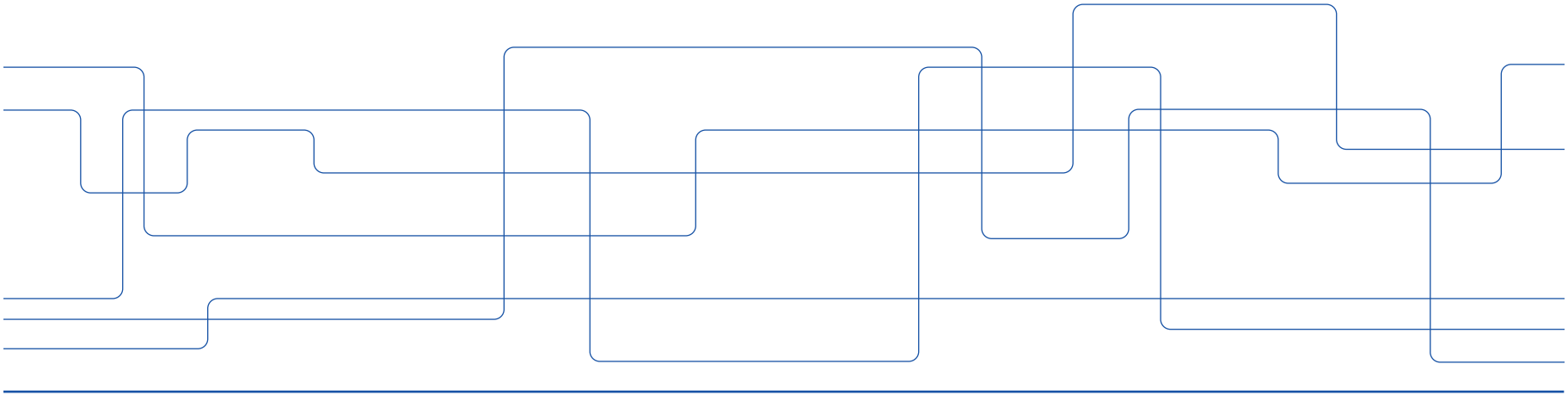
Mobile Application Programming

Anders Västberg <vastberg@kth.se>



Two Major Platforms

- Android’s market share in the world is 85% while iOS has 15%
- In US it is 40% for Android and 60% for iOS.
- In EU it is 80% for Android and 20% for iOS.
- The revenue from Mobile Apps in Apple App Store is about twice that of Google Play
- To maximize profit and coverage, both platforms needs to be covered

iOS Features

- Apple propriety OS
- Apple controls the development of hardware
  - Better integration with external devices
  - Less quality problems of the hardware due to much fewer models
- UNIX kernel based on Darwin
  - Developed from NextStep that was in itself developed from BSD.
- Limited possibilities to customize the interface (unless jailbreak)
- Better security compared to Android
  - Controlled app store
  - Secure architecture

Android Features

- Open source software
  - Though google apps are propriety software
- Many different manufactures of hardware
- Based on Linux kernel
- Large possibility to customize the interface
- Google play is the main app store
  - Downloads from other sources are possible
- Less secure due to
  - Many different manufacturers of hardware
  - The possibility to download apps from other sources
  - That the OS is open to customize

General Development Tools

- Two major platforms
  - develop two different native apps
  - web application
- Third alternative – use frameworks
- React Native
  - Uses JavaScript
  - Based on the React JavaScript library for building user interfaces
  - Can wrap native code written in Java for Android or Swift for iOS
- Flutter
  - Uses Dart programming language
  - Two versions of Widgets
    - > *Material Design (Android) and Cupertino (iOS)*

Programming Tools for iOS Development

- Xcode IDE
  - Runs on MacOS
- Swift is the main programming language
- Introduced in 2014
- Replaces Objective-C
  - Can interact with existing code base
  - Supported on MacOS, Windows and Linux
  - Faster code than Objective-C

Structure of Android Apps [1]

- One or more interactive screens
- Written using Java or Kotlin programming languages and XML
- Uses the Android Software Development Kit (SDK)
- Uses Android Libraries and Android Application Framework
- Executed by the Android Runtime Virtual machine

Kotlin for Android

- Introduced 2016
- Google supports Kotlin as programming language for Android 2017
- Interaction with Java
- Object-oriented
- Statically typed
- Type inference
- Inspired by Scala

Kotlin Design Goals

- Designed to reduce code size
- NullPointer exceptions avoided
- Uses JVM
- Can compile to JavaScript
- Support for programmers in Android Studio

