

Mobile- and Web-based Software

Lecture 1: Introduction, Hello World

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

Requirements

- A. Mobile home assignment. (15%)
- B. Mobile midterm test. (35%)
- C. Web home assignment. (15%)
- D. Web midterm test. (35%)

No endterm exam!

Requirements

- Mobile midterm: 16 October. 18:00-20:00
- Web midterm: 20 November. 18:00-20:00
- Retake: 11 December. TBA
- Minimum requirement for grade:
 - > Each item (A,B,C,D) over 40%.
- Grades: 40-55-70-85% (2-3-4-5)

Requirements

- Assignments with deadlines
- Github registration necessary
- Android Studio need to be installed to do the assignments and for practicing.
- Android smartphone is not required.
- More info will be handed out with the assignments (Moodle).

Schedule

- Mobile-based software
 - > 1-7th week
- Mobile test (theoretical and practical)
 - > 7th week
- Web-based software
 - > 8-14th week
- Web test (theoretical and practical)
 - > 12th week

Contact

- Mobile part: David Sik, david.sik@aut.bme.hu
- Web part: Dr. Mohammad Saleem, mohammad.saleem@aut.bme.hu
- Lectures, practices:
 - > Wednesday 16:15-17:45 QB203
 - > Thursday 16:15-17:45 QB203
- No lessons:
 - > 23 October (Holiday)
 - > 21 November (TDK)
 - > TBD (bonus time for home assignments, always check in Moodle)
- Lectures with demos and coding together
- <https://github.com/siktdavid/MobWeb2024>
 - > Github repo: demos, sources (access info on Moodle)
- <https://edu.vik.bme.hu/course/view.php?id=12537>
 - > Moodle: slides, assignment information, results

Question time



Agenda

- Mobile Software Development
- iOS
- Android
- Hello World project 😊
- Android Studio

Mobile Software Development

Mobile Devices

- What is a *mobile device*?
 - > Earlier: device for radio network based voice communication
 - > Nowadays: miniaturized personal computer (smartphone, tablet, smartwatch...) with high processing power, loads of sensors and networking capabilities

Characteristics of Mobile Devices

➤ Main characteristics:

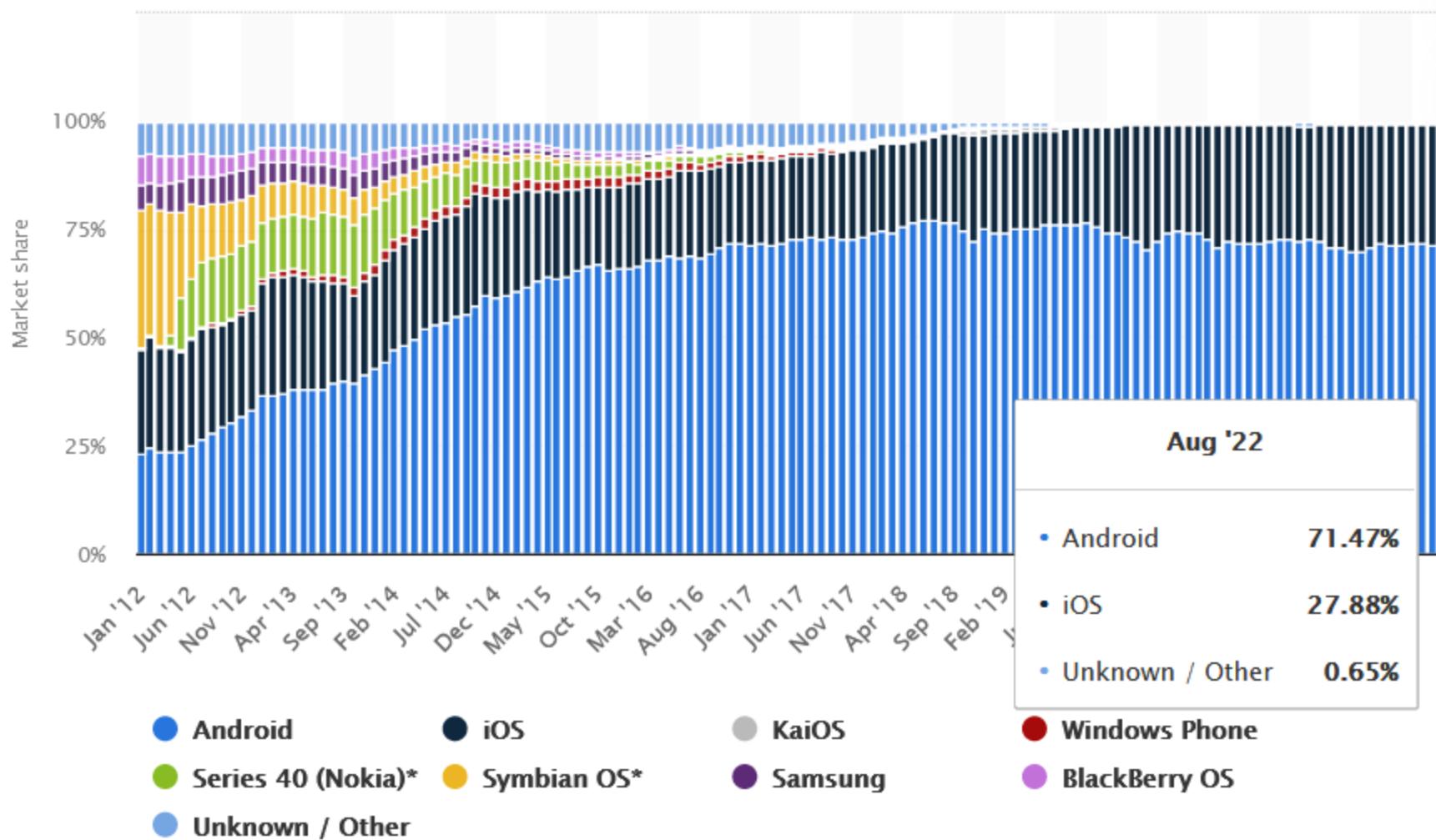
- Battery constrained
- Have a lot of different sensors and peripherals
- Wireless network communication
- Smaller form factor and various screen sizes
- Touch-based user interface
- Limited hardware capabilities (compared with a standard desktop/laptop computer)

➤ Mobile Software Development: creating software for mobile devices via different SDKs and programming languages

Platforms

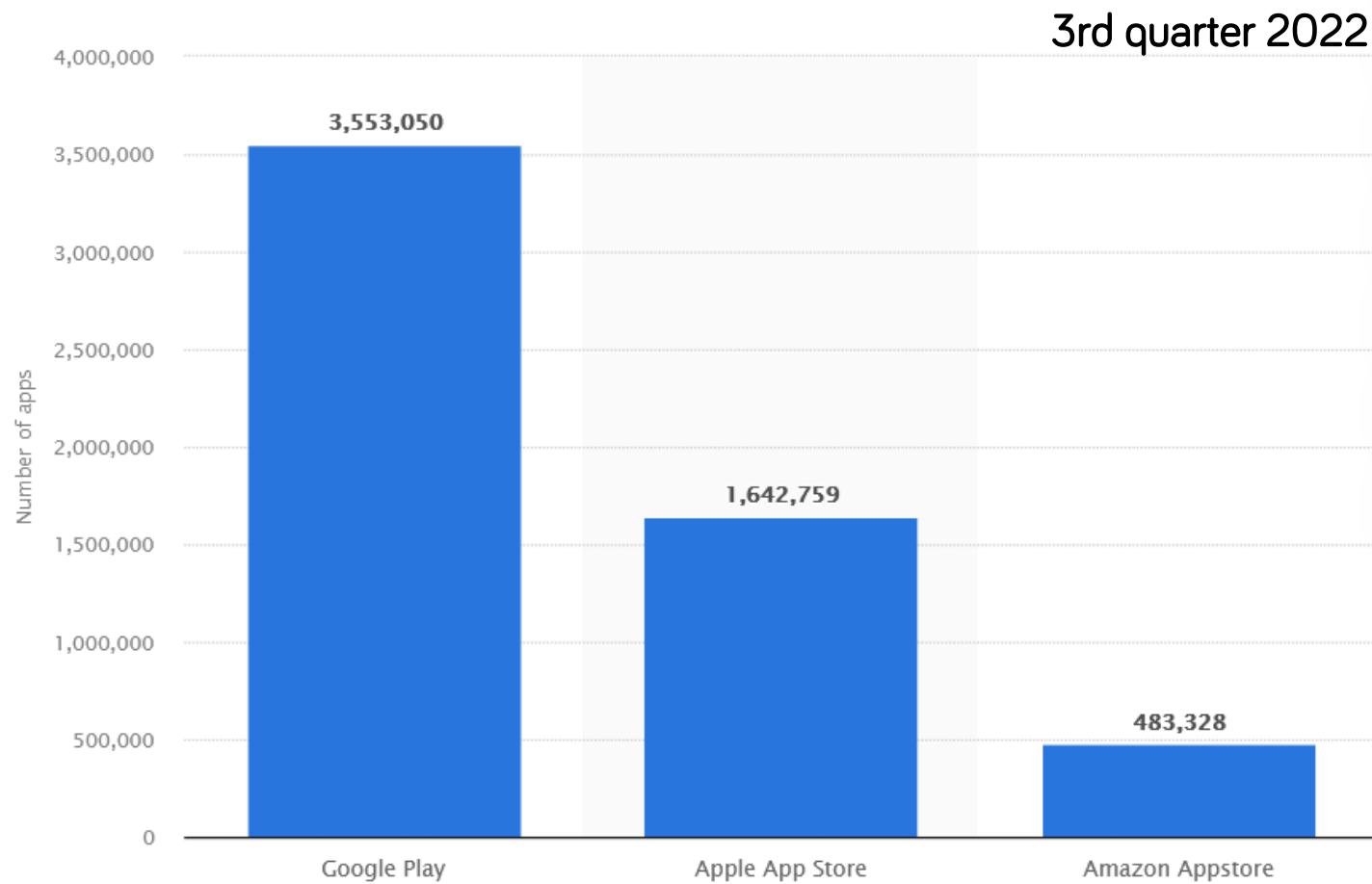
- Current major mobile platforms
 - > Android
 - > iOS
- Lots of minor platforms
 - > Like windows (phone)

Platform Market Share



<https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/>

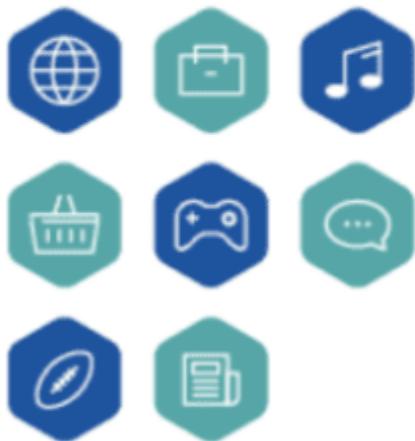
Apps in Stores



<https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>

HOW MANY APPS DO WE USE?

On average, a person now has more than **80 apps** installed on their phone



The average person uses...

9

mobile apps on a daily basis and

30

apps every month

<https://buildfire.com/app-statistics>

Usage time of applications

App Users Spend 77% of Their Time on Their Top 3 Apps

Percentage of individuals' app usage spent on each user's personal top 10 apps



<https://buildfire.com/app-statistics>

Mobile Software Development

iOS

What devices run iOS?

- iPhone
- iPod Touch
- iPad
- Apple Watch
- Apple TV



iOS as an Operating System

- Based on the same core as OS X
 - > Most of the kernel (XNU) code is shared (UNIX/BSD components)
 - > iOS and OS X applications are not binary compatible
- Closed system, restrictions apply to how and what applications can be created for iOS
 - > Application development is only possible on OS X

Why It Became a Hit

- Novel UI
 - > Optimized for touch screens
 - > Multitouch
 - > UI speed is top priority
- First really popular/usable mobile web browser: Safari
- App Store model: centralized application distribution model
 - > All applications are accessible at one place
 - > Only small initial investment is needed
- All iOS devices have a unified UI and similar capabilities

Shortcomings/problems of the Platform

- Somewhat restricted APIs, limited background execution and inter-app communication
- Development is only possible on OS X
- Wired connection only via iTunes (no USB mass storage mode)
- Applications can only be acquired from the Apple App Store
 - Apple decides what applications are allowed; lot of criticism
- To get full control, you need to jailbreak your device
- Storage is not extendable
- Connecting to external devices/accessories only with a license (except for Bluetooth Low Energy since iPhone 4S)
- Only premium devices (Android is available for all device segments)

Brief History of iOS Devices

- June 2007: release of iPhone 1.0
- March 2008: the iPhone SDK is released
 - > Developing 3rd party apps became possible
- July 2008: iPhone 3G and iOS 2.0 is released, App Store opens
- April 2010: iPad is released (the OS is renamed to iOS from „iPhone OS”)
- April 2015: Apple Watch is released
- Currently iPad, iPhone 15, Mini, Pro, Max etc. Watch 8
- iOS 18 on 16 September 2024



What You CANNOT do in your iOS app (in general)

- Keep the app running continuously (as a service)
- Access a shared storage on the disk to share data between different apps
- Control phone calls (besides initiating a call)
- Connect to external devices using IR, Bluetooth 3.x or USB/wire
 - > Only after applying to the “Made for iPhone” program
- Launch an app without any user interaction (on device startup or push notification)
- ...

What You CAN do in iOS apps

- Wake up your app when it enters a location (geofencing) or receives a push notification
 - To show the UI, user interaction is still needed
- Launch other apps from your app
- Create custom keyboards
- Create “widgets” (Home Screen Extensions)
- Connect to Bluetooth Low Energy devices
- Vibrate the phone
- Create fancy visual effects using physical simulation and 3D graphics either using low level (OpenGL) or high level (UIKit) APIs
- ...

iOS Native Applications

- **Native:** written in Swift or Objective-C, compiled to machine code
 - > iOS devices use ARM CPUs
- Great performance, no virtual machine or on-device compilation is required

Cross platform technologies

- Xamarin.iOS / MonoTouch
 - > C# / .NET
 - > Ahead-Of-Time (AOT) compilation: IL code is compiled on the development computer to iOS binaries (no JIT or interpreter)
 - > Binding to most Cocoa Touch APIs, e.g. MonoTouch.UIKit.UIButton
 - > .NET garbage collection, .NET class libraries (Web Services, System.Xml, System.Threading, etc.)
 - > Development still requires OS X



Cross platform technologies

- Ionic
 - > Hybrid mobile app development
 - > Built on Angular
 - > HTML/CSS/JavaScript
- React Native
 - > HTML/CSS/JavaScript
 - > Compiles to native application
- Flutter
 - > Dart language
 - > Supported by Google

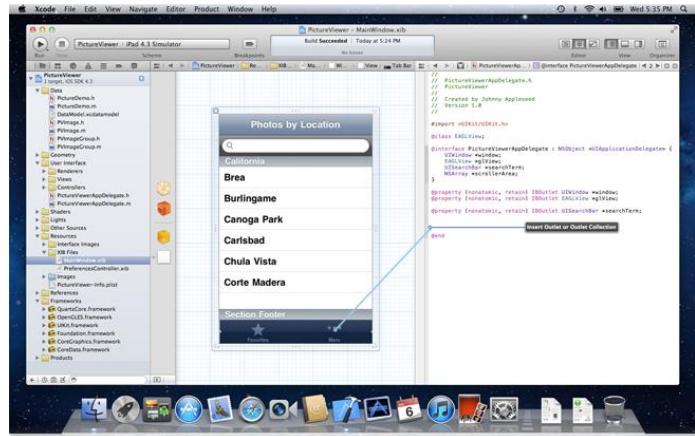
Development Tools

- Xcode (downloadable from OS X App Store), contains everything
 - > Xcode app: IDE, debugger, interface builder
 - > iOS Simulator
 - > Instruments: profiling tool, analyzes application during running (e.g. finding memory leaks)
- Xcode is free and updated with each new iOS version



Xcode the IDE

- Editor
- Interface builder
- Compiler: LLVM (Clang)
- Debugger: LLDB
- Used for both iOS and OS X development

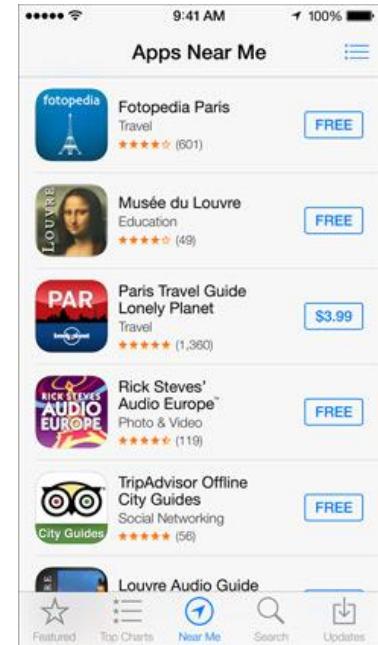


Distributing Apps

- Apple App Store
 - > Main distribution channel
- Ad-Hoc
 - > Maximum 100 devices, restricted by UDID
 - > Installer package (.IPA file) can be shared via email/websites/etc. Only devices with the given UDIDs can install the app
- Enterprise
 - > In-house distribution for companies; similar to ad-hoc distribution but no limit on device number

App Store

- Distribution channel for apps, accessed via the device/iTunes or web
 - > Developer can pick the price
 - > 70% of all sales revenue goes to developer
 - > Apps must comply with the App Store rules
 - All apps are evaluated by Apple before they are released to the public (~ 5 days)
- Developer must sign up for the Apple Developer Program
 - > \$99 / year



Apple Services

- Push Notification
- In-App Purchase (StoreKit)
 - > 70% goes to developer
- iCloud
 - > Cloud storage, all Apple IDs gets 5GB for free
 - > API: file, settings, database (Core Data) storage
- iAd
 - > In-app ads (similar to Google AdMob)
 - > 70% goes to developer
 - > Per click and per impression
- Game Center
 - > Matchmaking, leaderboards, achievements

Mobile Software Development

Android

Origin of Android

- Android is one of the most popular mobile operating systems of present days
 - > It become a brand by intent (TV spots, ads, etc.)
 - > Google stands behind the platform
- Android offers the picture of a well functioning OS
- Revolutionized the picture of mobile operating systems



Android devices

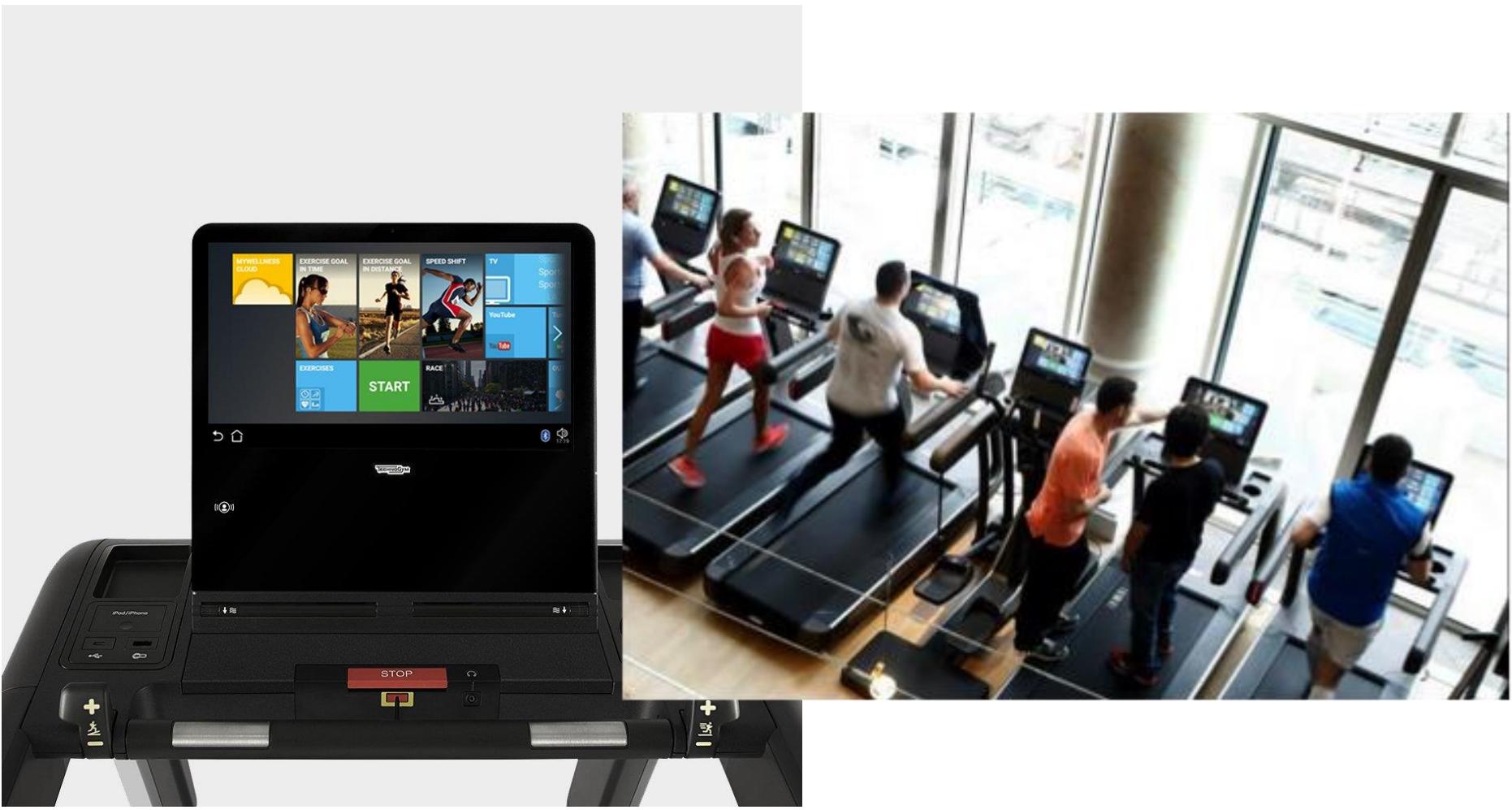
- Mobile phone and tablet manufacturers
- Car navigation and multimedia systems
- Wear OS
- Industrial solutions
- Android has potential on many other fields:
 - > Small and large screen
 - > Limited resource capabilities (e.g. battery)
 - > Touch screen support
 - > Android@Home



Android in cars



Android in fitness rooms



Where does Android came from?

