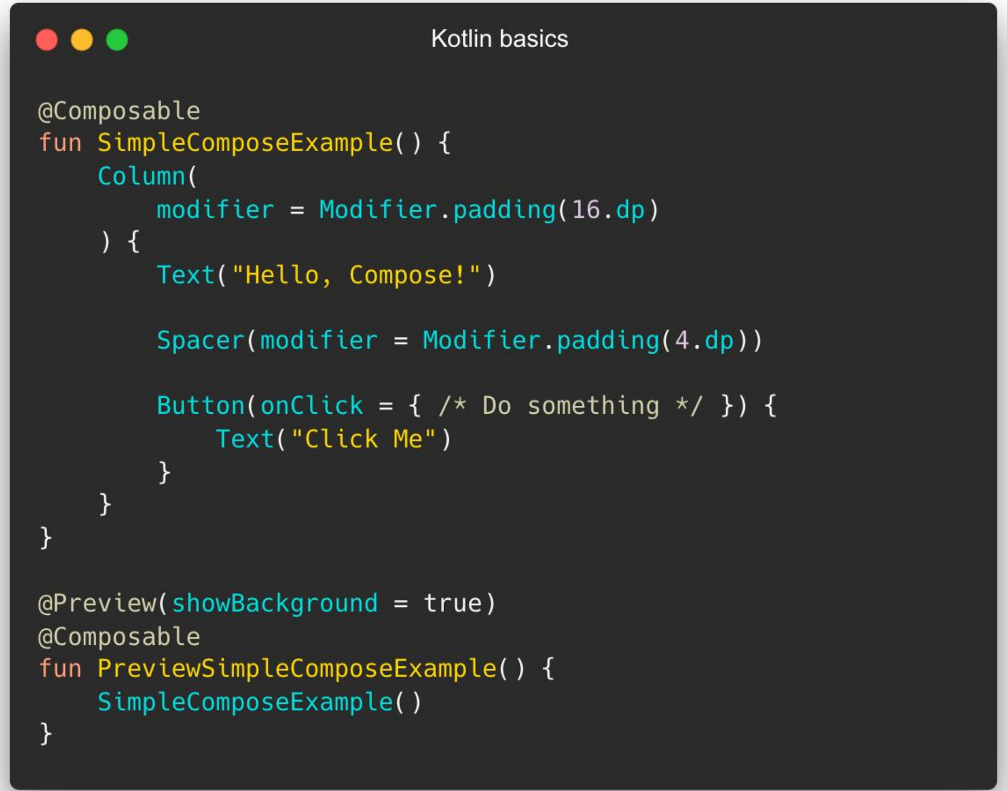


```
@Composable
fun StyledText() {
    Text(
        text = "Hello, Compose!",
        modifier = Modifier
            .padding(16.dp)
            .background(Color.LightGray)
            .clickable {
                // Handle click action
            },
        color = Color.Black,
        fontWeight = FontWeight.Bold,
        textAlign = TextAlign.Center
    )
}
```

Gen

Jetpack Compose: Basic components (example)

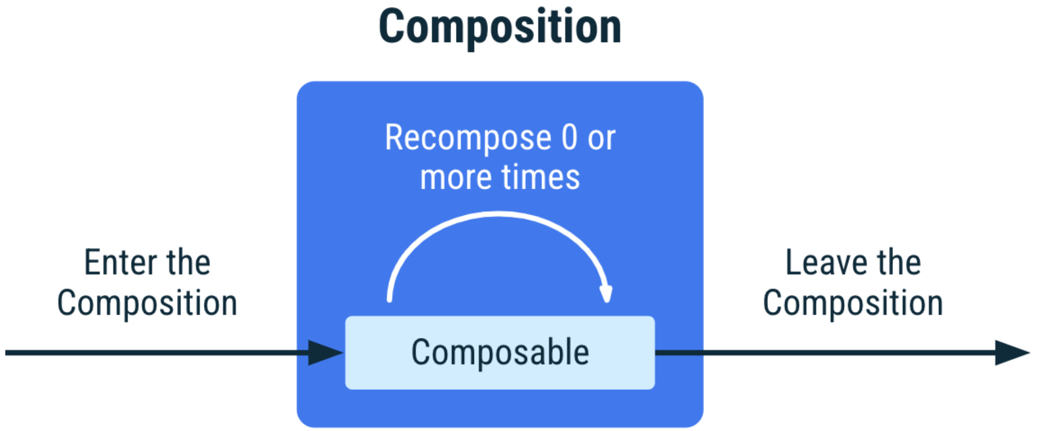



Kotlin basics

```
@Composable
fun SimpleComposeExample() {
    Column(
        modifier = Modifier.padding(16.dp)
    ) {
        Text("Hello, Compose!")

        Spacer(modifier = Modifier.padding(4.dp))

        Button(onClick = { /* Do something */ }) {
            Text("Click Me")
        }
    }
}

@Preview(showBackground = true)
@Composable
fun PreviewSimpleComposeExample() {
    SimpleComposeExample()
}
```