Kotlin Syntax

- Semicolons optional
- Variables and Constants

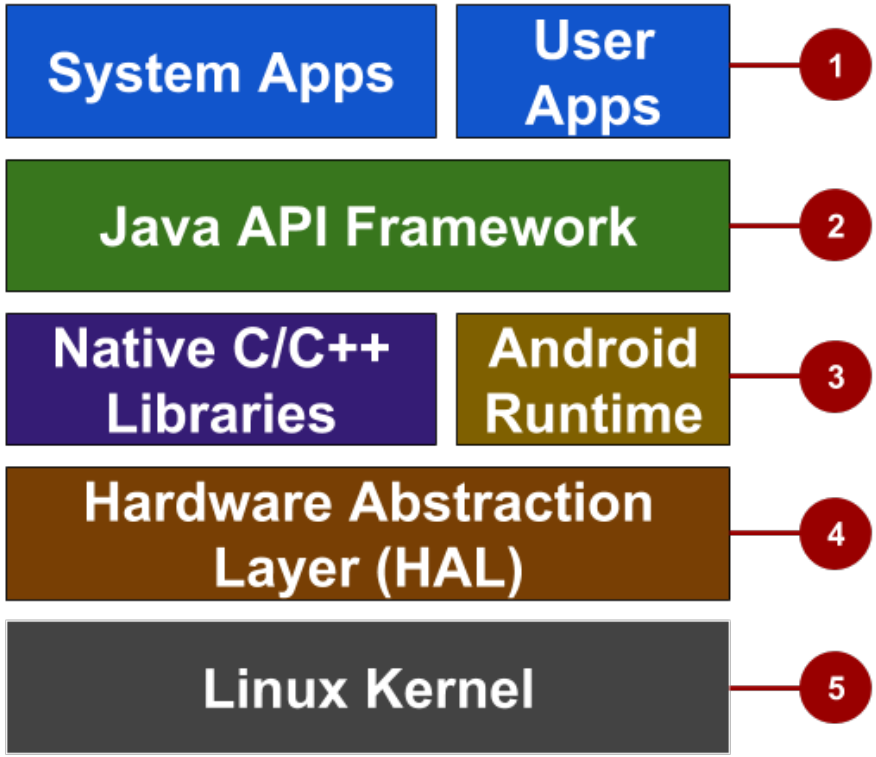
```
var name = “MyString” //variable

val name2 = “MyString” //constant
```

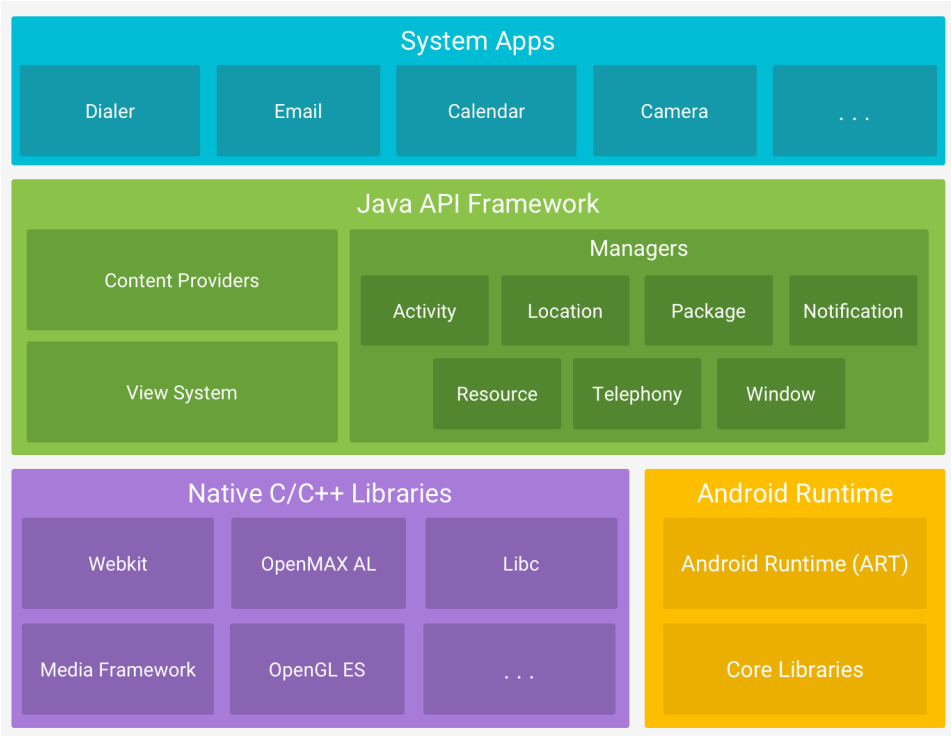
- Types need not be specified
- Static objects and functions can be declared outside classes
- Extension functions as in C#
- Classes are public by default
- Classes are final by default

Android Architecture [1]

1. Core system apps for email, SMS messaging, calendars, internet browsing, and contacts.
2. All features for Android development, such as UI components, resource management, and lifecycle management,
3. Each app runs in its own process and instance of the Android runtime. Native libraries written in C and C++ and are available to apps through the Java API framework.
4. Implements an interface for a specific type of hardware component, such as the camera or Bluetooth module.
5. For threading, low-level memory management, and other underlying functionality. Based on Linux 4.14+



Android Architecture [1]