- An operating system not only for mobile phones!
- The collection of all devices that support Android
- It is in the hands of Goolge
- **2005**: Google buys a company, called: *Android Incorporated*
- **2007**: The first news have been leaked that Google will enter into the mobile market
- **November 5, 2007**: The newly formed Open Handset Alliance has introduced the Android platform
- **End of 2008**: The HTC G1 (Dev Phone) was available on the market by T-Mobile

Open Handset Alliance

- Android is not only Google's development
 - > More than 50 member companies
 - > 11 manufacturer (HTC, LG, Motorola, Samsung, Asus, Garmin, Huawei, Sony Ericsson, Ericsson, Toshiba, Acer)
 - > 13 semiconductor manufacturer (not at least Intel, Marvell, nVidia, Broadcom, SiRF, ARM and Atheros)
 - > 10 operator (example T-Mobile and Vodafone)
 - > 12 software company (Google, eBay)
 - > 7 other corporation
- Members are changing frequently
- Android:
 - > Product of huge knowledge and lot of work

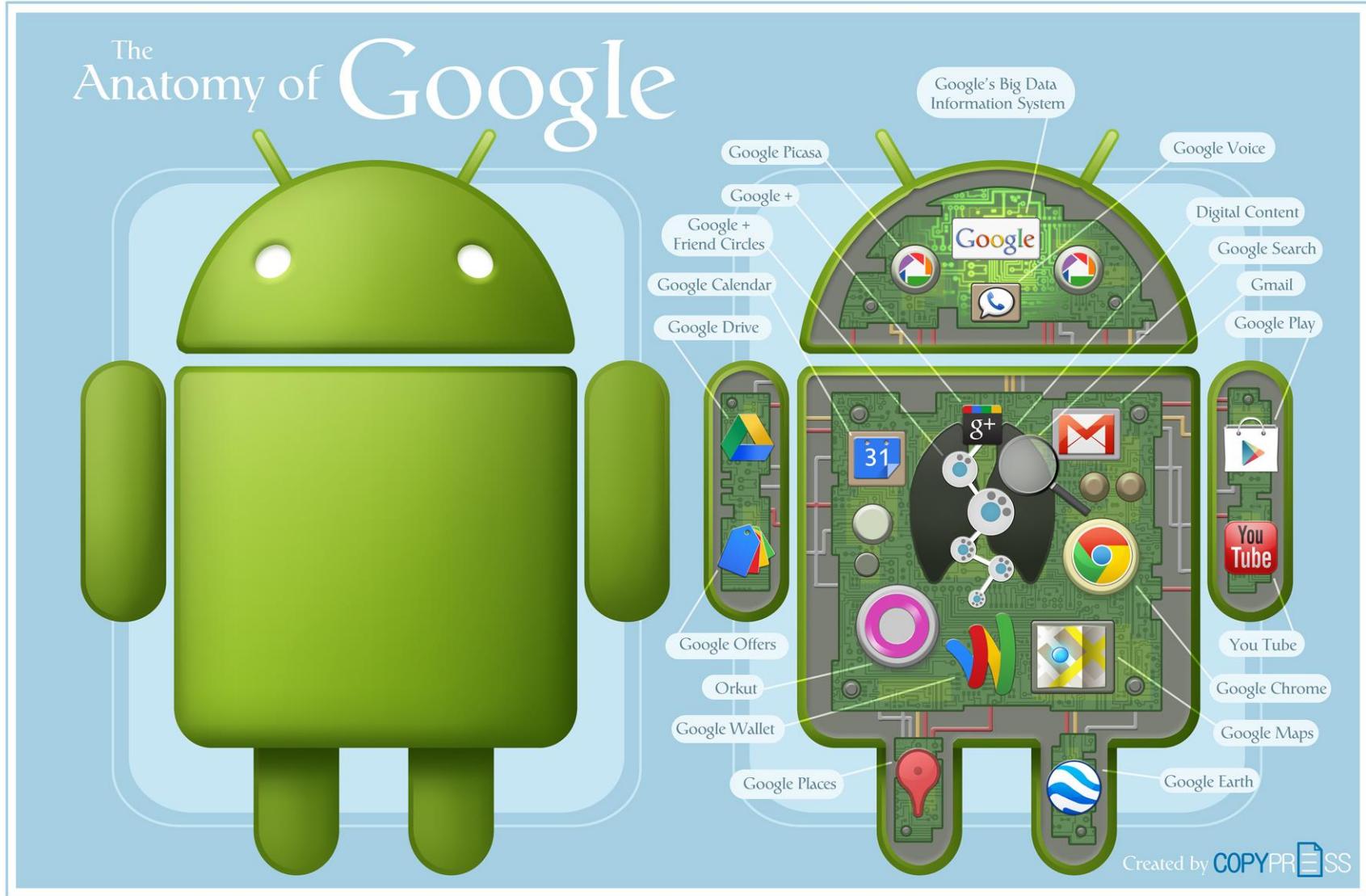


open handset alliance

Why Android become successful?

- User experience (iPhone like...)
- Several devices (high end and low end)
- Small hardware requirements
- Open source, free to use (with some limitations)
- Fast application development
- Marketing

Android and Google integration



Android and Google integration

- After assigning a Google/GMail user, the OS will keep the contact list, the messages, the conversations, and the calendar synchronized with it
- When the user starts a conversation with a new partner in **Gmail**, then the new name will be shown in Android contact list
- When the user adds a new contact after a phone call, it will be synchronized with **Gmail**. This works for calendar and **GTalk** too.
- Users can share their photos created with the camera of the devices at **Picasa**
- Using Latitude, users can share their position with friends
- **Google+** community network functions

Android: from the perspective of larger manufacturers

- Their sales were decreasing
- It was necessary to support Android for them
- They did not have resource to develop their own system (expect Samsung Bada)
- Android is free



Android: from the perspective of new manufacturers

- Easy to reach market: free and popular operating system
- Excellent value for money: open doors to the market
- New mobile brands (e.g.: Huawei, ZTE, Xaomi, Meizu)
- Open doors to reach operator markets

And the others?

- **Apple**: own successful solution (iOS)
 - > Based on their success they introduces new features on their devices, that were previously on Web (e.g.: map)
- **Microsoft**: can not support Android directly, but it is good that there is something against iOS while Microsoft develops Windows Phone further
- **Nokia**: too long lifetime for Symbian
 - > Refused Android
 - > Partnership with Microsoft
 - > ... after Microsoft?



Android versions



ANDROID

- Android 1.0 – 2008. October
- Android 1.1 – 2009. February
- Android 1.5 (Cupcake) – 2009. April
- Android 1.6 (Donut) – 2009. September
- Android 2.0 and 2.1 (Eclair) – 2009. October
- Android 2.2 (Froyo) – 2010. May
- Android 2.3 (Gingerbread) – 2010. December
- Android 3.0-3.2 (Honeycomb) – 2011 January-July
- Android 4.0 (Ice Cream Sandwich) – 2011. October
- Android 4.1 (Jelly Bean) – 2012. July
- Android 4.2 (Jelly Bean) – 2012. November
- Android 4.3 (Jelly Bean)
- Android 4.4 (KitKat)
- Android 5.0, 5.1 (Lollipop)
- Android 6.0 (Marshmallow)
- Android 7.0, 7.1 (Nougat)
- Android 8.0, 8.1 (Oreo)
- Android 9.0 (Pie)
- Android 10 (Q)



Android versions

- Developers must follow version changes
- Could be significant differences between versions
- Google pursue backward compatibility, but there are deep gaps (v 3.0)
- Before developing, it's necessary to choose the minimum supported version well
- Version codename: some candy ☺
 - But this was stopped when Android 10 arrived ☹

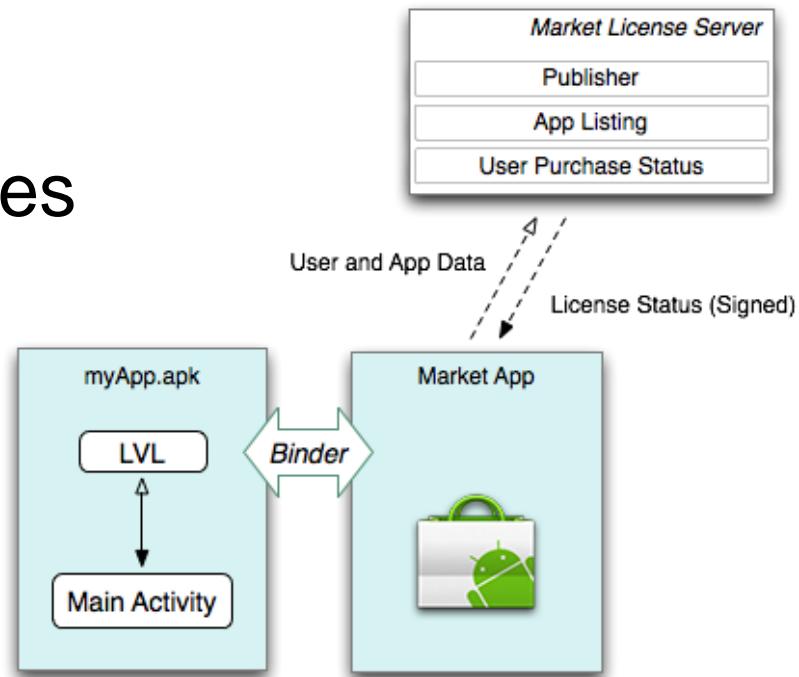
Google Play (earlier: Android Market)

- <https://play.google.com/store>
- Available since 22. October, 2008.
- Application store for Android devices
- The Play Store application is not open source!
- Pre-installed on all major Android device
- One of the biggest application store nowadays



Google Play Services

- Android Licensing Service
- In-app Billing
- Google authentication
- Google+ integration
- Google Play Gaming Services
- Google Cloud Messaging
- Fused Location
- ...



The capabilities of the Android platform 1/2

- Rich application behavior capabilities (background services, lazy connected components, easy to reach system services)
- Full JavaSE class library + reach Java API for Android functions like network handling, telco, etc.
- Rich UI widget library, easy to develop custom widgets, styles and themes
- Dynamic UI behavior, animations, rich resource capabilities

The capabilities of the Android platform 2/2

- Secure file management, SQLite and easy-to-use key-value storage
- Linux based; native development is supported
- Rich external library collection (Java and native):
 - Always check first! – GitHub, Google Code, etc.
- Rich Google Services for publishing, advertising and for development
- ... **there are always new APIs, tools and libraries!**

Application behavior

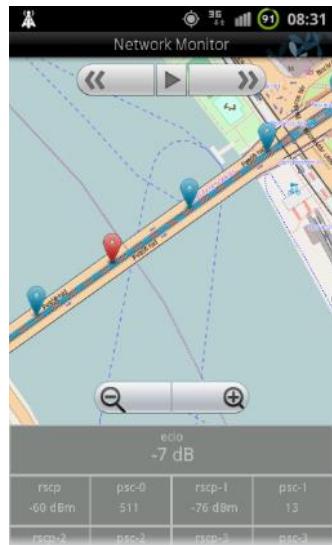
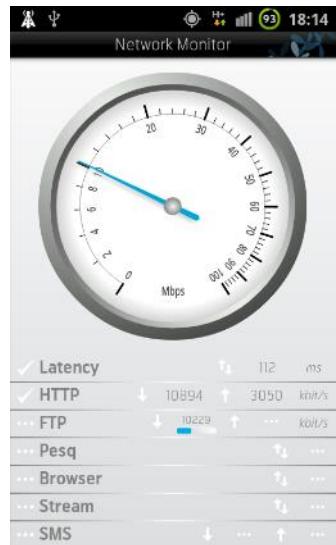
- When the application starts and activates
 - > Manual start (clicking on an icon)
 - > Activation by some external event (incoming call, battery status, boot ready)
 - > Services and contents provided by the application
- Application running type
 - > Foreground running
 - > Background services
- Special needs
 - > Widget support
 - > Live wallpaper

***How do we handle
these on Android?***

**Android application
components**

Case Studies

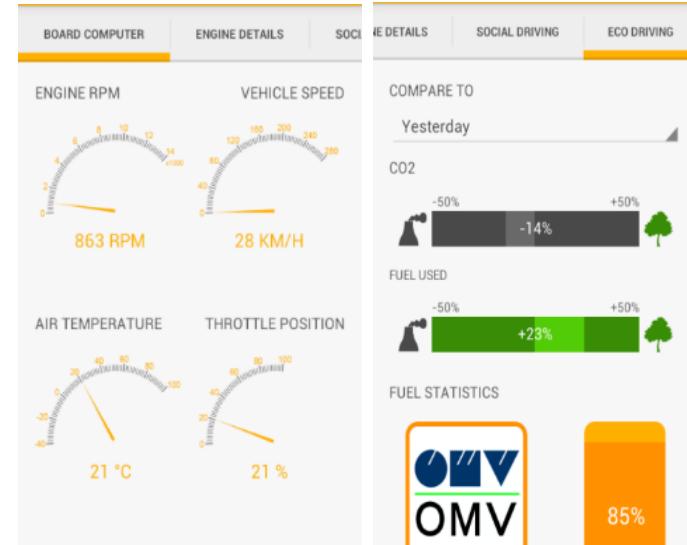
NetworkMonitor



DrTorrent



VehicleICT



Summary

- Why iOS became popular
- Why Android became popular
- iOS and Android platform versions
- App store vs. Play
- Simulator vs. Emulator
- Xcode vs. Andorid Studio
- Case studies



Homework - Reading

- About Android:
 - > <https://developer.android.com/about/android.html>
- Android Studio Install Guide:
 - > <https://developer.android.com/studio/install.html>



Some Learning Materials

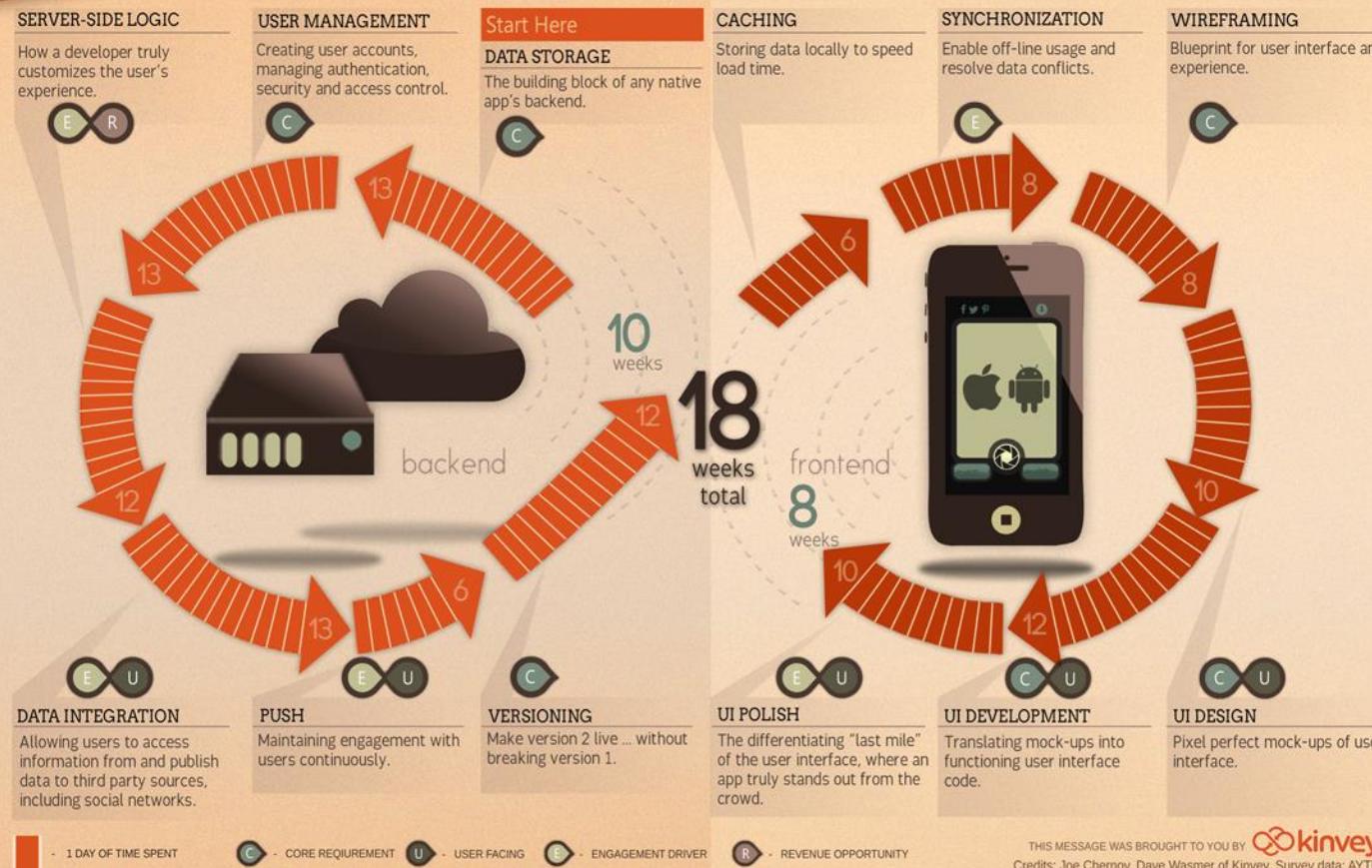
- Android programming
 - > Android Development with Kotlin
 - <https://www.packtpub.com/application-development/android-development-kotlin>
- Android developer portal
 - > <http://developer.android.com/develop/index.html>

Android platform structure

How long does it take to build a mobile app?

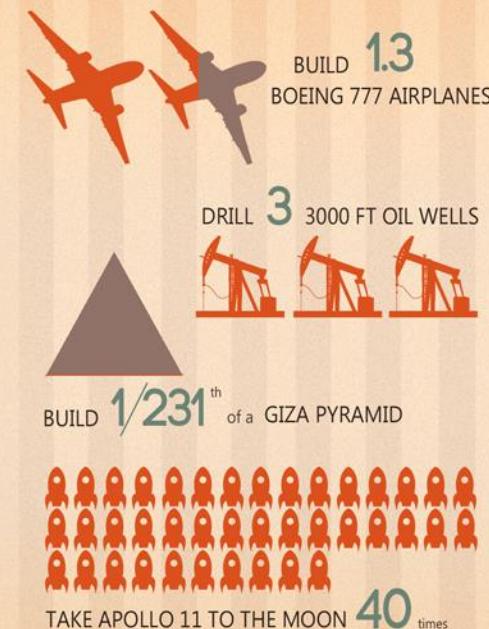
How Long Does it Take to Build an iOS or Android App?

Based on a survey of 100 iOS, Android and HTML5 developers, it takes nearly 18 weeks to build v1 of a native app.