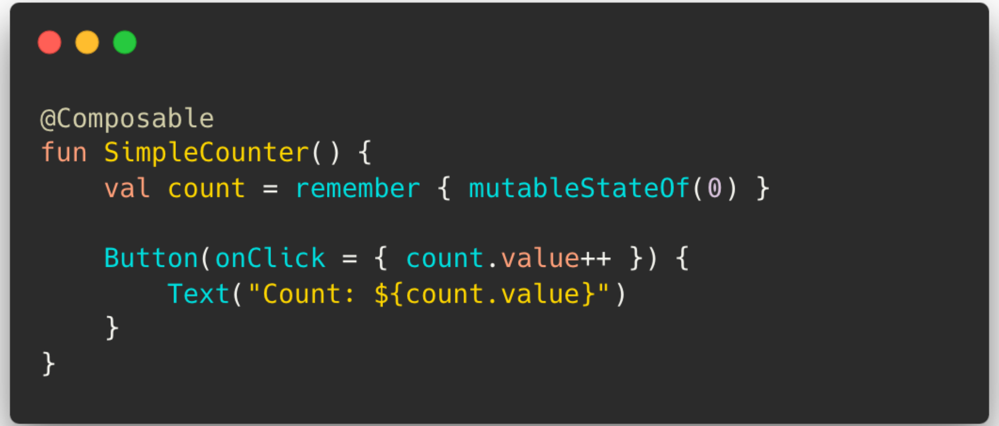
Jetpack Compose: Lifecycle

- **Key stages**: Composition, Recomposition, Disposal
- **Side-effect:** Change to the state of the app that happens outside the scope of a composable function
- **Effects**: Code that is triggered in response to changes in state or composition
  - *LaunchedEffect, SideEffect, DisposableEffect, ...*



Jetpack Compose: State

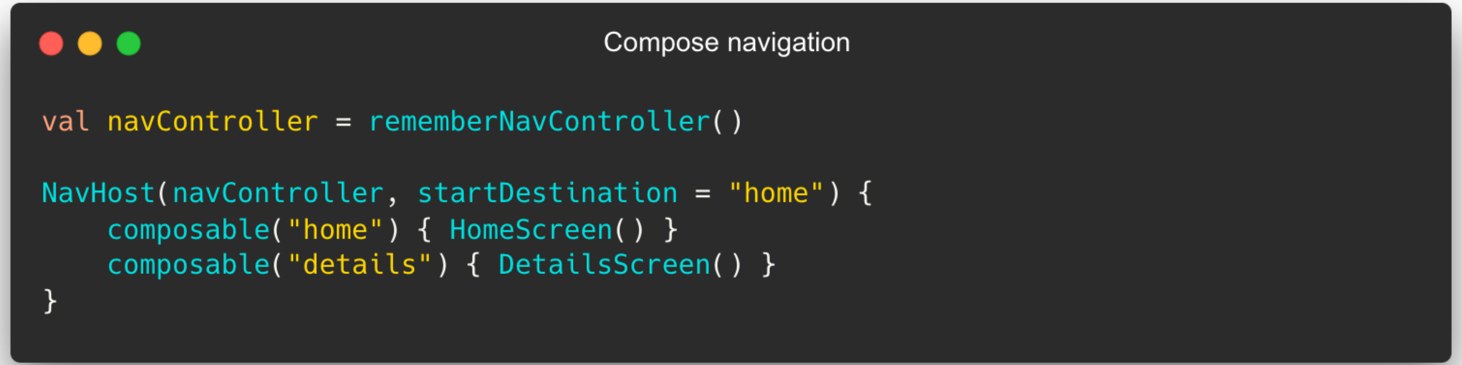
- State = mutable data that can affect UI
- Changes in state trigger recomposition
- remember, rememberSaveable
- State hoisting (careful usage)



```kotlin
@Composable
fun SimpleCounter() {
    val count = remember { mutableStateOf(0) }

    Button(onClick = { count.value++ }) {
        Text("Count: ${count.value}")
    }
}
```

Navigation

- Navigation component
- Destinations are Composables
- *NavController*: Holds navigation graphs and provides API to move between Composables
- *NavHost*: Manages which composable is currently displayed based on NavController
- Documentation



Compose navigation

```kotlin
val navController = rememberNavController()

NavHost(navController, startDestination = "home") {
    composable("home") { HomeScreen() }
    composable("details") { DetailsScreen() }
}
```

Gen

Demo: Simple UI in Compose

Thank you

Lukáš Prokop
Simona Kurňavová

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

Gen™