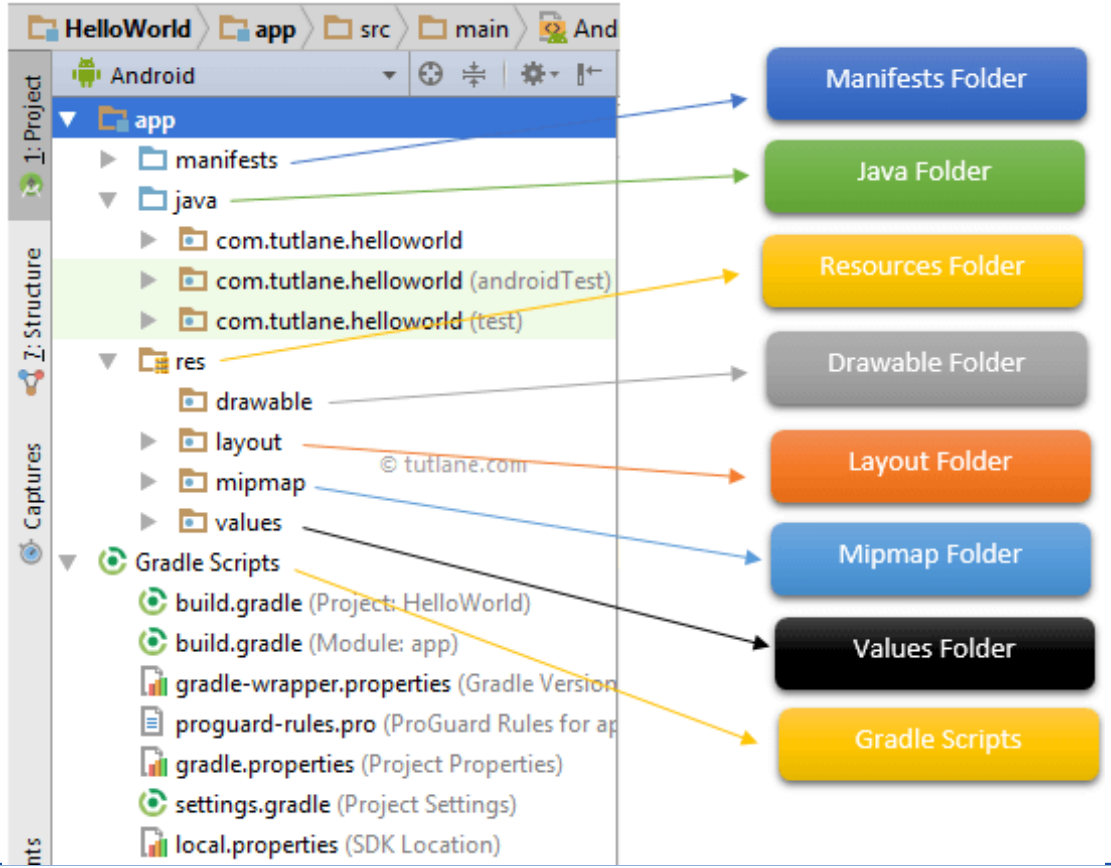


Android Software Developement Kit (SDK) [1]

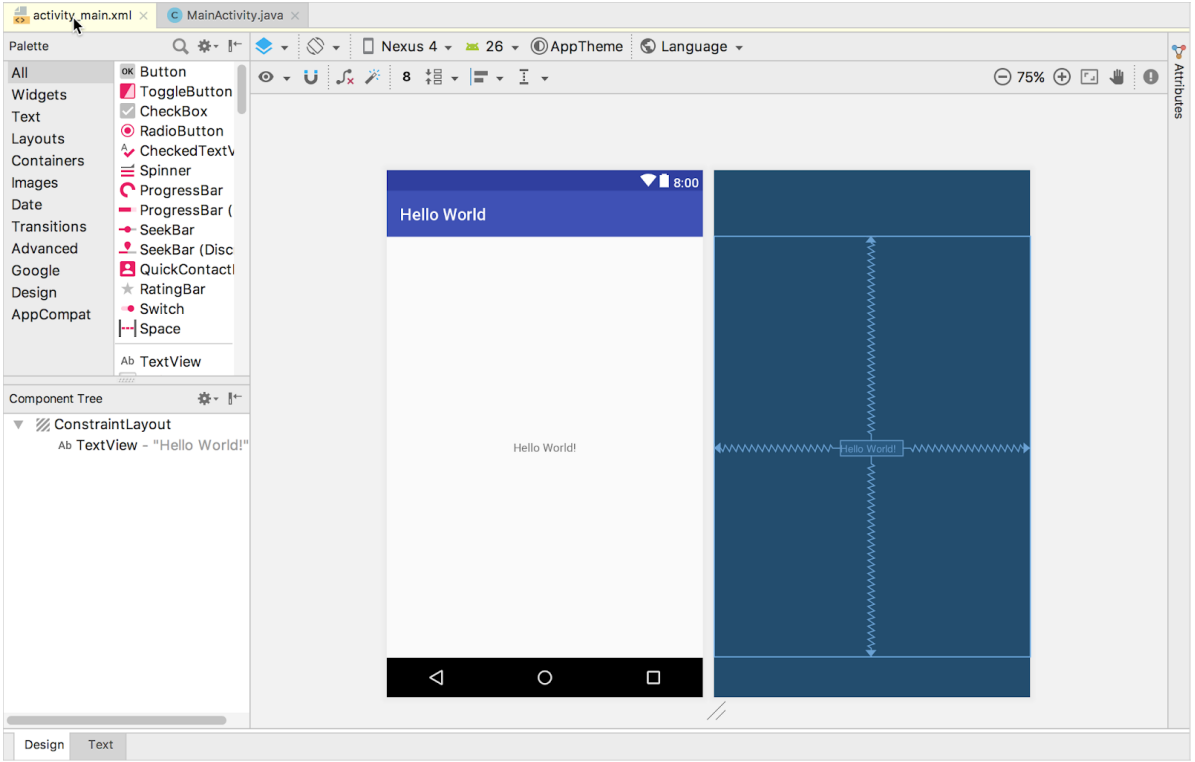
- Development tools (debugger, monitors, editors)
- Libraries (maps, wearables)
- Virtual devices (emulators)
- Documentation (developers.android.com)
- Sample code

Programming Tools for Android Development [2]