It Also Takes 18 Weeks to:

In the 18 weeks it takes to develop and publish a native app, one could:



Primary sources:

AYTM survey of 100 iOS, Android and HTML 5 developers

Other sources:

<http://www.universetoday.com/13562/how-long-does-it-take-to-get-to-the-moon/>

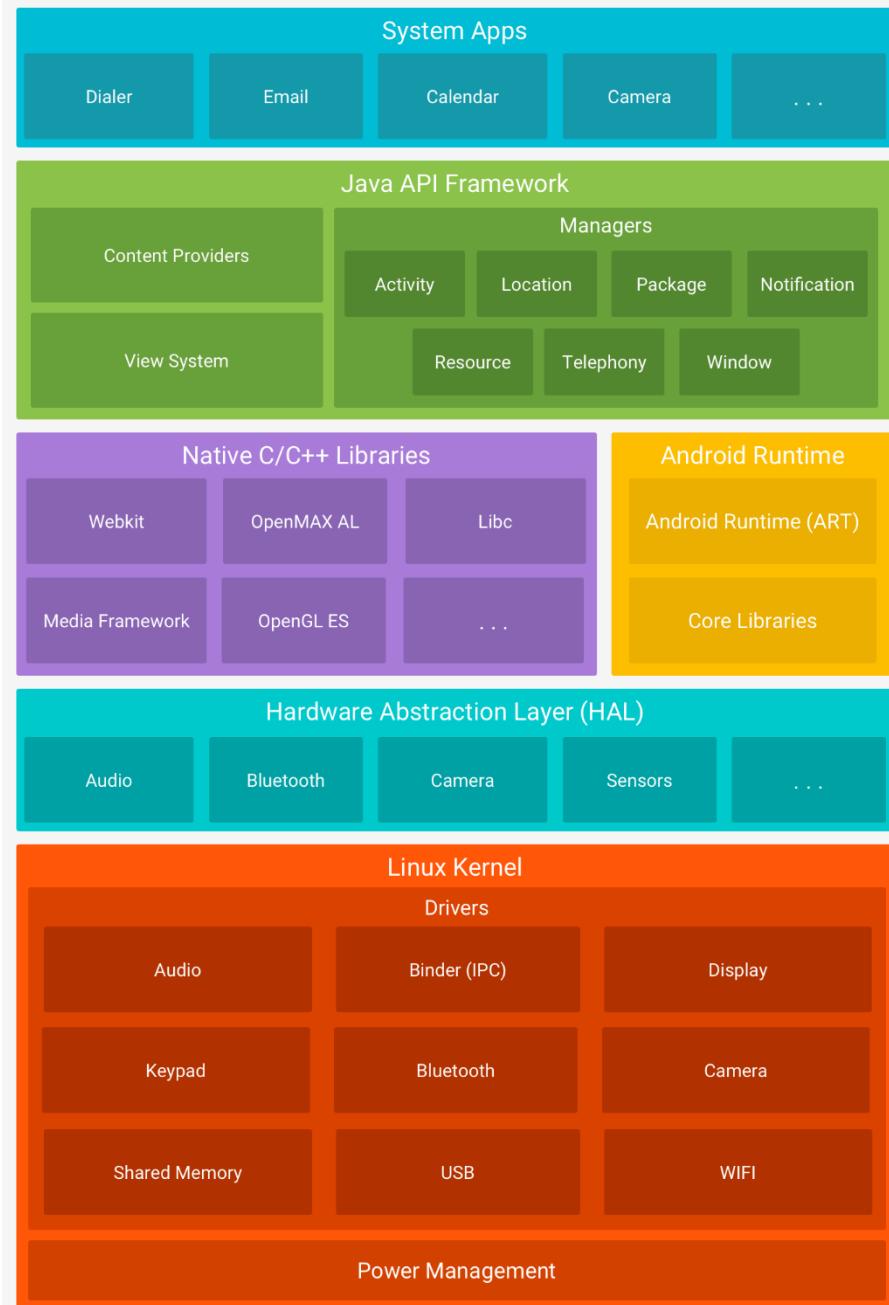
<http://www.airlinereporter.com/2012/01/how-long-does-it-take-to-build-a-boeing-777/>

<http://www.cedarvalleyenergy.com/FAQs.htm>

<http://www.pbs.org/wgbh/nova/ancient/who-built-the-pyramids.html>

Dedicated by

Structure of Android platform



Managed code

- Managed code runs on a virtual machine (Dalvik)
- It makes the application run secure
- An application can't ruin the whole system
- Memory management is happening with a garbage collector
- But! Be careful to create „good” source code

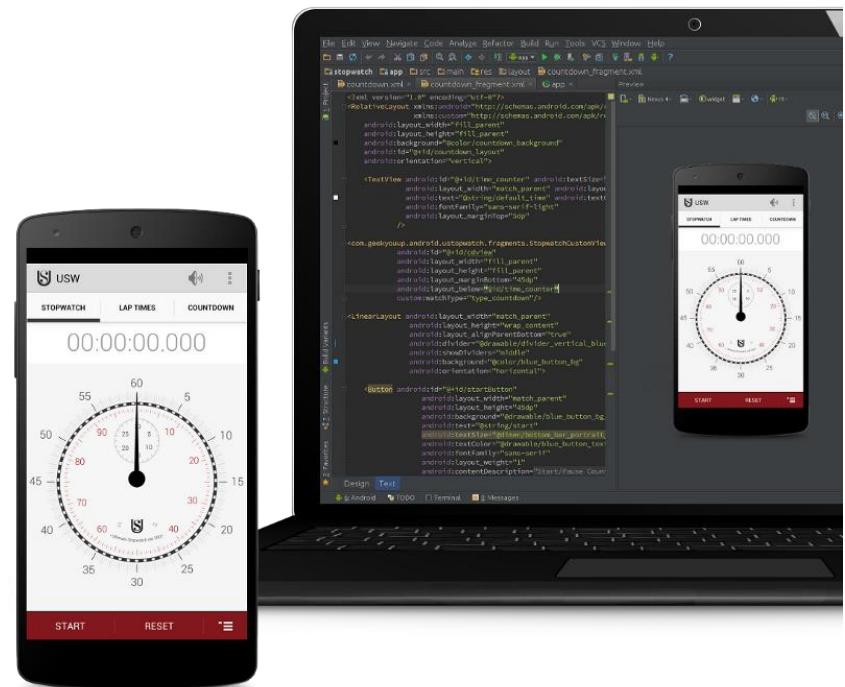
Development tools

Development Tools on Android

- **Android SDK (Software Development Kit):**
 - > Development tools, documentation, all Android version
 - > Emulator management (AVD Manager)
 - > Always up to date
 - > Java and Kotlin
 - > .apk generation
- **Android NDK (Native Development Kit):**
 - > Enables to compile native code
 - > C++
- **Android ADK (Accessory Development Kit):**
 - > Support for designing accessories like: docking station, health devices, etc.)
 - > Android Open Accessory protocol (USB and Bluetooth)

Android Studio

- Flexible Gradle-based build system.
- Build variants and multiple APK generation.
- Expanded template support for Google Services and various device types.
- Rich layout editor with support for theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform
- Version control integration



Let's see some examples

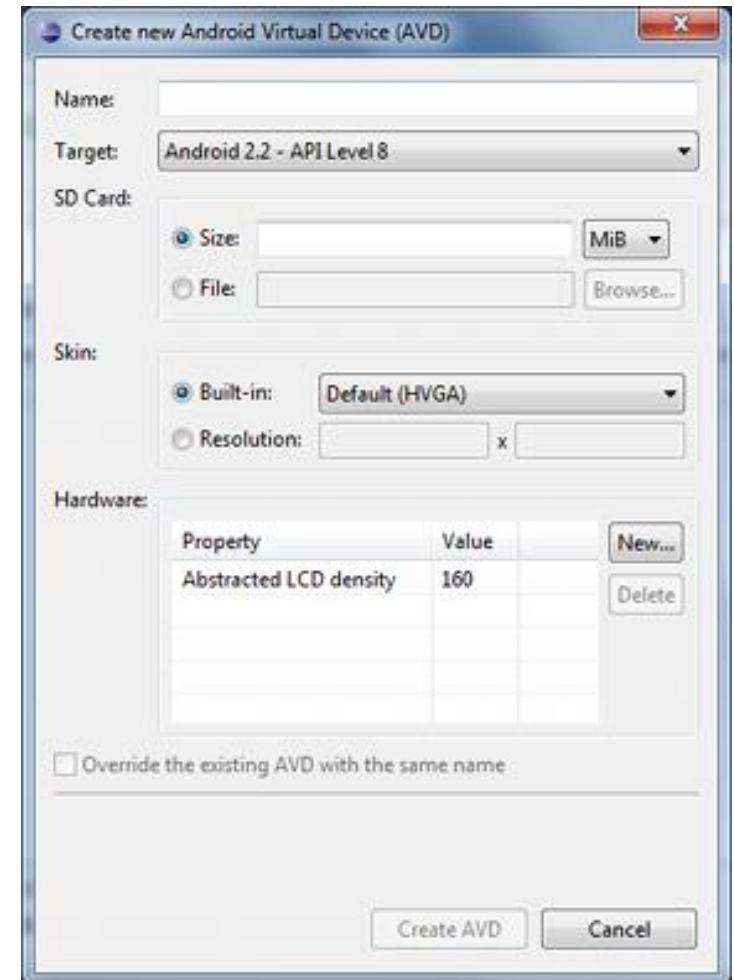
- Let's exam the structure of the Android SDK!
- Lets create and start an emulator (AVD – Android Virtual Device)!
- Try to connect a phone to Android Studio

SDK Manager

- Download and update SDK components
- Check installed versions
- USB Drivers
- Hardware acceleration
- Google Play APIs
- Support libraries

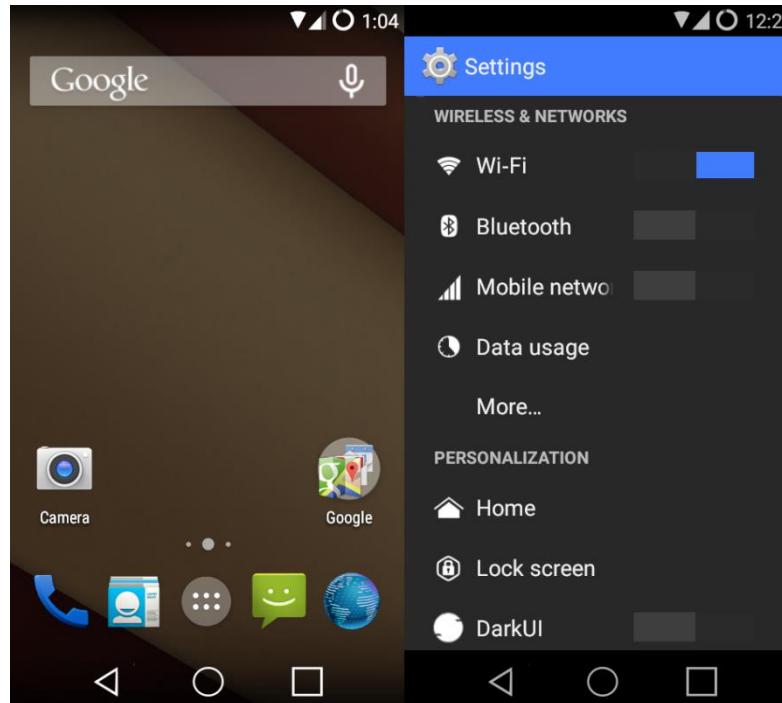
AVD Manager - Emulator configuration

- For launching the emulator an AVD (Android Virtual Device) must be configured
- AVDs can be created via the AVD Manager application that is part of the SDK
- AVD properties:
 - > Name
 - > Android platform version
 - > SD card size
 - > Skin
 - > Screen properties



Emulator

- Emulates the whole Android OS
- Hardware acceleration can be configured
- Built-in applications are available



Which programming language is not supported for Android development?

- A. Java
- B. C++
- C. Swift
- D. Kotlin

Practical info

- <https://developer.android.com/studio>
- At least 20 GB (15...) free space, 16 GB (8...) RAM
- Install Virtual Device too
- Standard install
- Android API 35, SDK, Build Tools, SDK Platform-Tools
- Intel x86 / AMD Emulator Accelerator HAXM (CPU dependent)
 - > If available, Virtualization, BIOS settings, Hyper-V disable
- Medium Phone API 35 emulator with quick boot
 - > Android API 35 x86_64, Android API 35 Google Play

First Android Application

Summary

- Android platform structure
- Development tools: SDK, Emulator, Android Studio
- First Android Application

Homework - Readings

- Platform Architecture:
 - > <https://developer.android.com/guide/platform/index.html>
- Meet Android Studio:
 - > <https://developer.android.com/studio/intro/index.html>
- Developer Workflow Basics:
 - > <https://developer.android.com/studio/workflow.html>
- Create a project:
 - > <https://developer.android.com/studio/projects/create-project.html>
- Run apps on the Emulator:
 - > <https://developer.android.com/studio/run/emulator.html>
- Run apps on Device:
 - > <https://developer.android.com/studio/run/device.html>



What have we learnt so far?

- Android platform structure
- Android development environment
- SDK structure
- Hello world project in Android Studio

Summary

- iOS
- Android ☺
- Android platform structure
- Development tools:
 - > SDK, Emulator, Android Studio
- First Android Application

Thank you!



Mobile- and Web-based Software

Lecture 2: Introduction, Kotlin

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

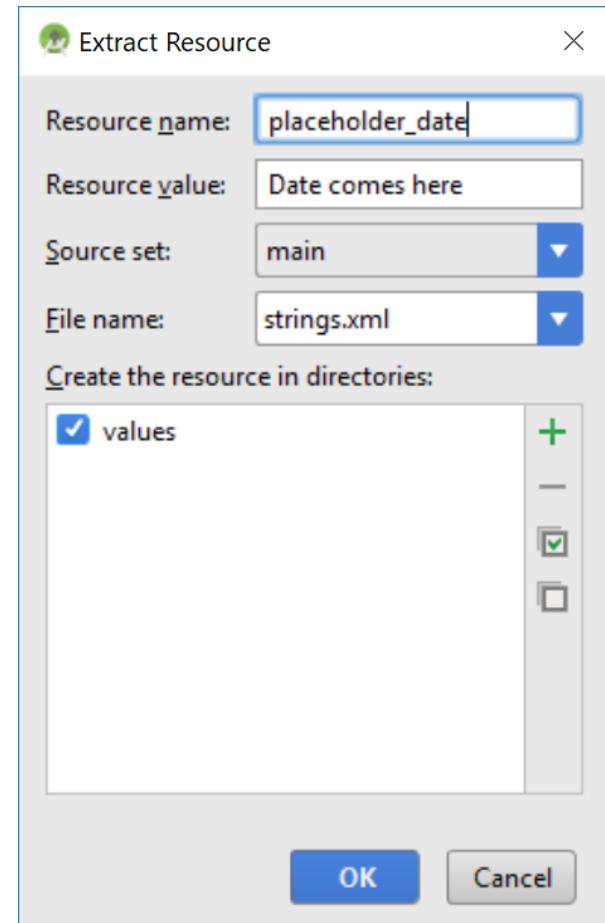
Agenda

- First Android Application
- R.Java
- String resources (extract string)
- LogCat, Debuging

First Android Application

Extract string resources

- OPT+Return / Alt+Enter
 - > Extract String Resource



Where is the R.java? 😊

Android Gradle Plugin (AGP) updates

Android Gradle plugin 3.6 and higher includes support for the Maven Publish Gradle plugin which allows you to publish build artifacts to an Apache Maven repository. The Android Gradle plugin creates a component for each build variant artifact in your app or library module that you can use to customize a publication to a Maven repository. This change will make it easier to manage the release lifecycle for your various targets. [Learn more](#)

Additionally, Android Gradle plugin has made significant performance improvement for annotation processing/KAPT for large projects. **This is caused by AGP now generating R class bytecode directly, instead of .java files.**

Android LogCat

- System debug output
- Application and system log can be monitored
- Built-in Log class with different log level support:
 - > v(String, String) (verbose)
 - > d(String, String) (debug)
 - > i(String, String) (information)
 - > w(String, String) (warning)
 - > e(String, String) (error)
- Log.i("MyActivity", "Debug message")
- Can be redirected into a file
 - > logcat -f <filename>
 - > <http://developer.android.com/tools/help/logcat.html>

Debugging

- Run in debug mode
- Breakpoints
- Step over (F8)
- Resume

What to do in case of crash?

- Look for exception message in Logcat
- Use Log.d(...) to print out messages
- Run app in debug mode (breakpoints)

What do you remember from the readings?

- Platform Archtecture:
 - > <https://developer.android.com/guide/platform/index.html>
- Running Your App:
 - > <https://developer.android.com/training/basics/firstapp/running-app.html>
- Building a Simple User Interface:
 - > <https://developer.android.com/training/basics/firstapp/building-ui.html>
- Write and View Logs with Logcat:
 - > <https://developer.android.com/studio/debug/am-logcat.html>

Android project elements

Android project elements

- Package: unique identifier
- Java/Kotlin source
- Resources
 - > Layout (XML)
 - > Themes and styles (XML)
 - > Text elements (XML)
 - > Animations and other dynamic graphics (XML)
 - > Images, sound, video, etc.
- R.java (generated automatically)
- AndroidManifest.xml
- build.gradle
- External libraries (.jar or other projects)

Manifest file

- Application descriptor
- Meta information
- Pre-defined component list
 - > Can be modified dynamically from code
- XML format
- The system checks the descriptors and components in the Manifest
- Other meta information like minimal Android version, permissions, etc.
- It is checked by the system before installation



Manifest content

- Java package – **unique identifier**
- Permissions (e.g.: Internet access, Location information, Phone calls, etc.)
- Minimum API level
- Hardware and software requirements (e.g. necessary camera, etc.)
- External APIs (e.g.: Google Maps API)

Manifest example 1/2

```
<?xml version="1.0" encoding="utf-8"?>  
  
<manifest xmlns:android=  
          "http://schemas.android.com/apk/res/android"  
          package="hu.bute.daai.amorg.examples"  
          android:versionCode="1"  
          android:versionName="1.0" >  
    <uses-sdk android:minSdkVersion="7" />  
  
    <application  
        android:icon="@drawable/ic_launcher"  
        android:label="@string/app_name" >  
        <activity ...>...</activity>  
    </application>  
  
</manifest>
```

Unique package name (identifier)