- Android Studio IDE



Editor Pane


activity_main.xml
MainActivity.java
Palette
All
Widgets
Text
Layouts
Containers
Images
Date
Transitions
Advanced
Google
Design
AppCompat
Button
ToggleButton
CheckBox
RadioButton
CheckedTextV
Spinner
ProgressBar
ProgressBar (
SeekBar
SeekBar (Disc
QuickContactI
RatingBar
Switch
Space
TextView
Component Tree
ConstraintLayout
TextView - "Hello World!"
Nexus 4
26
AppTheme
Language
75%
Attributes
8:00
Hello World
Hello World!
Hello World!
Design
Text

App Building Blocks [1]

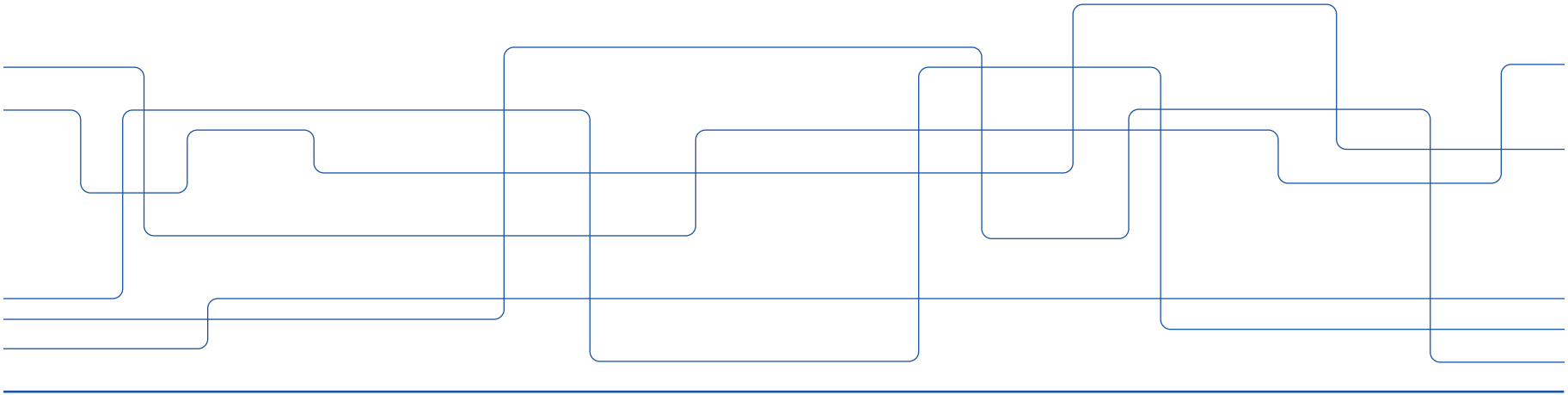
- Resources:
  - layouts, images, strings, colors as XML and media files
- Components:
  - activities, services, and helper classes as Java code
- Manifest:
  - Information about app for the runtime
- Build configuration:
  - APK (Android Package Kit) versions in Gradle config files

Challenges of Android Development [3]

- Performance
  - make your apps responsive and smooth
  - Varying networks speeds or networks failure
- Security
  - keep source code and user data safe
- Compatibility
  - Varying platform performance
  - Multiple screen sizes and resolutions
  - Difficulty to test applications fully
- Marketing
  - understand the market and your users

Android Programming [1]