Minimum supported version

Application icon and label

Manifest example 2/2

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest .../>  
...  
<application ...>  
    <activity  
        android:name=".MainActivity"  
        android:label="@string/app_name">  
        <intent-filter>  
            <action android:name=  
                    "android.intent.action.MAIN"/>  
            <category android:name=  
                    "android.intent.category.LAUNCHER"/>  
        </intent-filter>  
    </activity>  
</application>  
</manifest>
```

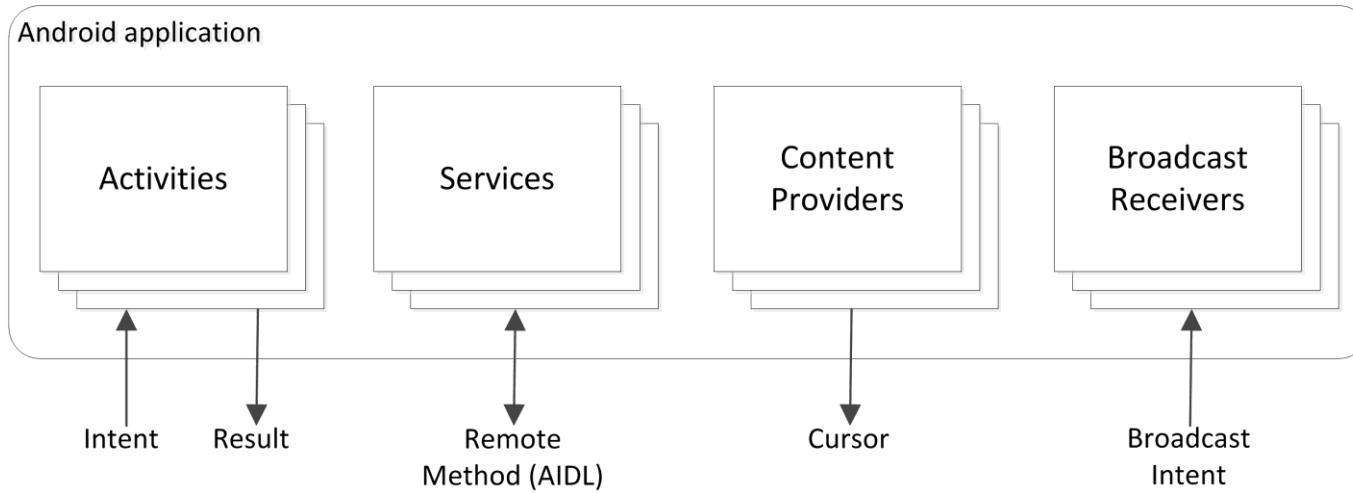
Activity class and title

Application entry point

Appears in the applications list

Android Application Components

- An Android application stands from one or more components:
 - > Activity
 - > Service
 - > Content Provider
 - > Broadcast Receiver



Manifest attributes and tags

- android:icon: application icon
- android:name: Application name with package name
- android:label: Application title
- List of components:
 - > <activity>: Activity
 - > <service>: Service
 - > <provider>: Content provider
 - > <receiver>: Broadcast receiver
- The *Activities*, *Services* and *ContentProviders* that are not present in the Manifest, are not visible for the system
- However *BroadcastReceivers* can be defined dynamically from code:
`registerReceiver()`

What is true about Manifest file?

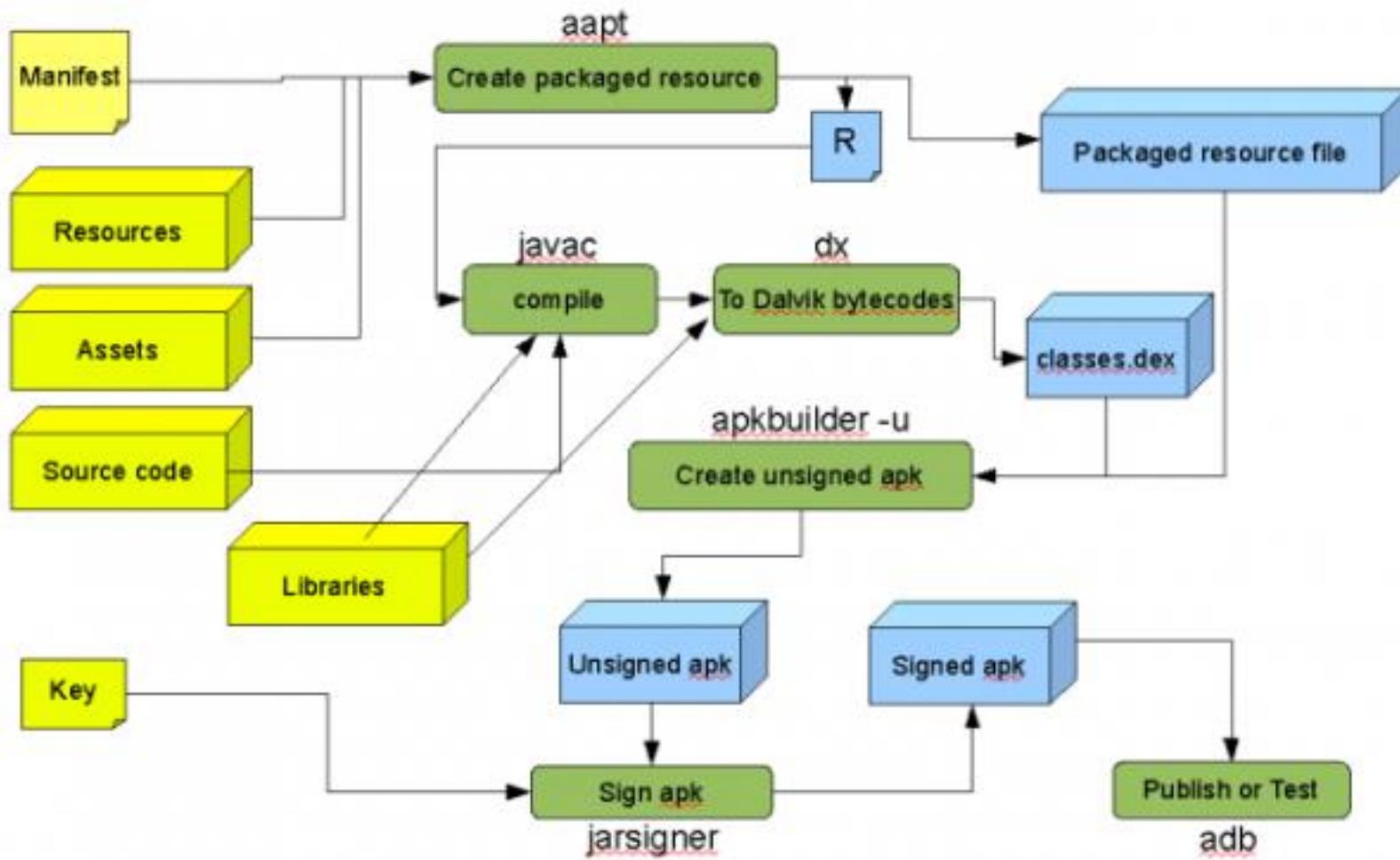
- A. Only Activities should be listed in it.
- B. It can contain only one Activity component.
- C. All components must be listed in it (except dynamically attached BroadcastReceivers).
- D. Can contain XML and Java/Kotlin code as well.

What is true about Manifest file?

- A. Only Activities should be listed in it.
- B. It can contain only one Activity component.
- C. All components must be listed in it (except dynamically attached BroadcastReceivers).
- D. Can contain XML and Java/Kotlin code as well.

APK file

Build mechanism (source >.apk)



Source: <http://androidmaterial.blogspot.hu/2011/05/how-to-build-android-application.html>

The .apk file

- It is close to Java .jar files, but has major differences
- ZIP file, can be extracted
- Compressed file with the following typical content:
 - > META-INF folder
 - CERT.RSA: certificate
 - MANIFEST.MF: meta information
 - CERT.SF: resource list and SHA1 prints
 - Signature-Version: 1.0
 - Created-By: 1.0 (Android)
 - SHA1-Digest-Manifest: wxqnEA10UA5nO5QJ8CGMwj kGGWE=
 - ...
 - Name: res/layout/exchange_component_back_bottom.xml
 - SHA1-Digest: eACjMjESj7Zkf0cBFTZ0nqWrt7w=
 - ...
 - Name: res/drawable-hdpi/icon.png
 - SHA1-Digest: DGEqylP8W0n0iv/ZzBx3MW0WGCA=
 - > Res folder: resources
 - > AndroidManifest.xml (meta): name, version, permissions, components, etc.
 - > classes.dex: Dalvik binary
 - > resources.arsc

APK reverse engineering

- Dex2jar
- JDGUI
- Play Store -> myBackup
- APK location:
 - > ..\app\build\intermediates\apk\debug\app-debug.apk
- Solution: Obfuscation

Kotlin

Kotlin Introduction

- Statically typed programming language for modern multiplatform applications
- Now official on Android!
- <https://kotlinlang.org/>
- <https://kotlinlang.org/docs/tutorials/kotlin-android.html>
- <https://play.kotlinlang.org/>



Kotlin on Android

```
class MainActivity : AppCompatActivity() {  
  
    private lateinit var binding: ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        binding = ActivityMainBinding.inflate(layoutInflater)  
        setContentView(binding.root)  
  
        binding.btnActivateViewStub.setOnClickListener {  
            binding.stubDemo.setVisibility(View.VISIBLE)  
        }  
    }  
}
```

- Differences?
 - > No findViewById
 - > No OnClickListener boilerplate code

Constants, variables (val vs. var)

- Constant – „val”

```
val score: Int = 1 // set value
val idx = 2 // type can be hidden
val age: Int // type is required when data is not set immediately
age = 3 // set data later
```

- Variables – „var”

```
var score = 0 // type can be hidden
score += 1
```

- String templates

```
var score = 1
val scoreText = "$score score"

score = 2
// simple templating with strings:
val newScoreText = "${scoreText.replace("score", "now it is ")} $score"
```

Methods

- Method syntax

```
fun add(a: Int, b: Int): Int {  
    return a + b  
}
```

- Body and return type is „optional”

```
fun add(a: Int, b: Int) = a + b
```

- Return without value – Unit

```
fun printAddResult(a: Int, b: Int): Unit {  
    println("$a + $b is: ${a + b}")  
}
```

- Unit is not mandatory

```
fun printAddResult(a: Int, b: Int) {  
    println("$a + $b is: ${a + b}")  
}
```

Java – Car class

```
class Car {  
    private String type;  
    public Car(String type) {  
        this.type = type;  
    }  
}
```

Kotlin – Car class

```
class Car constructor(type: String) {  
  
    fun demo() {  
        Log.d("TAG", type)  
    }  
}
```

Classes

```
class Car constructor(type: String) {  
    val typeUpper = type.toUpperCase()  
  
    init {  
        Log.d("TAG_DEMO", "Car created: ${type}")  
    }  
  
    constructor(type: String, model: String) : this(type) {  
        Log.d("TAG_DEMO", "Car model: ${model}")  
    }  
  
    // instantiation  
    val car1 = Car("Toyota")  
    val car2 = Car("Toyota", "Corolla")
```

constructor can be left

primary constructor with parameters

primary constructor arguments can be used here

primary constructor init block

secondary constructor

Extension

By default everything is final

```
open class Item(price: Int) {  
    open fun calculatePrice() {}  
    fun load() {}  
}
```

inheritance

```
class SpecialItem(price : Int) : Item(price) {  
    final override fun calculatePrice() {}  
}
```

Later it can not be overriden

Data class

```
data class Ship(val name: String, val age: Int)
```

- Created automatically:
 - > equals()/hashCode()
 - > toString(): "Ship(name=Discovery, age=31);"
 - > componentN() methods
 - > copy() methods

```
val discovery = Ship("Discovery", 31)  
val (name, age) = discovery
```

```
//val name = discovery.component1()  
//val age = discovery.component2()
```

- Requirements for Data classes:
 - > Primary constructor with at least 1 argument
 - > Every primary constructor argument is *val* or *var*
 - > *Data* classes can not be *abstract*, *open*, *sealed*, or *inner*

Kotlin Null Safety

Kotlin Null Safety

- Kotlin's type system is aimed at eliminating the danger of null references from code

```
var a: String = "abc"  
a = null // compilation error
```

```
var b: String? = "abc"  
b = null // ok
```

More info: <https://kotlinlang.org/docs/reference/null-safety.html>

Safe Calls - ?.

- This returns *b.length* if *b* is not null, and *null* otherwise:

```
val a = "Kotlin"  
val b: String? = null  
println(b?.length)  
println(a?.length)
```

? operator

- ‘?’ allows null as a value
 - > List that can contain **null** values:

```
var x: List<String?> = listOf(null, null, null)
```

- > List that can be **null** itself:

```
var x: List<String>? = null
```

- > List that can be **null** or it's values can be **null**:

```
var x: List<String?>? = null  
x = listOf(null, null, null)
```

Elvis Operator - ?:

- Basic implementation:

```
val l: Int = if (b != null) b.length else -1
```

- Elvis operator:

```
val l = b?.length ?: -1
```

The !! Operator

- The not-null assertion operator (!!) converts any value to a non-null type and throws an exception if the value is *null*
- We can write *b!!*, and this will return a non-null value of *b* (e.g., a String in our example) or throw an NPE (Null Pointer Exception) if *b* is null:

```
val l = b!! .length
```

Clean Code

Reference: <https://cleancoders.com/>



What is Clean Code? Why do we need it?

- What are the values of the software?
 - > Maintainability
 - > Continuous delivery
- Nowadays:
 - > continuously changing requirements
- Agile development
- Team work
- Code quality
- Software life cycle

Basic Clean Code Principles

- Naming
 - > Small scope: long, descriptive name
 - > Big scope: short name
- Small classes
- A method does only one thing
- Small methods
 - > Maximum ~4 lines!
- Few arguments
 - > Maximum 3
- No boolean arguments
- No output arguments



- enableWeekendMode()
 - > turnOffFireWall()
 - > turnOnElectricity()
- turnOffFireWall()
- turnOnElectricity()

Java fields vs. Kotlin properties

Java

```
public class Car {  
    private String type;  
  
    public String getType() {  
        return type;  
    }  
  
    public void setType(String type) {  
        Log.d("TAG_CAR", "type SET");  
        this.type = type;  
    }  
}
```

Kotlin

```
class Car {  
    var type: String? = null  
    set(type) {  
        Log.d("TAG_CAR", "type SET")  
        field = type  
    }  
}  
  
class Car {  
    var type: String? = null  
    get() {  
        Log.d("TAG_CAR", "type GET")  
        return field  
    }  
    set(type) {  
        Log.d("TAG_CAR", "type SET")  
        field = type  
    }  
}
```

Singleton – object (in Kotlin)

- Java Version:

```
public class TicTacToeModel {  
  
    private static TicTacToeModel instance = null;  
  
    public static TicTacToeModel getInstance() {  
        if (instance == null) {  
            instance = new TicTacToeModel();  
        }  
  
        return instance;  
    }  
  
    private TicTacToeModel() {  
  
    }  
  
}
```

- Kotlin Version:

```
object TicTacToeModel {  
  
}
```

Questions?



Mobile- and Web-based Software

Lecture 3: Activity

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

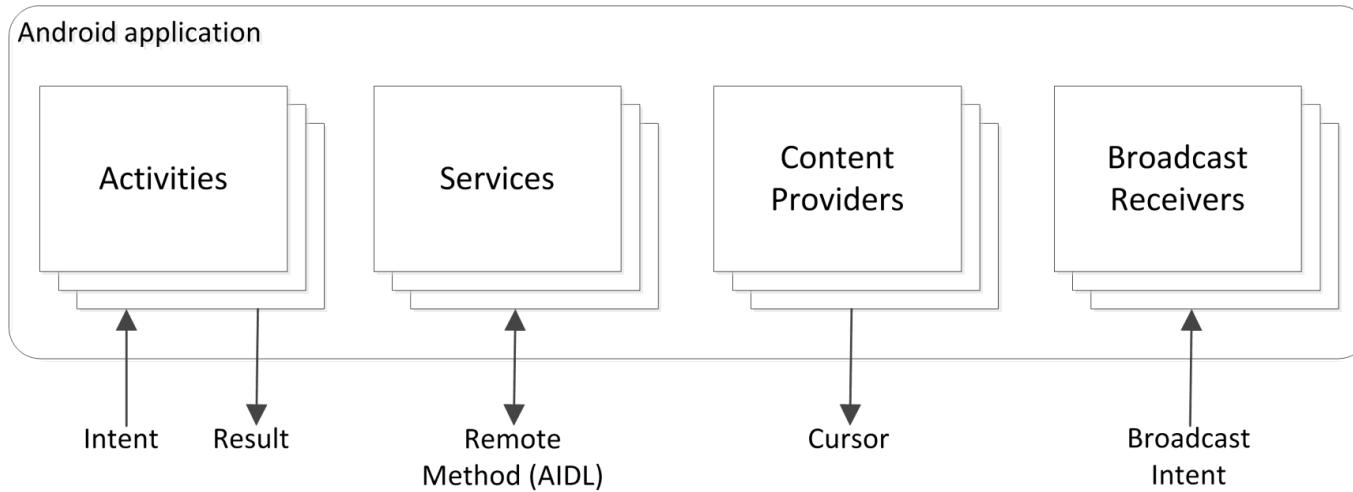
Android Studio Shortcuts

- ALT/Opt + Enter/Return: Quick Fix
- CTRL/Cmd + ALT/Opt + L: Reformat code
- SHIFT + F6: Rename everywhere
- F2: Jump to next error
- CTRL/Cmd + Z and CTRL/Cmd + SHIFT + Z: Undo/Redo
- CTRL/Cmd + P: Show parameter
- ALT/Opt + INSERT: Generate method
- CTRL/Cmd+ O: Override method

Android Application Components

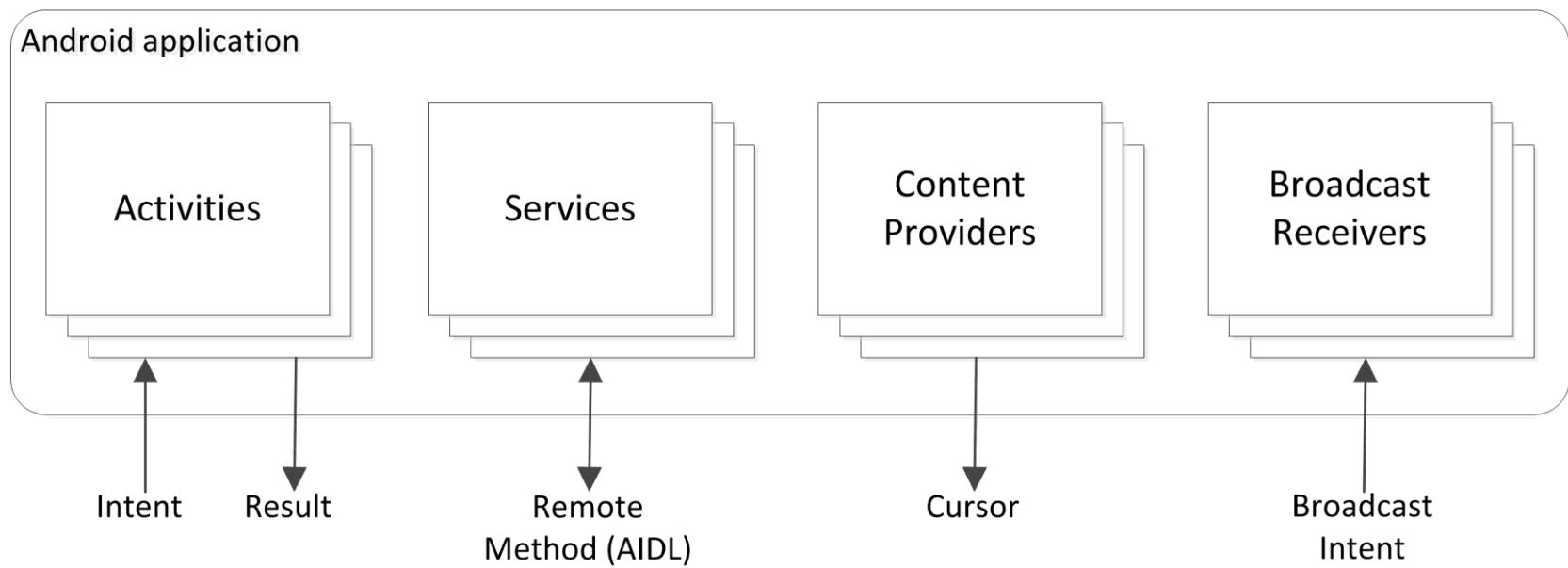
Android Application Components

- An Android application stands from one or more components:
 - > Activity
 - > Service
 - > Content Provider
 - > Broadcast Receiver



Component Activation

- Each component has special role
- Each component can be activated independently
- Even another application can activate a component



Activity

- Special view with own UI
- Example:
 - > Todo application
 - > 3 Activities: ToDo list, new ToDo form, ToDo details view
- Independent *Activities*, but they have an important role in the whole application
- Another application can start the *Activity*, e.g.:
 - > Camera application can start the new Todo activity with the instant camera photo that will be stored with the new Todo element
- Extends from the **android.app.Activity** class

Service

- Service represents a background task in the application
- Does not have specific UI
- For example: a download process, while we are using the application in the foreground
- Other components can start or bind to it
- Extends from the **android.app.Service** class

ContentProvider

- Main objective of *ContentProvider* is to share a data source with other components
- The data can be stored in different locations: SQLite, file, *SharedPreferences*, remote server, etc.
- Other applications can work on the same data by using *ContentProvider*
- For example: CallLog *ContentProvider*
- Extends from the
`android.content.ContentProvider` class

BroadcastReceiver

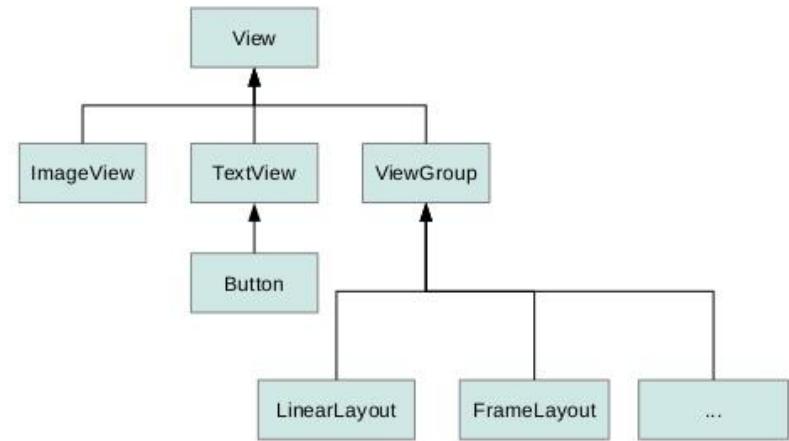
- *BroadcastReceiver* activates by different system or application events
- For example: screen turns off, low battery, incoming call, photo ready, etc.
- An application can also fire custom broadcast event, for example a Service broadcasts that the download is ready
- Does not have specific user interface, typically it awakes other components, or display a notification
- Extends from the
`android.content.BroadcastReceiver` class

Android UI Basics

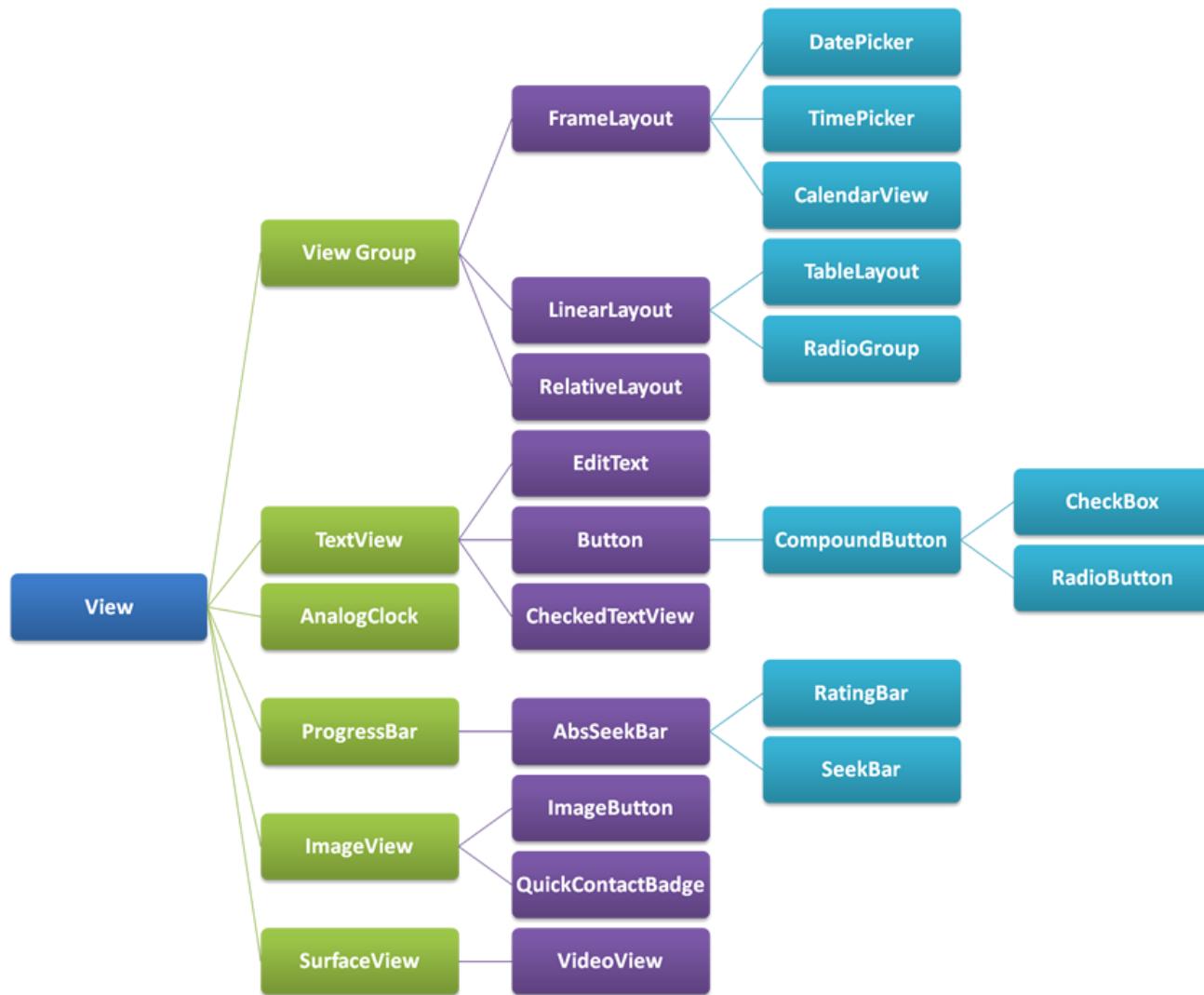
View Hierarchy 1/2

- View: can be displayed on the screen
 - > TextView
 - > Button
 - > ImageView
 - > Custom views (extends View)
 - > Etc.
- ViewGroup ~ Layout:
Defines the visual structure for a user interface
 - > LinearLayout
 - > RelativeLayout
 - > TableLayout
 - > Etc.

View hierarchy



View Hierarchy 2/2



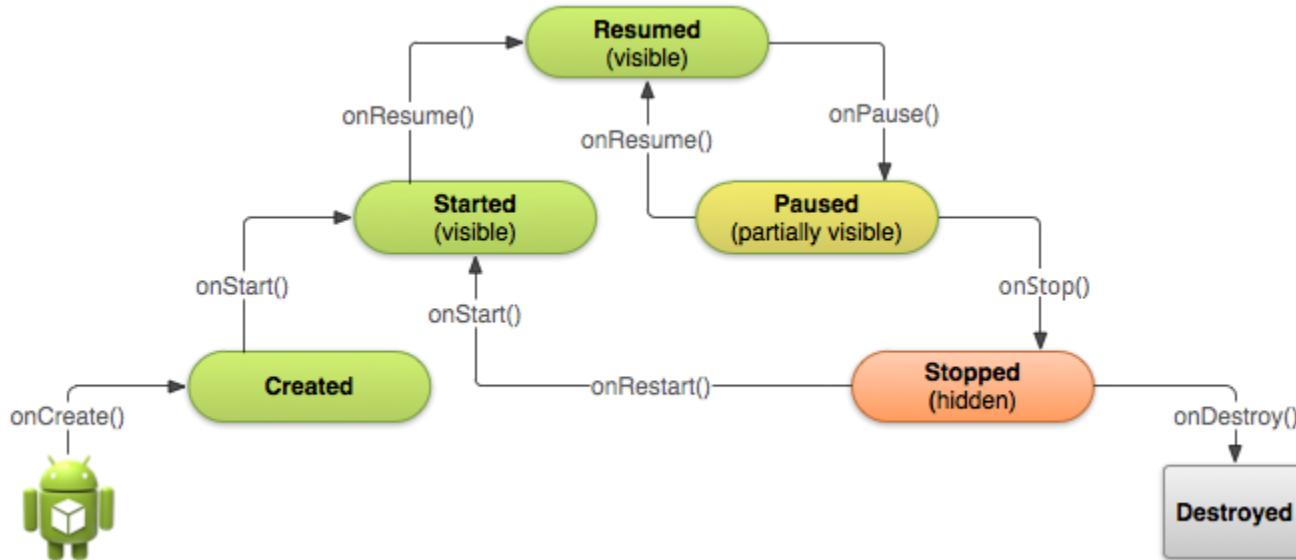
Custom Views

- Custom class that extends View (or any other View subclass)
- Custom drawing can be easy or complex according to your application's needs.
- Typical methods that we override:
 - > onDraw(...): drawing
 - > onTouchEvent(...): handling touch events
 - > onMeasure(...): fine control of layout parameters, like custom sizing

Activity Component

Activity start

- No *main()* method!
- The Android system initiates code in an Activity instance by invoking specific callback methods
- During the life of an activity, the system calls a core set of lifecycle methods in a sequence similar to a step pyramid

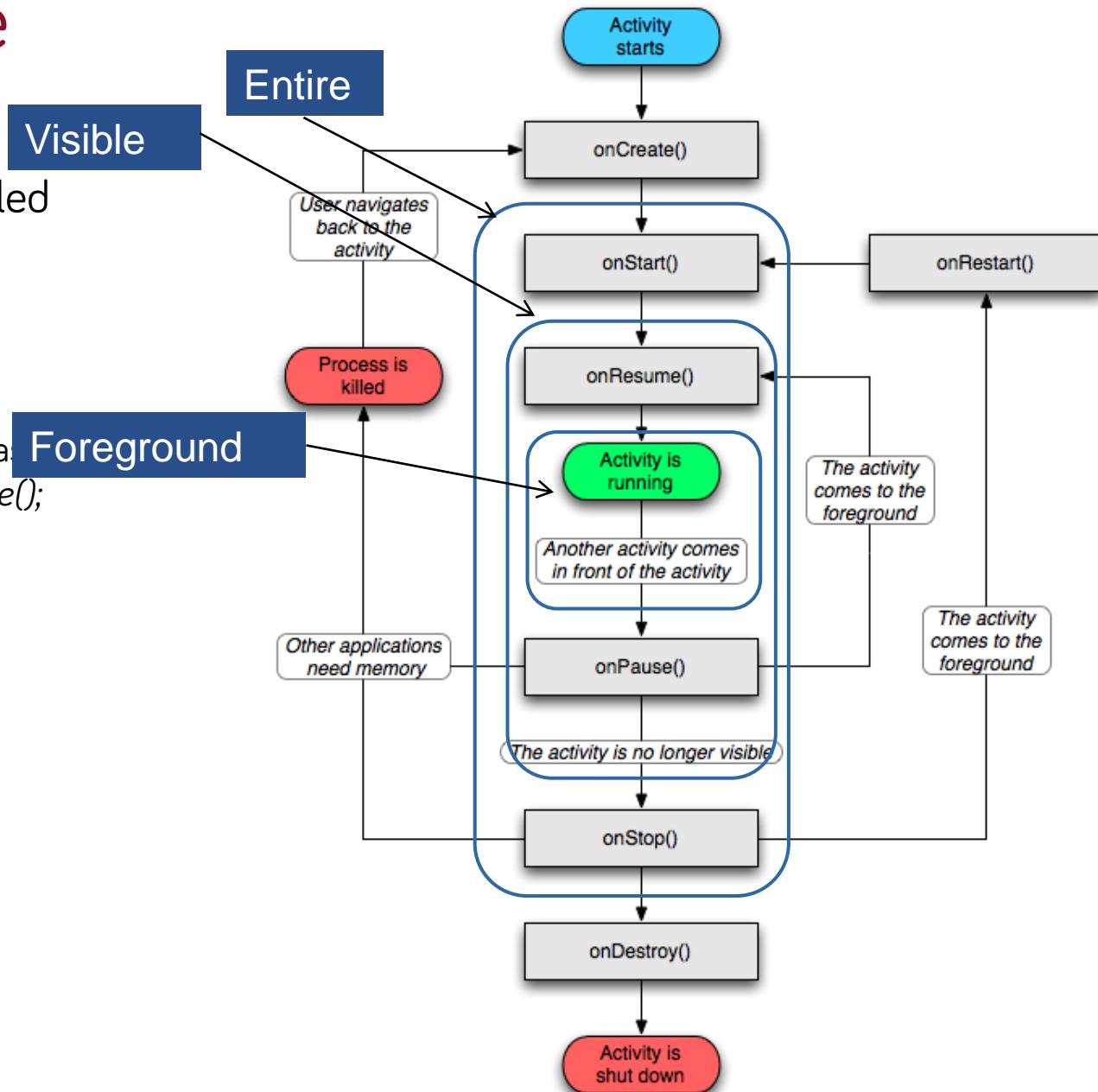


Activity start - notice

- After the system calls `onCreate()`, it quickly calls `onStart()`, which is quickly followed by `onResume()`
- The system quickly moves between states by calling the next lifecycle callback method
- The *Activity* never resides in the Created or Started state

Activity life cycle

- Life cycle functions are called during state change
- Life cycle functions can be overwritten
 - It is necessary to call base class function, e.g.: `super.onCreate();`
- Huge responsibility on the developer!



Activity skeleton 1/2

```
class ExampleActivity : Activity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        // Activity created  
    }  
  
    override fun onStart() {  
        super.onStart()  
        // Activity becomes visible  
    }  
  
    override fun onResume() {  
        super.onResume()  
        // Activity gets focus  
    }  
}
```

Activity skeleton 2/2

```
override fun onPause() {
    super.onPause()
    // Activity loses focus
}

override fun onStop() {
    super.onStop()
    // Activity not visible
}

override fun onDestroy() {
    super.onDestroy()
    // Activity will be destroyed
}

}
```

Ensure correct behavior

- Not necessary to implement all life-cycle methods, but **implement those that ensure the correct behavior in your application:**
 - Does not crash if the user receives a phone call or switches to another app while using your app.
 - Does not consume valuable system resources when the user is not actively using it.
 - Does not lose the user's progress if they leave your app and return to it at a later time.
 - Does not crash or lose the user's progress when the screen rotates between landscape and portrait orientation.

Launcher Activity

- How/where do we define which activity will launched when the user clicks on the icon?
 - > In *AndroidManifest.xml*

```
<activity android:name=".MainActivity"  
    android:label="@string/app_name">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name=  
            "android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>
```

Otherwise the application will not be visible

Homework – Readings

- Write and View Logs with Logcat:
 - > <https://developer.android.com/studio/debug/am-logcat.html>
- App Manifest (just the basics):
 - > <https://developer.android.com/guide/topics/manifest/manifest-intro.html>
- Activity Life Cycle:
 - > <https://developer.android.com/guide/components/activities/intro-activities.html>
 - > <https://developer.android.com/guide/components/activities/activity-lifecycle.html>

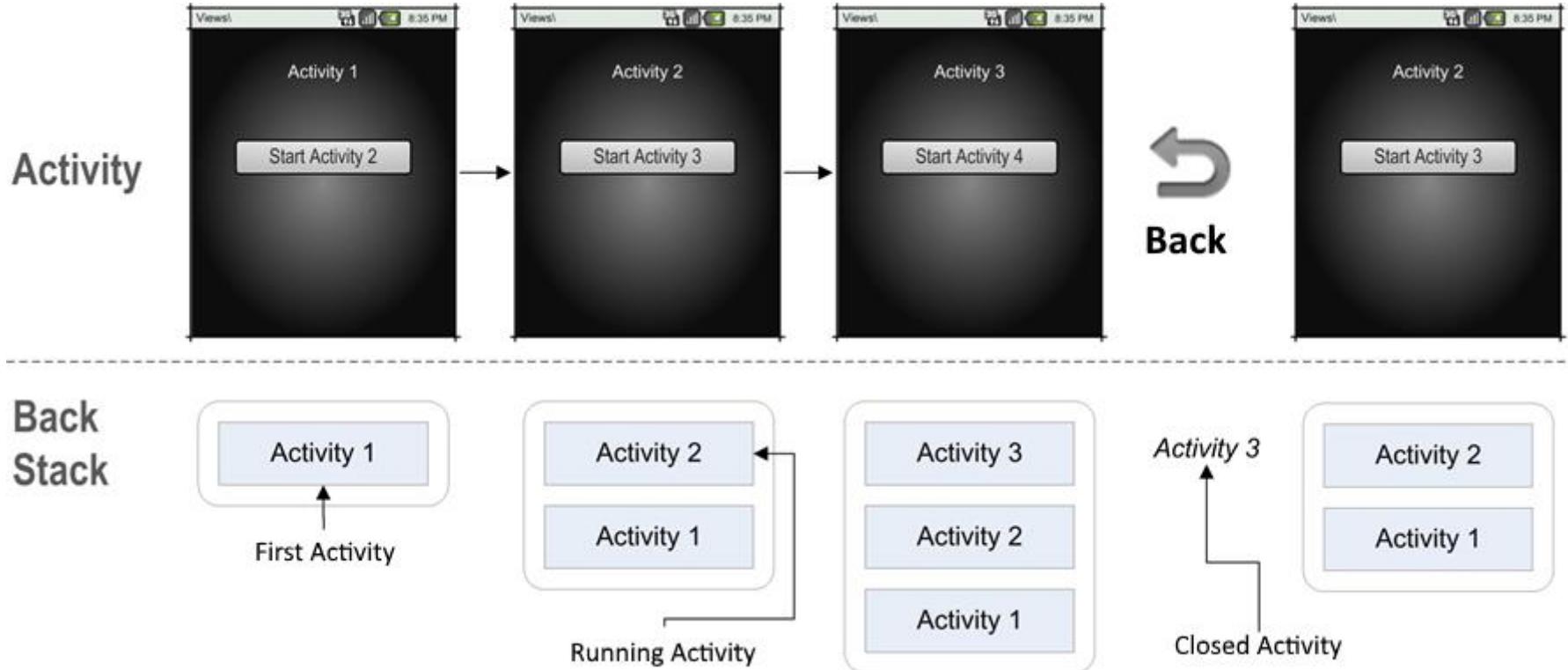


ACTIVITY BACK STACK

Activity Back Stack 1/2

- Typically we have more activities in an applicaiton
- The system stores the activities on a stack, called: Back Stack
- The foreground Activity is on the top of the Back Stack
- If another Activity goes foreground the previous one steps one level down in the Back Stack
- By pressing the back button, the system will check the next Activity in the stack and brings it in foreground
- Last in, first out

Activity Back Stack 2/2

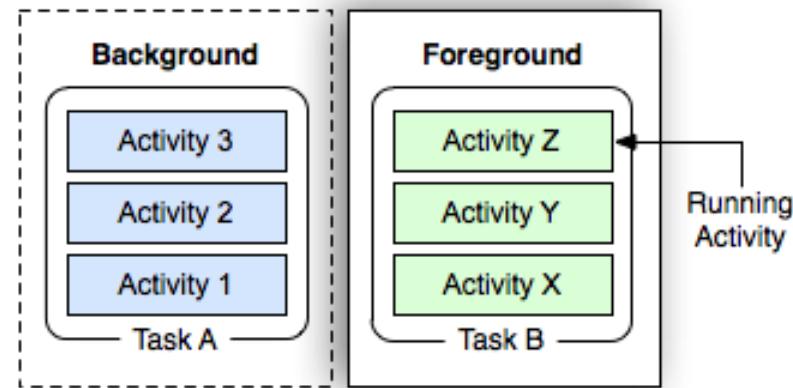


Activity switch

- Order of callback methods:
 - > *onPause()* of Activity A
 - > *onCreate()*, *onStart()* and *onResume()* of Activity B
(B has the focus)
 - > *onStop()* of Activity A
- Where should A save the data if B wants to read it out from the database?
 - > *onPause()* of A

Multitasking

- Task: A “process” that can use more Activities but not all Activity comes from the same application
- By pressing the home button, we can jump to start screen and start other applications
- By starting a new application, the system preserves the Back Stack of the previous one
- The new task gets a new Back Stack
- In case of memory issues the system will delete the oldest Back Stacks



Activity control

- In the Manifest, the <activity> tag can have different properties (how it should behave in relation to other activities):
 - > taskAffinity: task description
 - > launchMode: how the Activity should start (e.g. always new instance)
 - > allowTaskReparenting: create a new task for it
 - > clearTaskOnLaunch: clears other Activities from the task
 - > alwaysRetainTaskState: system will handle the task state
 - > finishOnTaskLaunch: stop Activity when user exits from it (e.g. Home button)
 - > screenOrientation: fix landscape/portrait layout
 - > etc.
- *startActivity(...)* function parameters:
 - > FLAG_ACTIVITY_NEW_TASK
 - > FLAG_ACTIVITY_CLEAR_TOP
 - > FLAG_ACTIVITY_SINGLE_TOP
- More details later

Launching another Activity

- Intent mechanism - later:

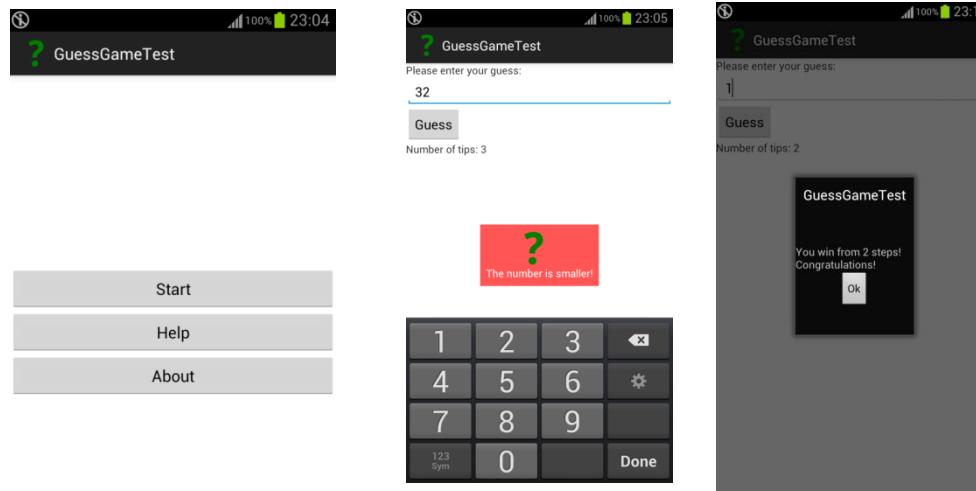
```
binding.btnExit.setOnClickListener {  
    var intentDetails = Intent()  
    intentDetails.setClass(this@MainActivity,  
        DetailsActivity::class.java)  
  
    intentDetails.putExtra(KEY_DETAILS, binding.etName.text.toString())  
  
    startActivity(intentDetails)  
}
```

Let's practice!

- Create an application that contains two Activities! The first one has an *EditText* and a button. On button click, the second Activity starts and the data in the *EditText* is passed to it. The second Activity displays the received data in a *Toast*.

Engagement

- Implement a HighLow game
- Handle screen rotation properly
- Use *TextInputLayout*
- Show a result dialog: android:theme="@style/Theme.AppCompat.Dialog"



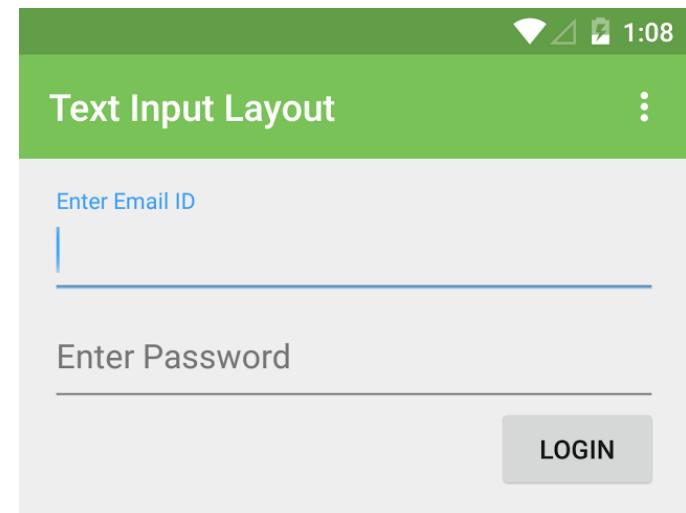
TextInputLayout

- Gradle dependency:
 - > implementation 'com.google.android.material:material:1.0.0'
- Usage:

```
<android.support.design.widget.TextInputLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">
```

```
<EditText  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Enter Email ID"  
    android:id="@+id/etName" />
```