Anders Västberg <vastberg@kth.se>

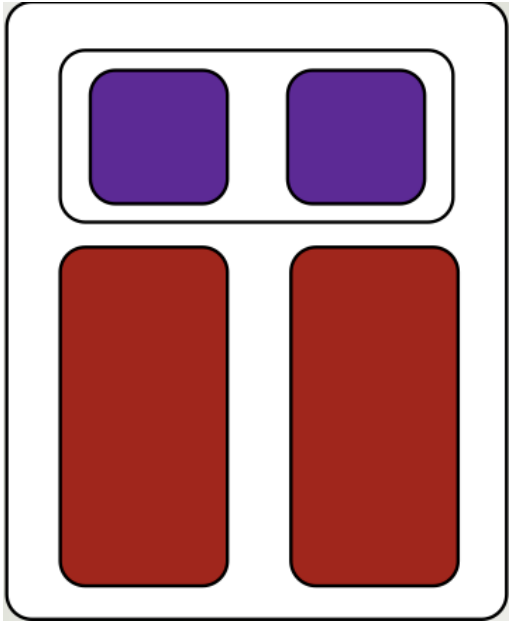


Views

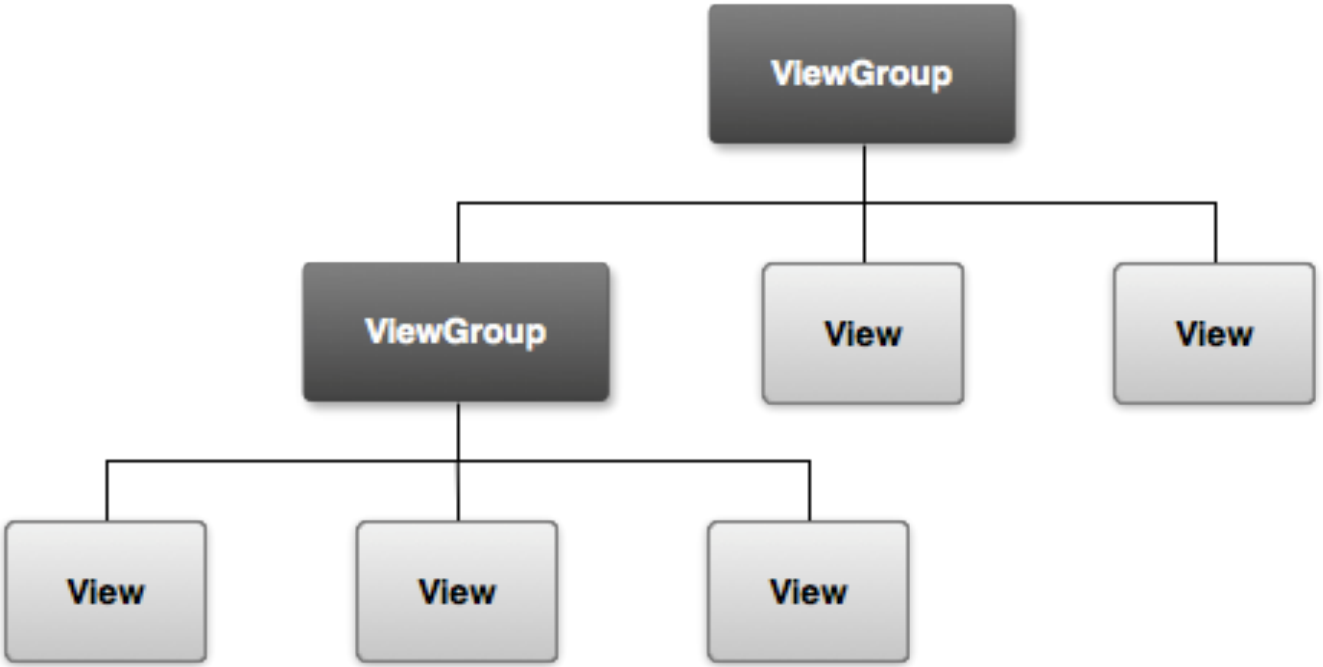
- Everything you see is a view
- View subclasses are basic user interface building blocks
  - display text (TextView), edit text (EditText)
  - Buttons (Button), menus, other controls
  - Scrollable (scrollView)
  - Group views (ConstraintLayout and LinearLayout)

View Hierarchy

- Views are the building blocks of Android User Interface
- Views can contain child views, which are positioned relative to the parent
- Specialized views: table views, image views, map views, etc.

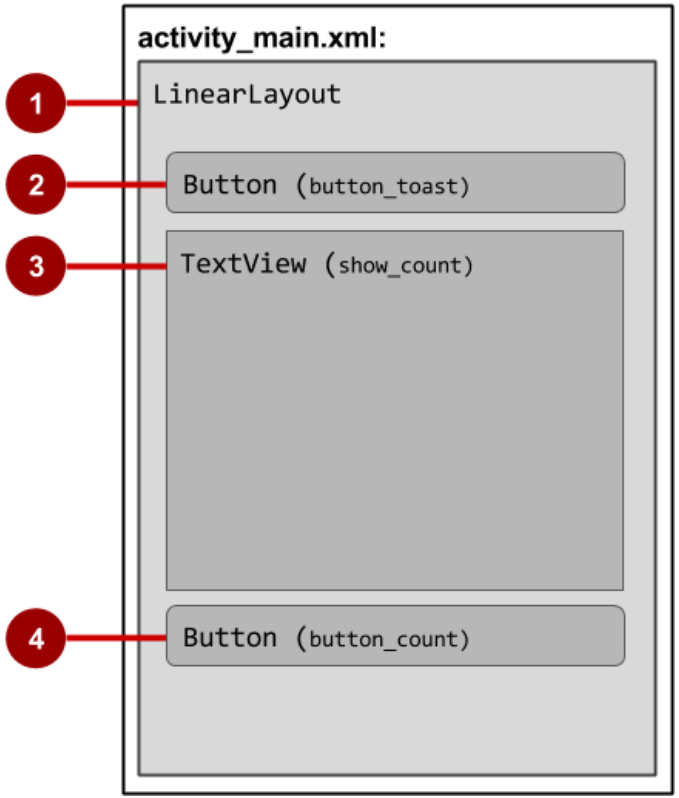


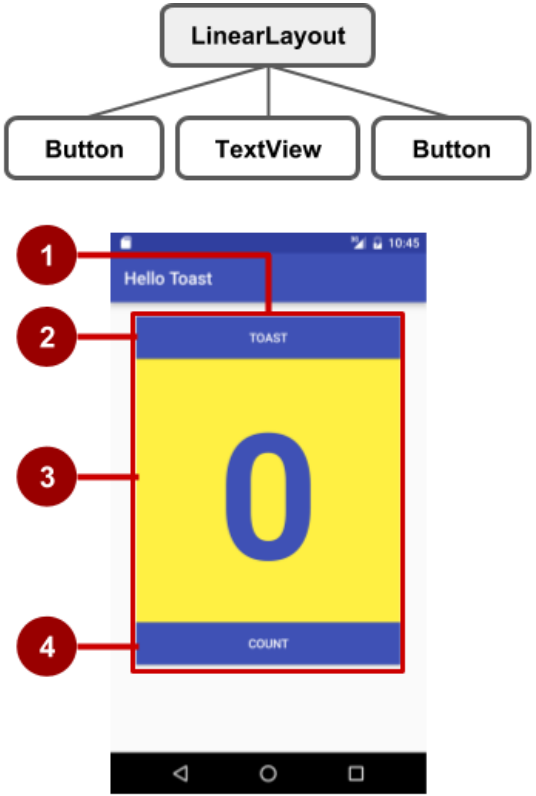
User Interface Layout



Example of LinearLayout

1. Contains the child views
2. First child at the top
3. Second child under the first child
4. Third child under the second child