```
</android.support.design.widget.TextInputLayout>
```



Fixing errors

android:inputType="numberDecimal"

etNum.error = "This field can not be empty!"

Codelabs

- Android fundamentals:
 - > <https://codelabs.developers.google.com/codelabs/android-training-hello-world/index.html?index=..%2F..android-training#0>
- First UI:
 - > <https://codelabs.developers.google.com/codelabs/android-training-layout-editor-part-a/index.html?index=..%2F..android-training#0>
- Activity life cycle:
 - > <https://codelabs.developers.google.com/codelabs/android-training-activity-lifecycle-and-state/index.html?index=..%2F..android-training#0>
- Tools:
 - > <https://codelabs.developers.google.com/codelabs/android-training-available-resources/index.html?index=..%2F..android-training#0>
- Debugger:
 - > <https://codelabs.developers.google.com/codelabs/android-training-using-debugger/index.html?index=..%2F..android-training#0>



Homework – Readings 1/2

- Activities:
 - > <https://developer.android.com/guide/components/activities/index.html> (and all sublink)
- Sending simple data to another application/Activity:
 - > <https://developer.android.com/training/sharing/send.html>
- Activity State Changes
 - > <https://developer.android.com/guide/components/activities/state-changes.html>



Homework – Readings 2/2

- Tasks and Back Stack
 - > <https://developer.android.com/guide/components/activities/tasks-and-back-stack.html>
- Processes and Application Lifecycle
 - > <https://developer.android.com/guide/components/activities/process-lifecycle.html>
- Parcelables and Bundles
 - > <https://developer.android.com/guide/components/activities/process-lifecycle.html>
- App Resources
 - > <http://developer.android.com/guide/topics/resources/index.html>
- Toolbar Tutorials
 - > <http://javatechig.com/android/android-lollipop-toolbar-example>
 - > <http://www.101apps.co.za/index.php/articles/using-toolbars-in-your-apps.html>



Homework - Readings

- Custom Views and Drawing:
 - > <https://developer.android.com/training/custom-views/create-view.html>
 - > <https://developer.android.com/training/custom-views/custom-drawing.html>
 - > <https://developer.android.com/training/custom-views/making-interactive.html>
 - > <https://developer.android.com/training/custom-views/optimizing-view.html>
- Toggle button:
 - > <https://developer.android.com/guide/topics/ui/controls/togglebutton.html>
- Toast:
 - > <https://developer.android.com/guide/topics/ui/notifiers/toasts.html>
- Snackbar:
 - > <https://developer.android.com/training/snackbar/action.html>



Questions?



Mobile- and Web-based Software

Lecture 4: Menu

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

Menu

Menu

- Can be defined in source code
- Can be defined in resources
- Dynamic menus
 - > Set visibility
 - > Manipulation from Java/Kotlin code
- Supporting submenus

Menu resource

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/action_about"
        android:title="About"
        android:icon="@mipmap/ic_launcher"
        app:showAsAction="always|withText"/>

    <item
        android:id="@+id/action_help"
        android:title="Help" />

</menu>
```

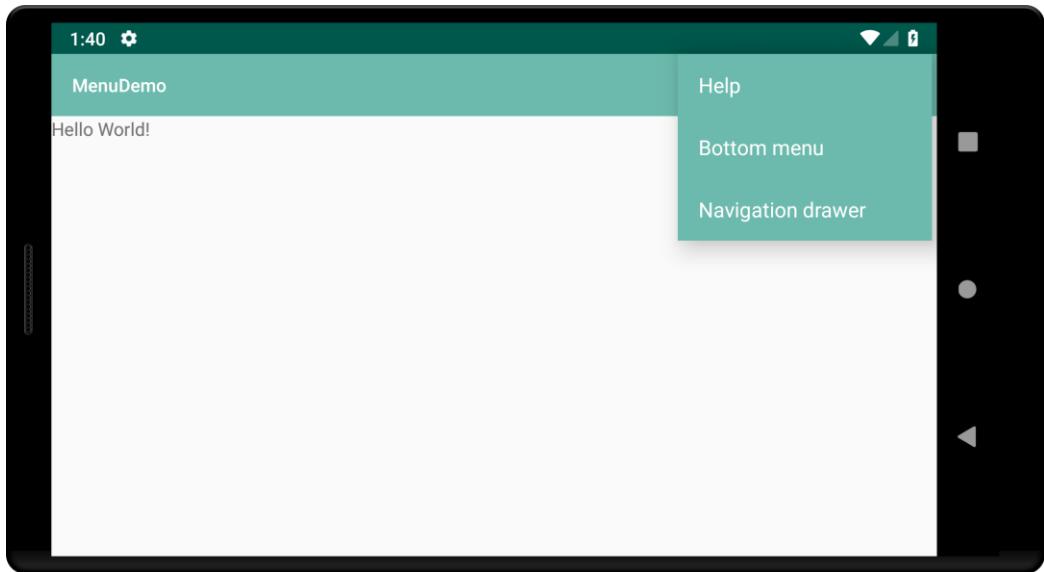
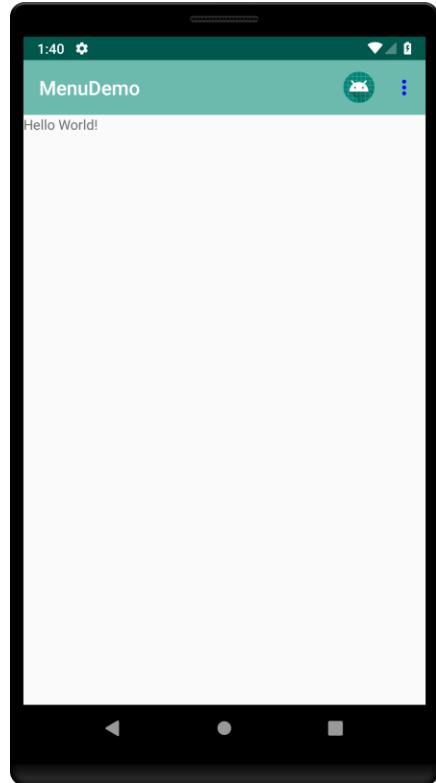
Menu event handling

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    menuInflater.inflate(R.menu.main_menu, menu)
    return super.onCreateOptionsMenu(menu)
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.action_about -> {
            Toast.makeText(this, "ABOUT", Toast.LENGTH_LONG).show()
        }
        R.id.action_help -> {
            Toast.makeText(this, "HELP", Toast.LENGTH_LONG).show()
        }
    }

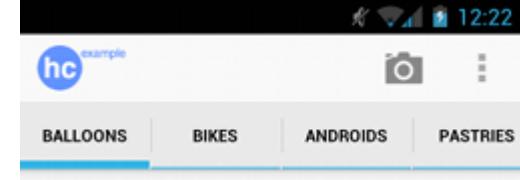
    return true
}
```

Menus and Submenus



ActionBar and Menus

- Dedicated area for application logo, main commands and navigation buttons
- The goal is to show the often used functions on a simple menu bar
- Supports Tabs as well
- Typical usage:
 - > Branding (logo/background) and application icon
 - > Consistent navigation through the whole application
 - > Shows the main functions of the current Activity



ActionBar specific XML attributes for Menus

```
<item android:id="@+id/action_time"  
      android:title="@string/action_show_time"  
      android:orderInCategory="5"  
      android:icon="@drawable/clock_icon"  
      app:showAsAction="always|withText" />
```

ActionBar -> Toolbar

- ActionBar was replaced with ToolBar
- Toolbar is more dynamic
- Toolbar also supports menu resources
- Toolbar can have custom elements and standard menus at the same time
- Toolbar can be positioned manually on the layout

Toolbar usage

- Use a theme with no ActionBar (styles.xml):
 - > Theme.AppCompat.NoActionBar

- Layout resource:

```
<androidx.appcompat.widget.Toolbar  
    android:id="@+id/toolbar"  
    android:minHeight="?attr/actionBarSize"  
    android:background="#2196F3"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
</android.support.v7.widget.Toolbar>
```

- Activity onCreate(...):

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    binding = ActivityMainBinding.inflate(layoutInflater)  
    setContentView(binding.root)  
  
    setSupportActionBar(binding.toolbar)  
}
```

Change Toolbar colors

- Styles.xml:

```
<style name="ToolBarStyle" parent="Theme.AppCompat">
    <item name="android:textColorPrimary">#ff0000</item>
    <item name="android:textColorSecondary">#0000ff</item>
    <item name="actionMenuTextColor">@android:color/white</item>
</style>
```

- Toolbar layout

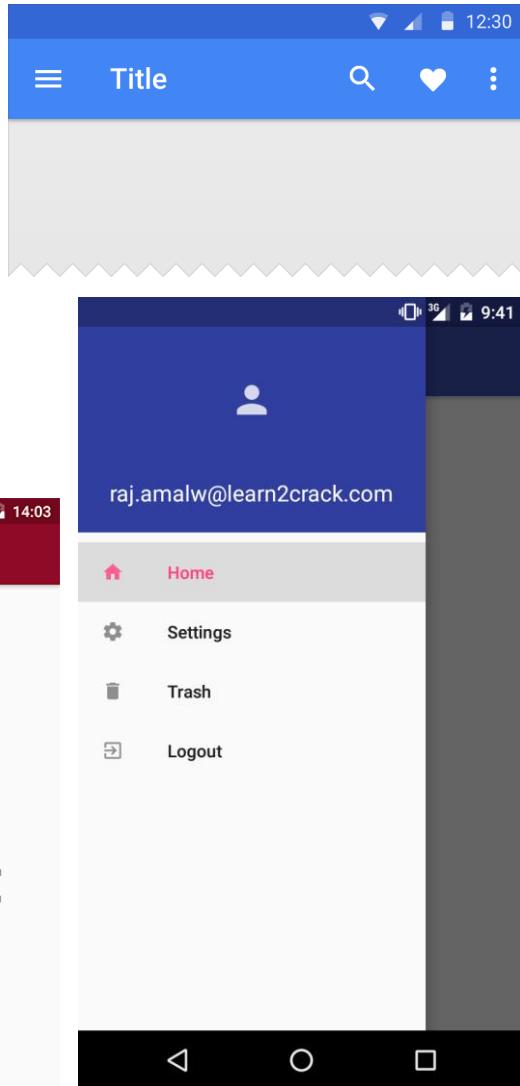
```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#21f348"
    app:theme="@style/ToolBarStyle"
    android:minHeight="?attr/actionBarSize" />
```

- Layout app namespace:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

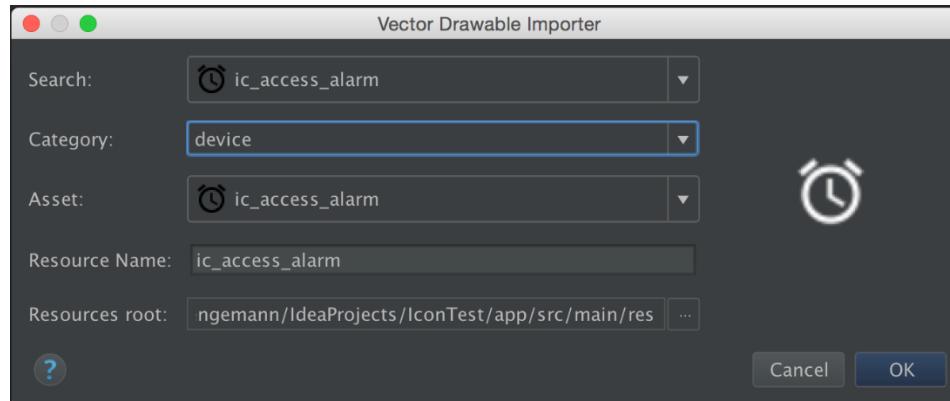
Menu types

- „Old“ Menu
- „Deprecated“ ActionBar
- Toolbar
- Bottom Navigation View
- NavigationDrawer



Android Drawable Importer

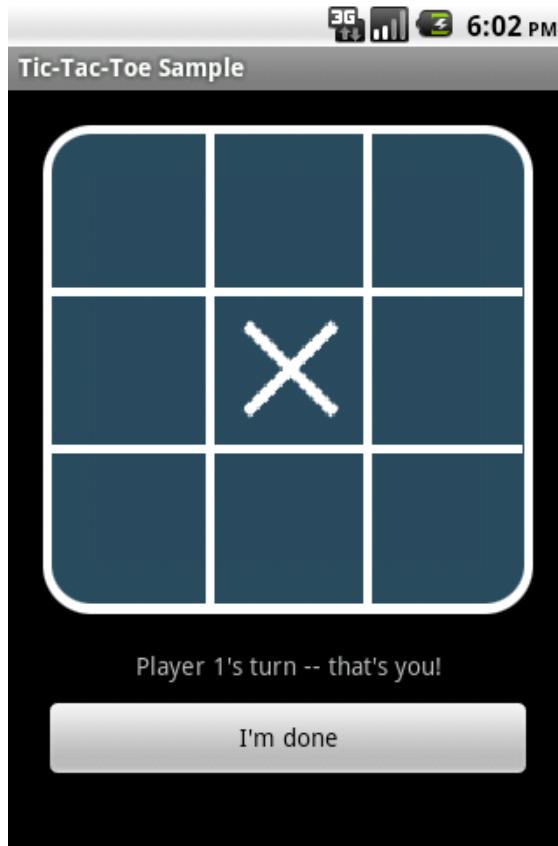
- Add icons and images to your project easily
- <https://plugins.jetbrains.com/plugin/7658-android-drawable-importer>



TicTacToe

Let's practice!

- Implement a TicTacToe game!



Layout of the game

- Try embedding LinearLayout into LinearLayout and use „gravity” property
- Override size of TicTacToeView (square size):

```
override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpec: Int) {  
    val w = View.MeasureSpec.getSize(widthMeasureSpec)  
    val h = View.MeasureSpec.getSize(heightMeasureSpec)  
    val d = if (w == 0) h else if (h == 0) w else if (w < h) w else h  
    setMeasuredDimension(d, d)  
}
```

Some special effects 😊

- Shimmer:



- Shimmer – text:
 - > <https://github.com/facebook/shimmer-android>
- Shimmer general:
 - > <http://facebook.github.io/shimmer-android/>
- More UI elements:
 - > <https://github.com/wasabeef/awesome-android-ui>

Additional Canvas Functions

- `drawText(...)`
 - > `paint.setTextSize(...)`
 - > `paint.getTextBounds(...)`
 - <http://stackoverflow.com/questions/7549182/android-paint-measuretext-vs-gettextbounds>
- `drawBitmap(...)`
 - > Use Image Drawable Plugin
 - > <https://plugins.jetbrains.com/plugin/7658-android-drawable-importer>
 - > ... \ TicTacToe \ app \ src \ main \ res \ drawable
 - > Use simple image names (e.g. „grass.png”)

Some useful links

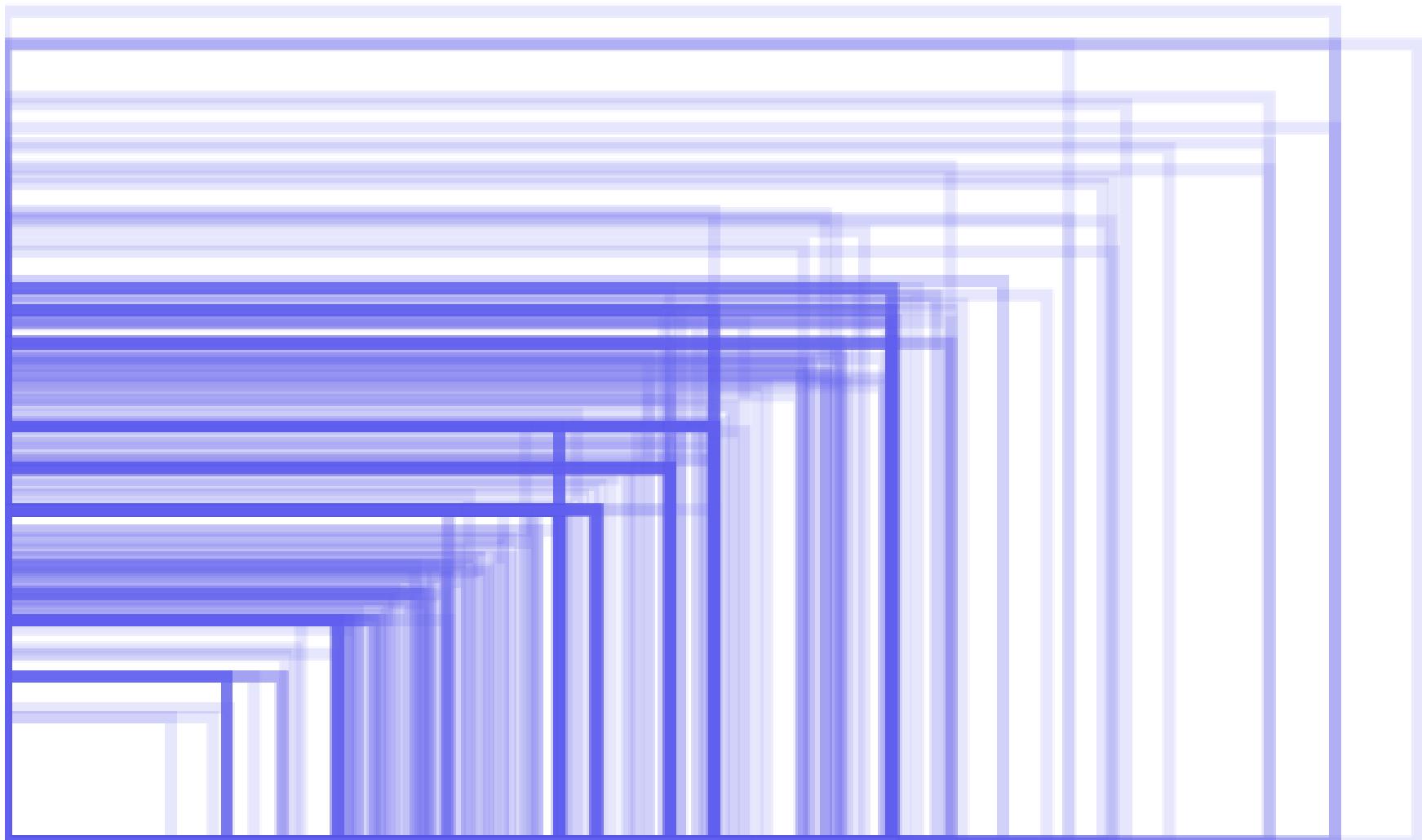
- Awesome Android UI
 - > <https://github.com/wasabeef/awesome-android-ui>
- Design patterns (what is besides Singleton?)
 - > <https://github.com/iluwatar/java-design-patterns>
- Best Android development resources:
 - > <http://www.anysoftwaretools.com/best-android-development-resources>
- Android library collection:
 - > <https://android-arsenal.com/>

User Interface Basic Principles

Question

- How many different screen sizes exists in the Android ecosystem?

Thousands!



Different screen sizes 1/2

- Android can run on different display resolutions and densities
- The OS grants a mechanism to support different displays (from version 1.6)
- It takes over the most of the work from the developer
- Developer/designer must create only the appropriate resources
- For example, there are huge differences between a screen of a tablet and a mobile phone

Different screen sizes 2/2

- The OS automatically scales and resizes the surface of the application to reach the maximal device compatibility
- But! It is important, that the surface and the resources (images) will be optimized for the supported resolutions and densities.
- This method can greatly increase the user experience
- Furthermore the device specific visualization will increase the user satisfaction
- Following this method will result an application in a single .apk file, which can support every device in UI regard

Most important definitions 1/2

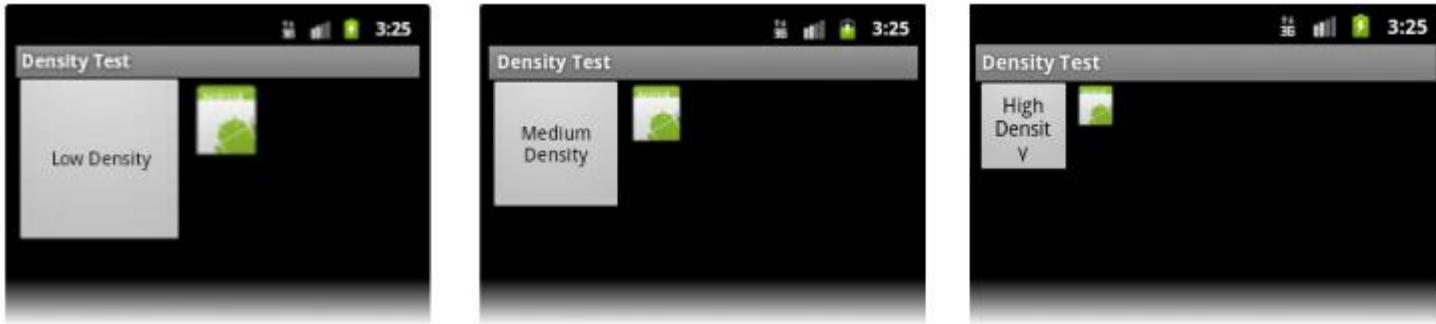
- Screen size:
 - > Physical screen diagonal
 - > For simplicity, Android distinguishes 4 different categories: small, normal, large and extra large
- Screen density (*dpi*): the number of pixels within an area. Typically dots per inch. (*dpi*)
 - > Android distinguishes 4 different categories: low, medium, high és extra high
- Orientation: the orientation from the user's point of view:
 - > Portrait
 - > Landscape
 - > Orientation can change in runtime, for example with the rotation of the device
 - > It's possible to lock the orientation

Most important definitions 2/2

- Resolution (*px*): Number of the pixels of the display
 - > At UI design, never work with resolution, only size and density
- Density-independent pixel(*dp*)
 - > Virtual pixel unit, which is expedient to use at UI design
 - > One dp is one physical pixel on a 160 dpi screen.
 - > The OS handles every necessary scaling at runtime according to the defined dp
 - > $px = dp * (dpi / 160)$
 - > For example on a 240 dpi screen, 1 dp is 1.5 physical pixel

Example

- Without density independency support:

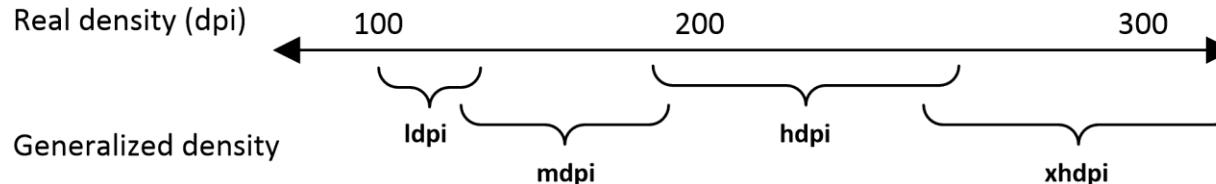
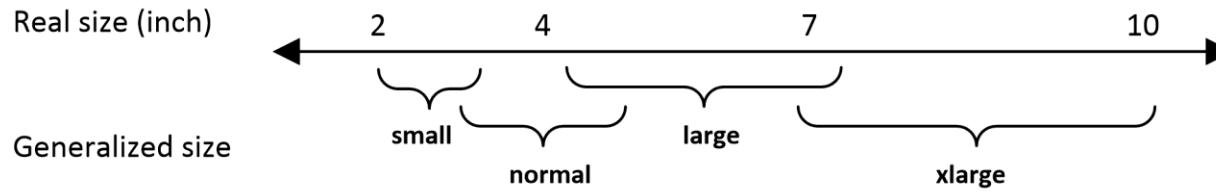


- With density independency support:



General screen sizes

- 4 general sizes:
 - > small, normal, large and xlarge
- 4 general densities:
 - > ldpi (low), mdpi (medium), hdpi (high), and xhdpi (extra high)



Runtime operation

- To optimize the visualization, it is possible to set alternative resources to support different screen resolutions and densities.
- Typically, we must define different layouts and distinct resolution images
- The OS choose the appropriate resources runtime
- Generally there is no need to define every resolution and density combination

Density independency

- An application can be density independent, when the UI components preserving their physical size on the different densities (from the user's point of view).
- Keeping density independency is very important. For example, a button can be larger on a smaller density screen.
- The problems which are related to density can significantly influence the usability of the application
- Android helps to reach density independency in two ways:
 - The OS calculates the dp, and scale the user interface from the actual screen density
 - The system scales image resources automatically from screen density

Most important factors

- Use the `wrap_content`, `match_parent` or `dp` units at UI design
- Do not use predefined pixels!
- Do not use `AbsoluteLayout` (obsolete)!
- We must create different image resources to handle different display densities anyway
- For text scaling it's worth to use scale-independent pixels (`sp`). They work like `dp`.

User Interface Resources, Resource Qualifiers

- Layout
 - > *res/layout*
- Text based resources:
 - > *res/values/strings.xml*
 - > *res/values-**hu**/strings.xml*
- Image resources:
 - > *res/drawable-hdpi/[image].[ext]*
- Animation resources:
 - > *res/anim*

Codelabs

- User navigation
 - > <https://codelabs.developers.google.com/codelabs/android-training-provide-user-navigation/index.html?index=..%2F..android-training#0>



Homework – Readings

- Application Components
 - > <https://developer.android.com/guide/components/fundamentals.html>
 - > https://www.tutorialspoint.com/android/android_application_components.htm
- App Resources
 - > <http://developer.android.com/guide/topics/resources/index.html>
- Toolbar Tutorials
 - > <http://javatechig.com/android/android-lollipop-toolbar-example>
 - > http://www.101apps.co.za/index.php/articles/using_toolbars-in-your-apps.html



Questions?



Mobile- and Web-based Software

Lecture 5: Layouts, Views

David Sik

david.sik@aut.bme.hu



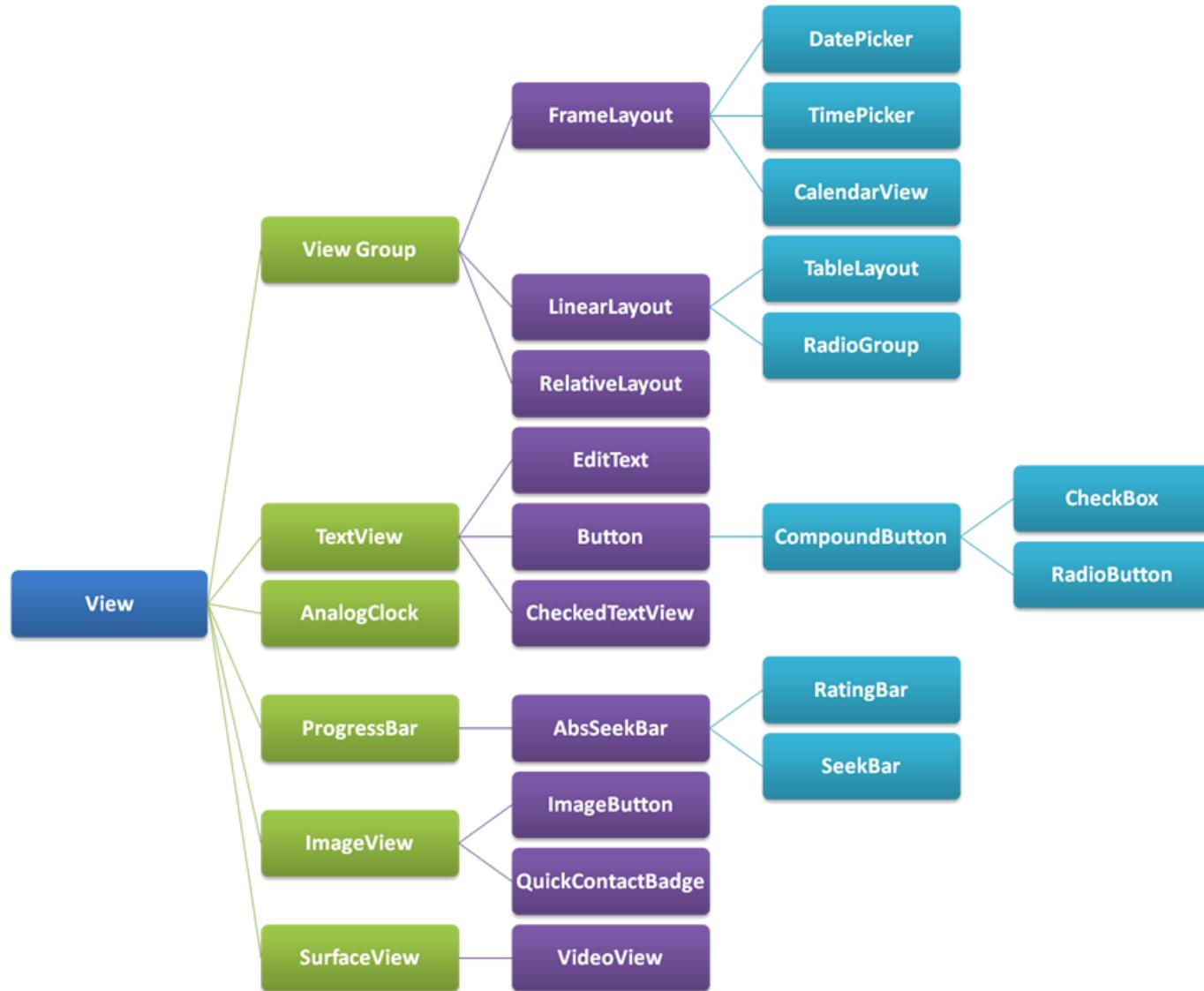
Department of
Automation and
Applied Informatics

LAYOUTS

What can we put on the screen?

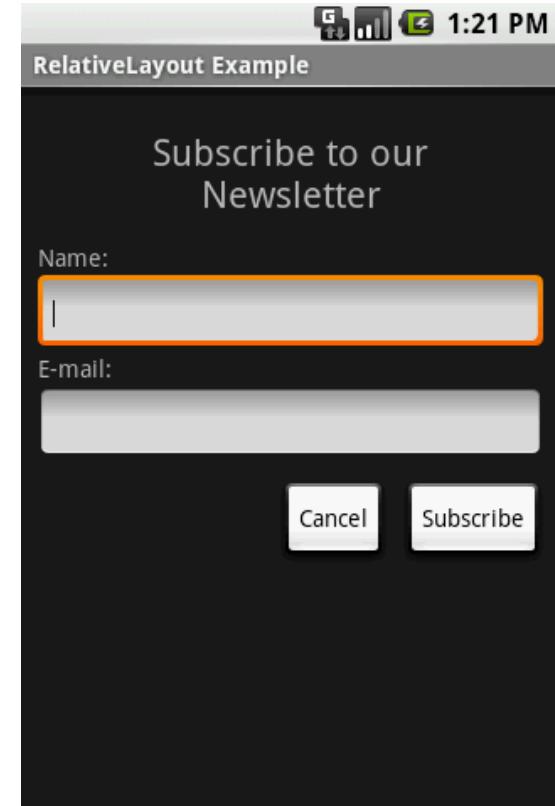
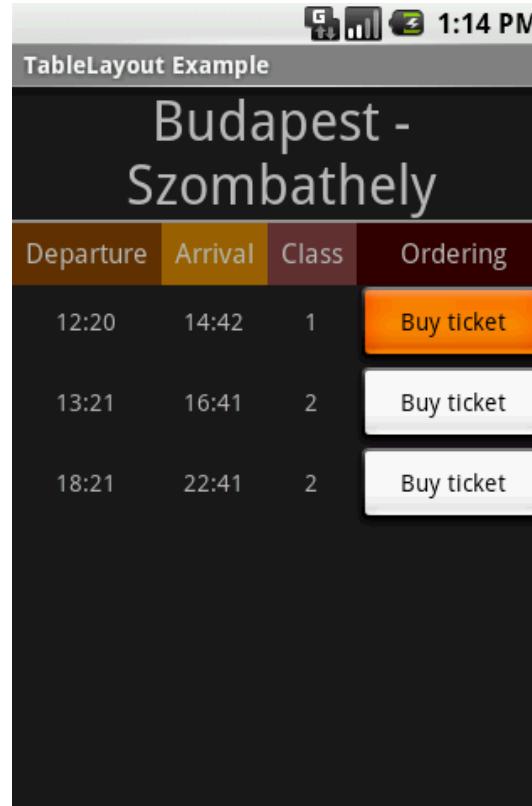
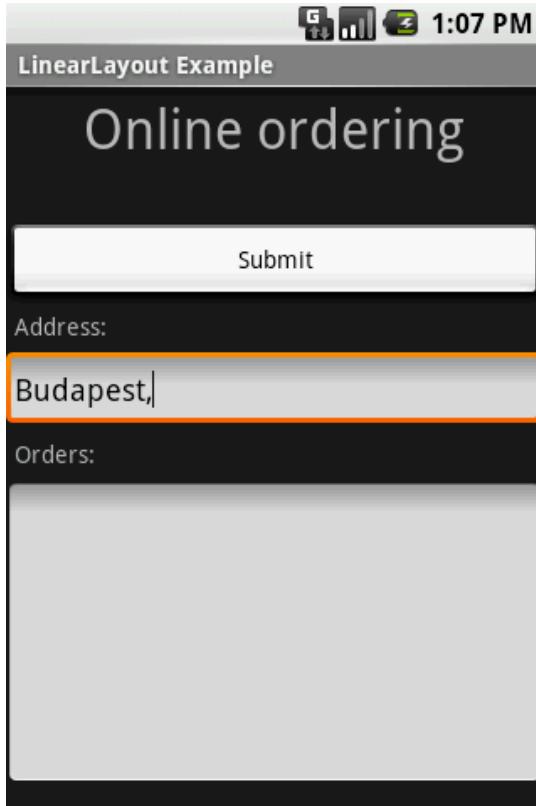
- ViewGroups (layouts)
 - > LinearLayout
 - > RelativeLayout
 - > Etc.
- Views
 - > Button
 - > TextView
 - > EditText
 - > ScrollView
 - > Etc.
- All must provide a layout_width and layout_height property!!!

Android Views



Some important layout

- *View/ViewGroup*
- *LinearLayout, TableLayout, RelativeLayout*



User Interface Resources, Resource Qualifiers

- Layout
 - > *res/layout*
- Text based resources:
 - > *res/values/strings.xml*
 - > *res/values-**hu**/strings.xml*
- Image resources:
 - > *res/drawable-hdpi/[image].[ext]*
- Animation resources:
 - > *res/anim*

Let's Practice!

- Portrait vs. Landscape mode support with resource qualifiers
- Write an application that displays different layout in portrait and in landscape mode!

LinearLayout properties

- orientation:
 - > *Orientation of component inside the ViewGroup*
- weightSum
 - > *Sum of weight for calculating view's size in ratio's*
 - > *Each View inside the LinearLayout can have a weight value*

LinearLayout Weight Example

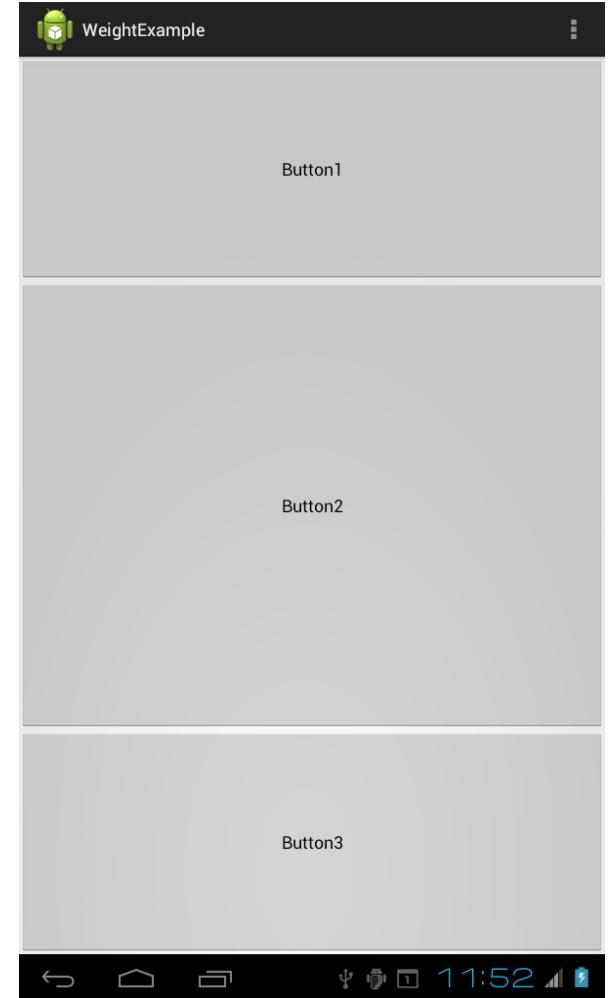
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="4"
    android:orientation="vertical">

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button1" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:text="Button2" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button3" />

</LinearLayout>
```

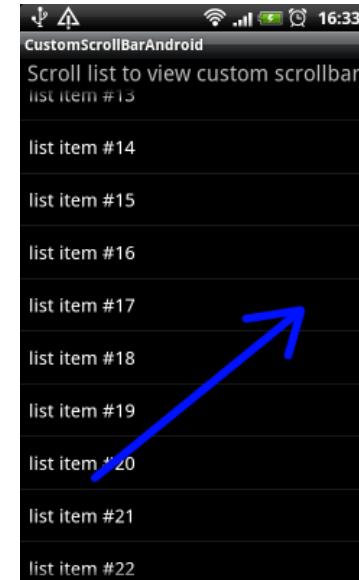


Additional examples and parameters

- Parameters
 - > Margin, padding
 - > Gravity
- How does the scroll work?
 - > ScrollView
 - > HorizontalScrollView

ScrollView

- ScrollView and HorizontalScrollView
- Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display
- The ScrollView must not fill the whole screen
- More ScrollView-s can be applied in one screen



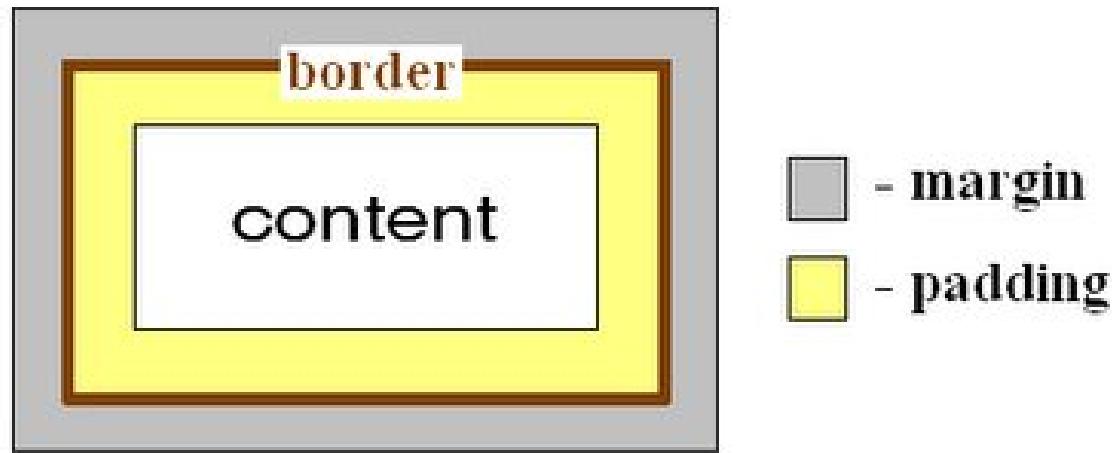
ScrollView example

```
<ScrollView xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="10dp"
    android:fillViewport="false">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="200dp"
            android:scaleType="centerCrop"
            android:src="@drawable/image" />