Android Developer ABC

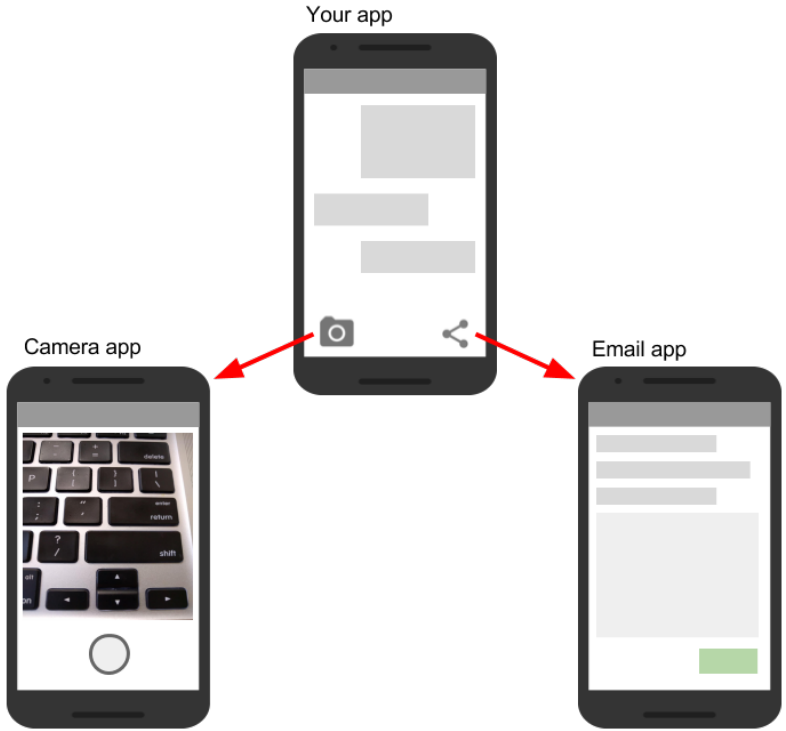
- App components
- Activity
- Services
- Content Provider
- Broadcast receivers

- App services
- Intents / Intent Filters
- App Widgets
- Processes and Threads

Activities

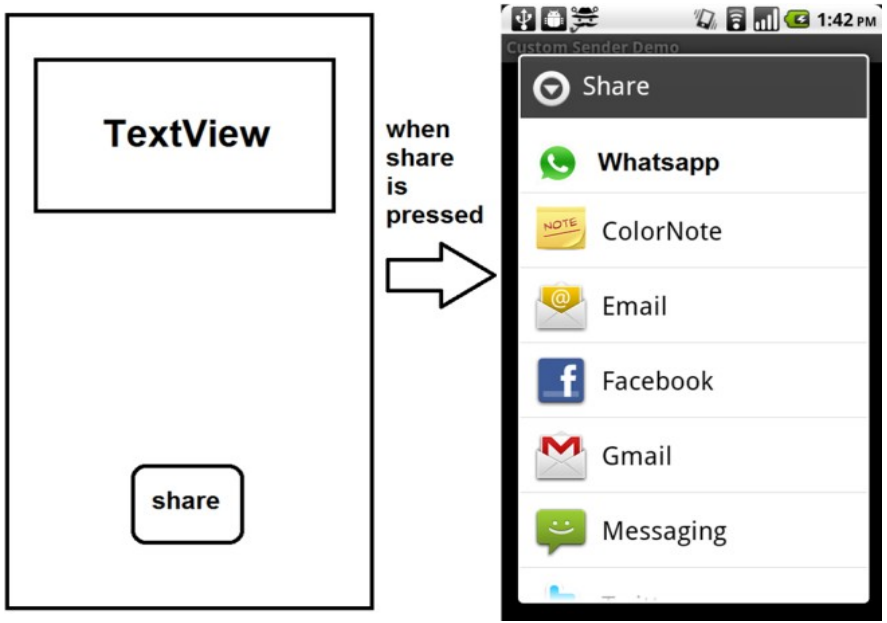
- An Activity:
  - application component that provides a screen
  - which users can interact with
- Typically correspond to one UI screen
- But can be:
  - Faceless
  - In a floating window
  - Return a value
- `MainActivity` is the starting point for an app

Activities Started in other Apps



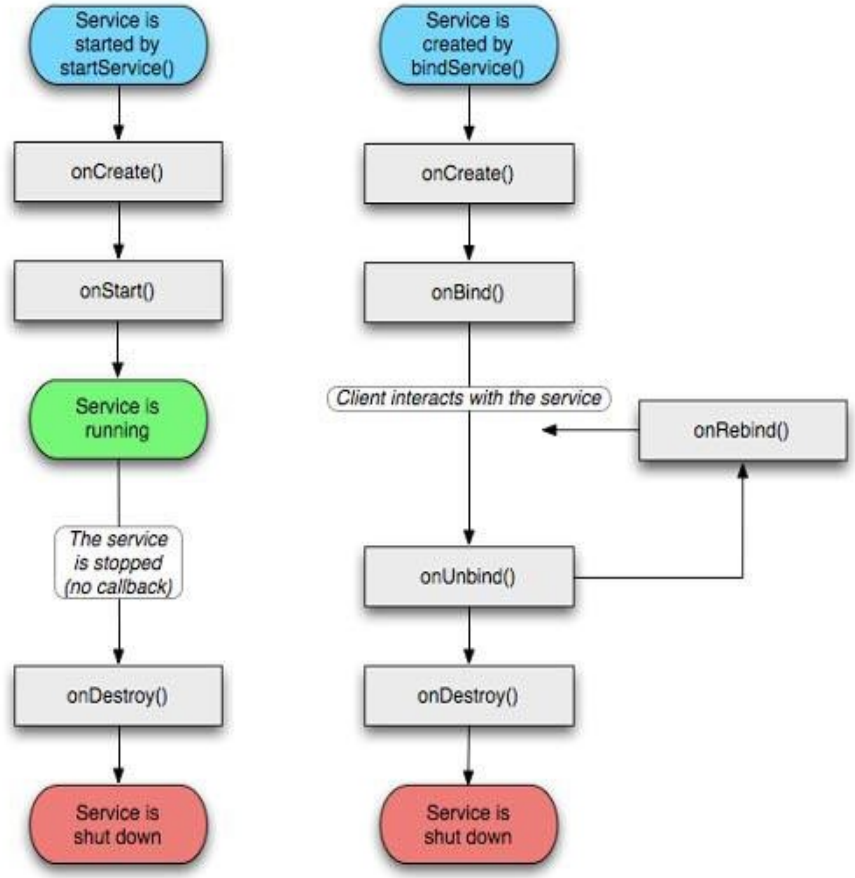
Intents

- Intents is a message object that starts activities
- Intents can also pass data between activities
- Explicit and implicit intents



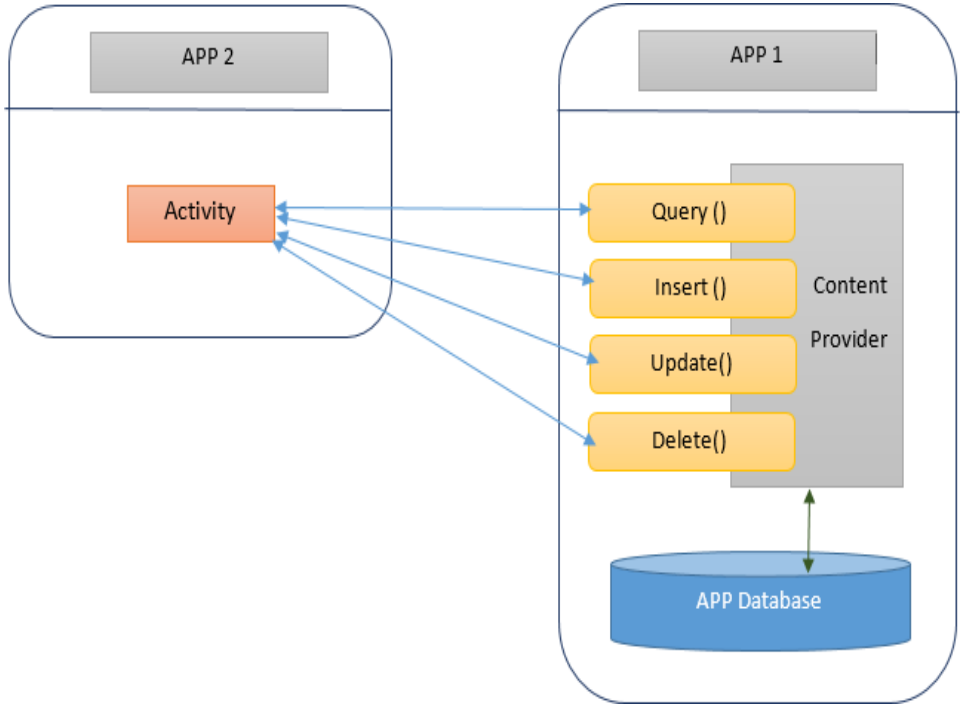
Services

- Long-running operations that mostly
  run in the background
  - Music player
  - Network downloads
  - …



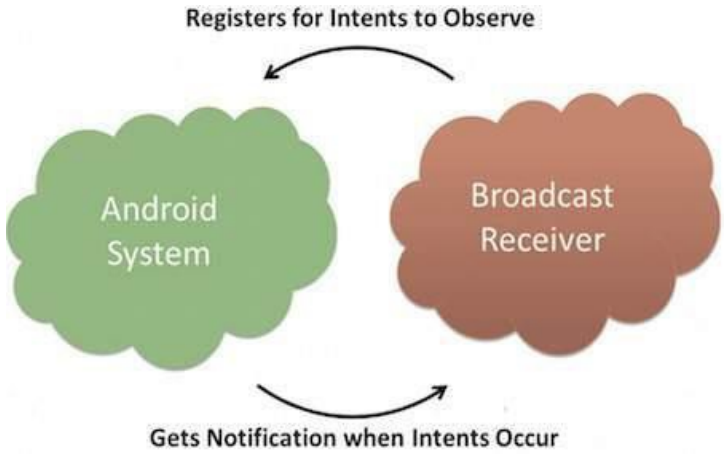
Content Providers

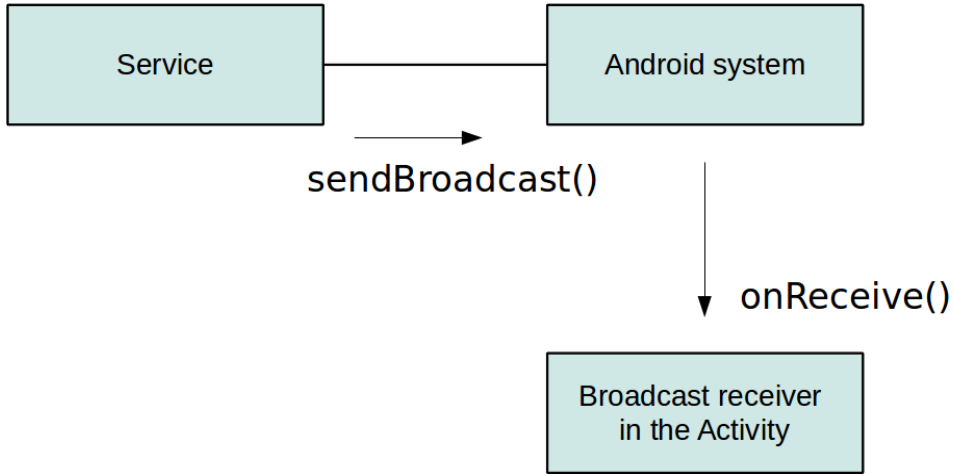
- Enables sharing of data across different apps:
  - Address book
  - Photo gallery
- Provides a uniform API for
  - querying
  - delete, update and insert
- Content is represented by
  - URI
  - MIME



Broadcast Receivers

- A broadcast receiver
  - component that responds to system-wide broadcast announcements
  - Sent when event occur that can affect other apps
    - > *Power connected, headphones disconnected, …*





App Widgets

- App Widgets
  - miniature application views
  - can be embedded in other applications (such as Home screen)
  - can receive periodic updates.

Application Files

- Manifest file – `AndroidManifest.xml`M
  - In the `app/manifests` folder
  - Defines structure and metadata for the application, its components, and requirements
- Gradle files define build configurations
  - Settings.gradle
  - Build.gradle (Project scoped and Module scoped)
- Java files for each activity
- XML files for each activity defining the UI
- XML files for values

Activities

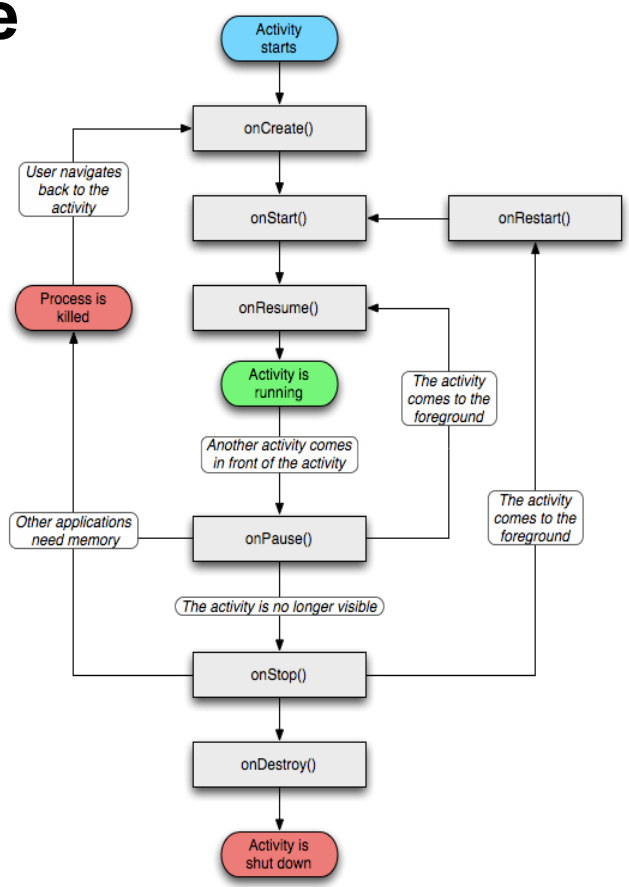
- An activity is
  - Application component
  - Represent one window or a hierarchy of views
- Activities handles
  - User interaction
  - Starting other activities
- Activities has a life-cycle
  - Created
  - Starting – Runnning - Pause - Resume
  - Stop
  - Destroyed
- Activities usually has a UI layout defined in an XML-file

Implementing Activities

- Define layout in XML
- Define Activity java class
  - Extends `APPCompactActivity`
- Connect Activity with Layout
  - Set content view in `onCreate()`
- Declare Activity in the Android manifest

Application Process / Lifecycle

- All components of one application:
  - Default: run in one process
- Processes started and stopped as needed
- Processes may be killed to reclaim resources



Intents

- An intent is an object used to request an action
  - from another app component via the android system
- Intents do
  - Start an activity:
    ```
    Intent intent = new Inten(this, ActivityName.class);
    startActivity(intent);
    ```
  - Start a service
  - Deliver a broadcast
  - Sending and receiving data
  - Either
    data (one piece of information - URI)
    extras (one or more piece of information – Bundle)

Advice for the Project

- Hands on course – learn by the keyboard
  - If using pair programming – switch roles often between driver and observer
- Use software version control system to collaborate
  - For example github
  - Divide the work
- Start small and work iteratively
- Use a coding convention
  - For example: https://source.android.com/setup/contribute/code-style
- Learn to use the debugger

Hello World Exercise

- Go to
  https://developer.android.com/training/basics/firstapp
  and follow the instructions

References

[1] Android Fundamentals Training, https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/, retrieved 2021-01-31

[2] Android Tutorial, https://www.tutlane.com/tutorial/android, retrieved 2021-01-31

[3] Tracy, Kim W. "Mobile application development experiences on Apple's iOS and Android OS." *IEEE Potentials* 31.4 (2012): 30-34.