        ...
    </LinearLayout>
</ScrollView>
```

Padding and Margin



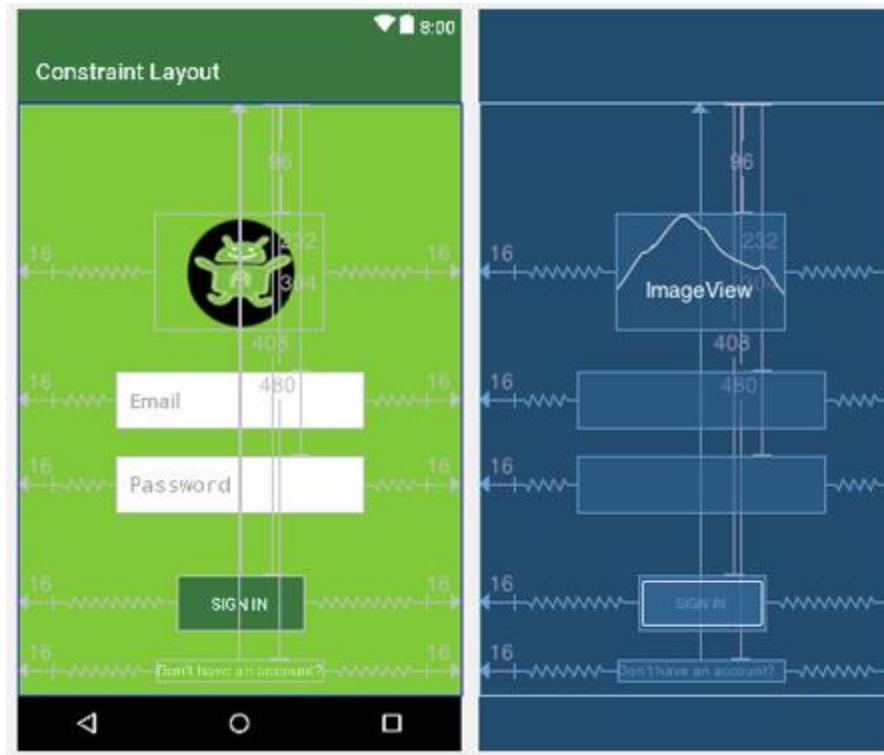
RelativeLayout properties

- The location of views can be set related to their parent and to each other
- Elements are referring to each others via IDs
- Some typical parameters:
 - > *android:layout_alignParentTop*
 - > *android:layout_centerVertical*
 - > *android:layout_below*
 - > *android:layout_toRightOf*

CONSTRAINTLAYOUT

ConstraintLayout...

- ... a better RelativeLayout



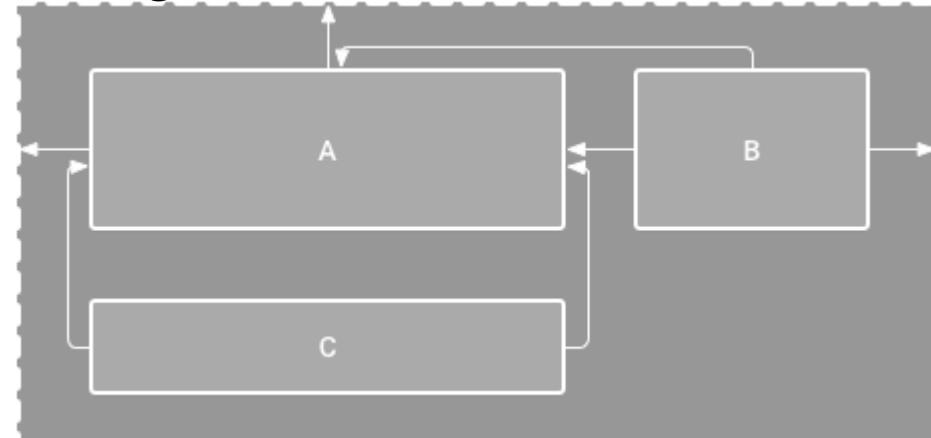
Responsive UI with ConstraintLayout

- Complex layouts with flat hierarchy
 - > No need for embedding layouts
- A bit similar to RelativeLayout
- Layout Editor support
- Supported from Android 2.3 (API Level 9)
- Examples:
 - > <https://github.com/googlesamples/android-ConstraintLayoutExamples>
 - > <https://github.com/googlesamples/android-ConstraintLayoutExamples/tree/master/app/src/main/res/layout>

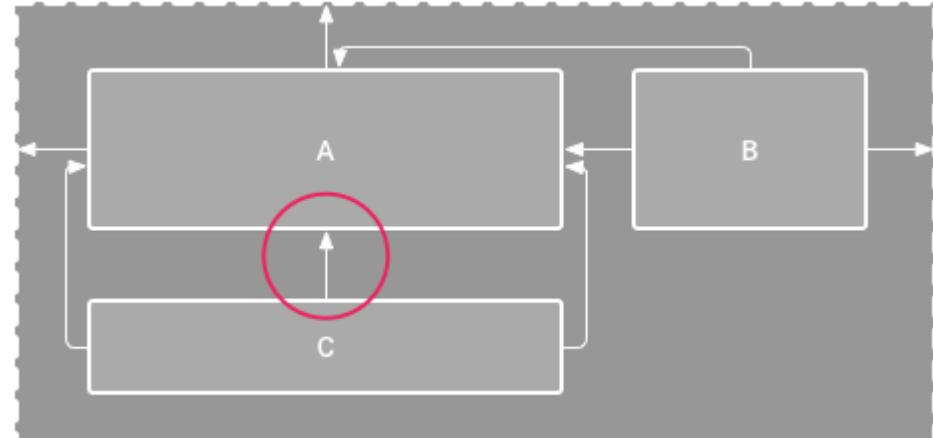
Overview

- Setting the position:
 - > Horizontal and vertical rules (constraints)
- Every rule is a connection connection/alignment:
 - > To another View
 - > To the parent
 - > To an invisible guideline
- LayoutEditor is not enough, test it on real device
- Android Studio warnings are important

Wrong:



Correct because C knows that it is bellow A:



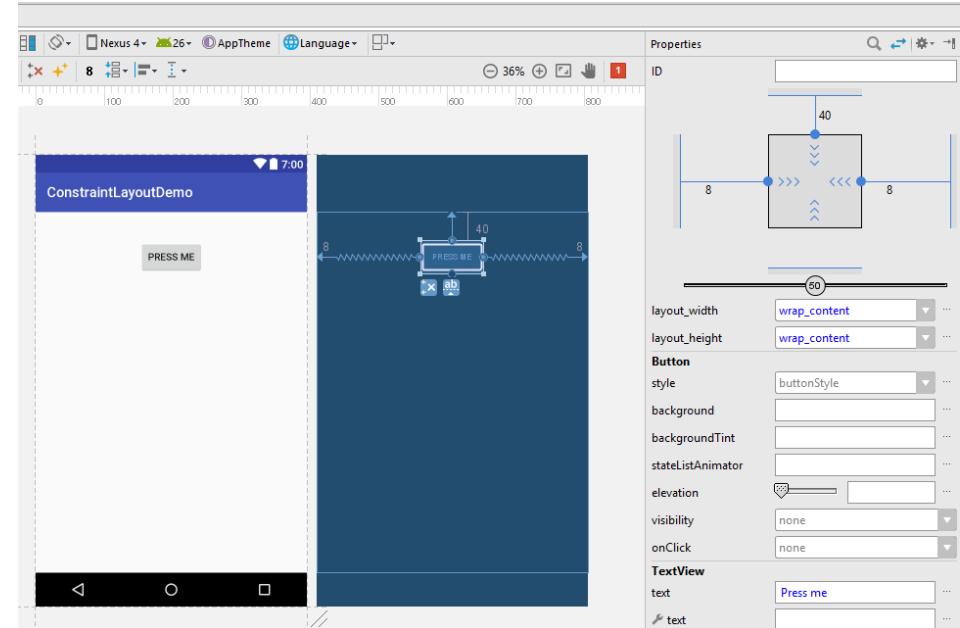
ConstraintLayout usage

- One horizontal and vertical rule is mandatory:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginRight="8dp",
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="40dp"
    />

</android.support.constraint.ConstraintLayout>
```



What will be the size of the button?

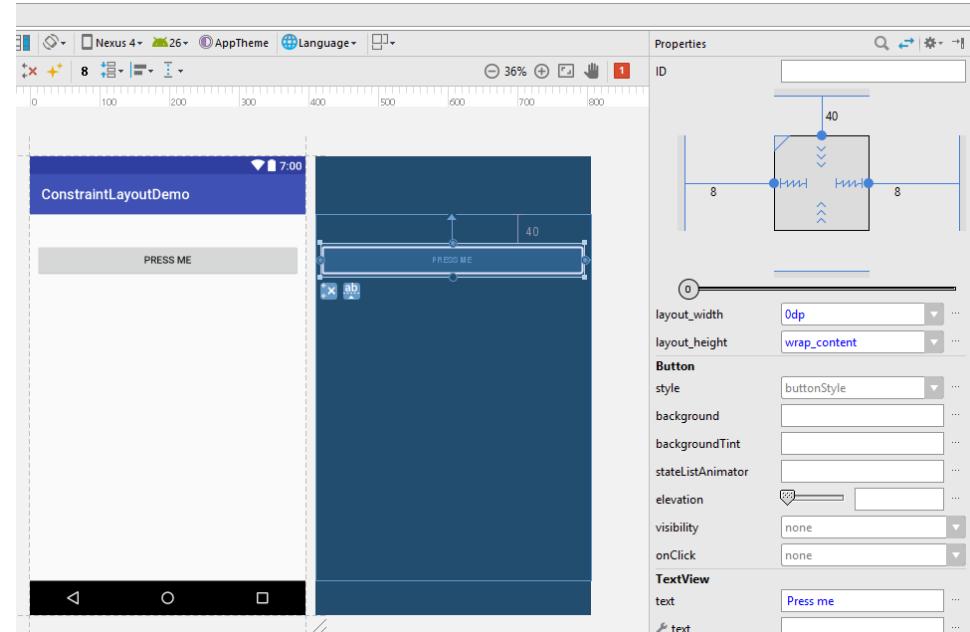
- Ambiguous situations, symbol:
 - > Places it between the two
- Auto sizing:
 - > `android:layout_width="0dp"`



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

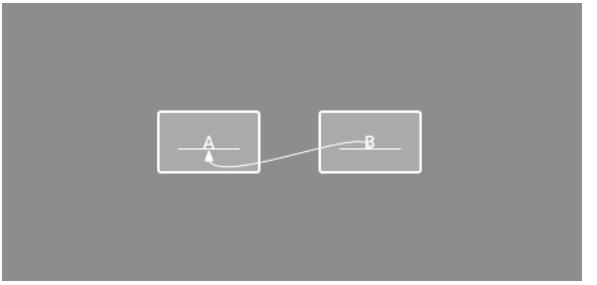
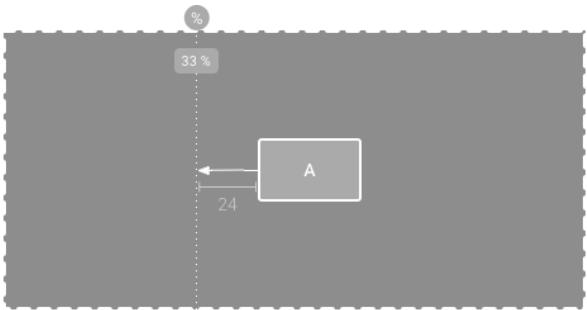
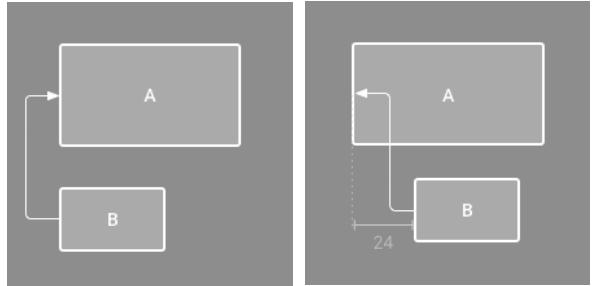
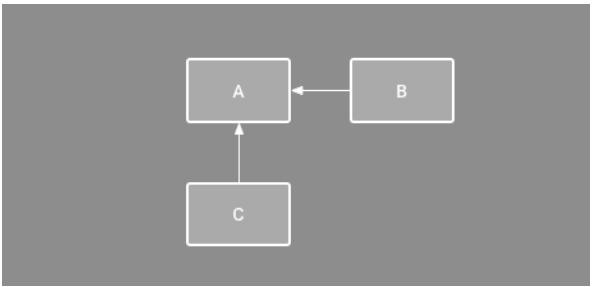
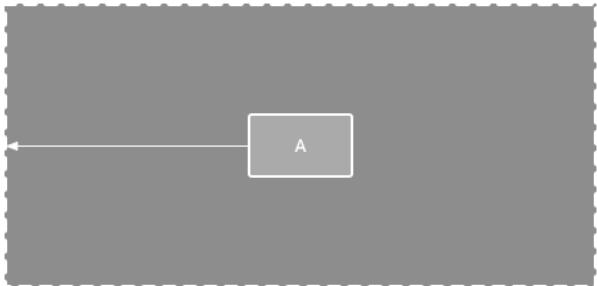
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginRight="8dp",
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="40dp"
    />

</android.support.constraint.ConstraintLayout>
```



Constraint options

- To the parent
- To the edge of another View
- To the baseline of Another View
- To an invisible GuideLine



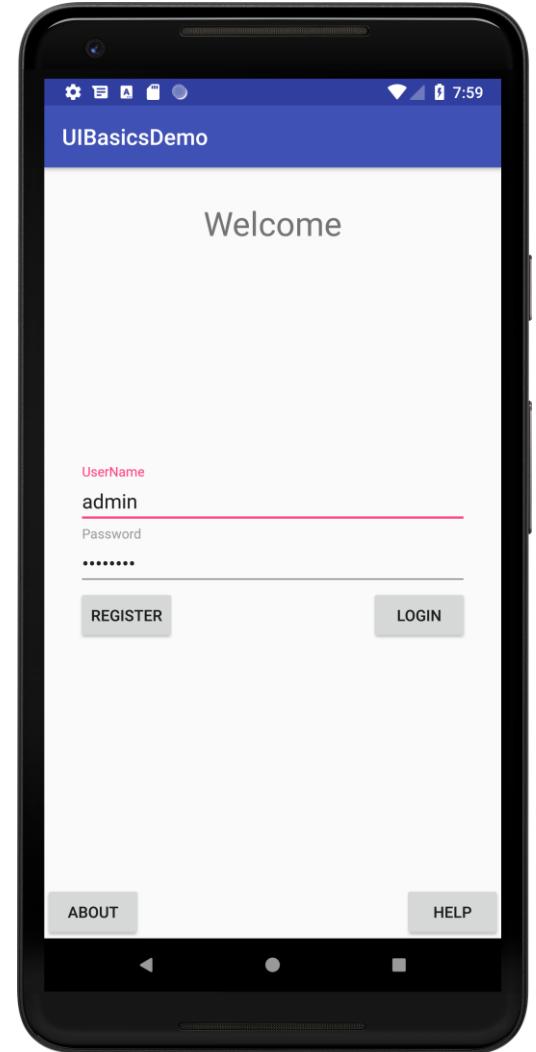
ConstraintLayout links

- Documentation:
 - > <https://developer.android.com/training/constraint-layout/>
- Performance:
 - > <https://android-developers.googleblog.com/2017/08/understanding-performance-benefits-of.html>

Let's practice



- Create the layout of a Login screen



ANDROID VIEWS

Android Views 1/2



- *Button, EditText, CheckBox, RadioButton, ToggleButton, ImageButton*
- *ListView*
- *GridView*
- *Spinner*
- *AutoCompleteTextView*
- *Gallery*
- *ImageSwitcher*
- *DatePicker, TimePicker*

Views 2/2

Views/Controls/1. Light Theme

Save

Checkbox 1

Checkbox 2

RadioButton 1

RadioButton 2

Star

OFF

OFF

Views/ImageButton



Views/Lists/1. Array

Abbaye de Belloc

Abbaye du Mont des Cats

Abertam

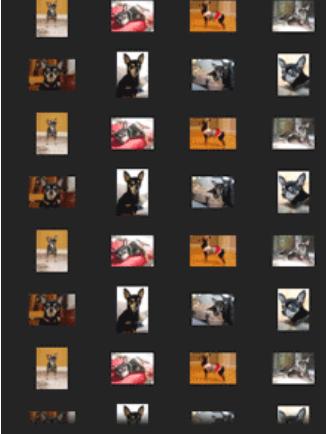
Abondance

Ackawi

Acorn

Adelost

Views/Grid/2. Photo Grid



Views/Spinner

Color: green

Planet: Mars

Views/Auto Complete/1. Screen Top

Type in the text field for auto-completion.

Country: CO

Give me suggestions

Cote d'Ivoire

Cocos (Keeling) Islands

Colombia

Comoros

Congo

Views/Gallery/1. Photos



Views/Date Widgets/1. Dialog

3-12-2008 20:49

change the date

Thu, August 14, 2008

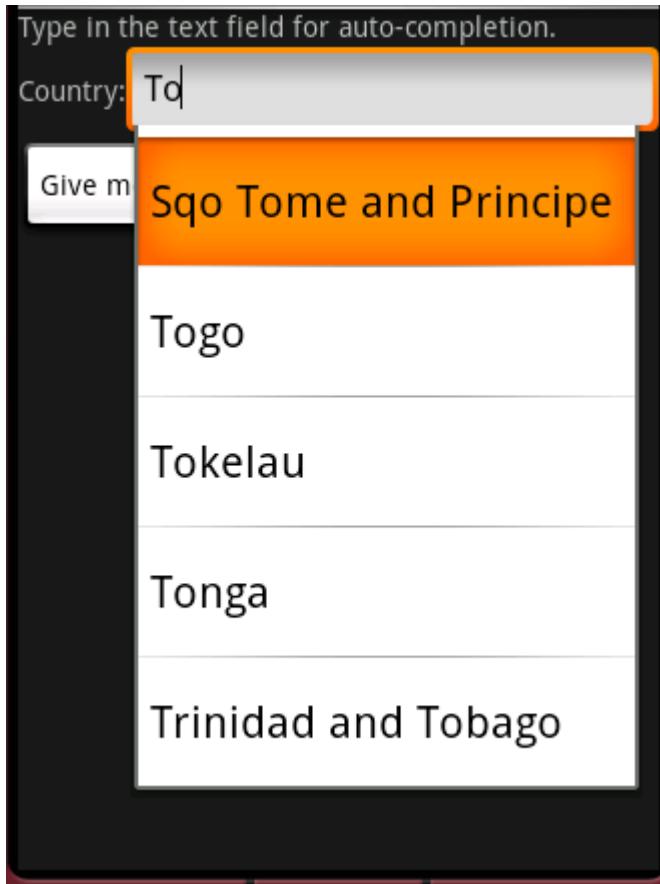
+ Aug - 14 - 2008

Set Cancel

ImageView

- Display images
- Supports image resources
- Dynamic behavior
- Supports different scale/crop strategies

AutoCompleteTextView 1/3



AutoCompleteTextView 2/3

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <AutoCompleteTextView

        android:id="@+id/autoCompleteTextViewCities"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" >

        <requestFocus />

    </AutoCompleteTextView>

</LinearLayout>
```

AutoCompleteTextView 3/3

```
class MainActivity : AppCompatActivity() {  
    private lateinit var binding: ActivityMainBinding  
    private val cityNames = arrayOf("Budapest", "Bukarest", "New York",  
        "New Delhi")  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        binding = ActivityMainBinding.inflate(layoutInflater)  
        setContentView(binding.root)  
  
        val cityAdapter = ArrayAdapter(  
            this,  
            android.R.layout.simple_dropdown_item_1line, cityNames  
        )  
        binding.autoCompleteTextViewCities.setAdapter(cityAdapter)  
  
    }  
}
```

RadioButton - demo

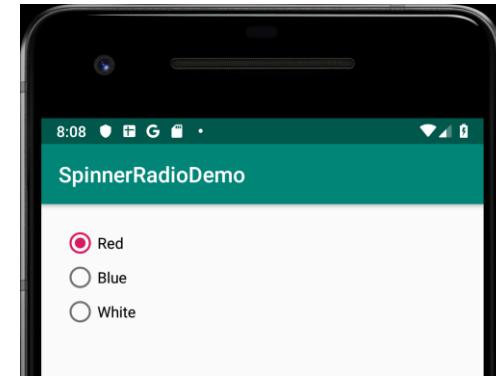
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_margin="20dp"
    tools:context=".MainActivity">

    <RadioGroup
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <RadioButton
            android:id="@+id/btnRed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="Red" />

        <RadioButton
            android:id="@+id/btnBlue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Blue" />

        <RadioButton
            android:id="@+id/btnWhite"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="White" />
    </RadioGroup>
</LinearLayout>
```



```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.btnRed.setOnClickListener{
            Toast.makeText(this@MainActivity,
                "Red", Toast.LENGTH_LONG).show()
        }
        binding.btnBlue.setOnClickListener{
            Toast.makeText(this@MainActivity,
                "Blue", Toast.LENGTH_LONG).show()
        }
        binding.btnWhite.setOnClickListener{
            Toast.makeText(this@MainActivity,
                "White", Toast.LENGTH_LONG).show()
        }
    }
}
```

Spinner - demo

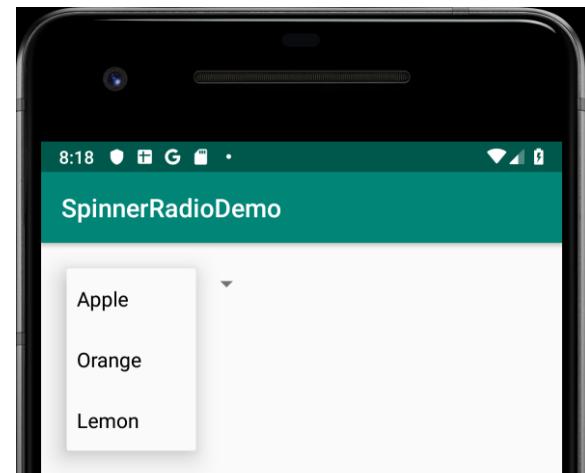
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_margin="20dp"
    tools:context=".MainActivity">

    <Spinner
        android:id="@+id/spinnerFruits"
        android:layout_width="150dp"
        android:layout_height=
        "wrap_content" />

</LinearLayout>
```

res/values/fruits_array.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="fruits_array">
        <item>Apple</item>
        <item>Orange</item>
        <item>Lemon</item>
    </string-array>
</resources>
```



```
class MainActivity : AppCompatActivity(),
    AdapterView.OnItemSelectedListener {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val fruitsAdapter = ArrayAdapter.createFromResource(
            this,
            R.array.fruits_array, android.R.layout.
            simple_spinner_item
        )
        fruitsAdapter.setDropDownViewResource(
            android.R.layout.simple_spinner_dropdown_item)
        binding.spinnerFruits.adapter = fruitsAdapter
        binding.spinnerFruits.onItemSelectedListener = this
    }
    override fun onNothingSelected(parent: AdapterView<*>?) {
    }
    override fun onItemSelected(parent: AdapterView<*>?,
        view: View?, position: Int, id: Long) {
        Toast.makeText(this, parent?.getItemAtPosition(
            position).toString(), Toast.LENGTH_LONG).show()
    }
}
```

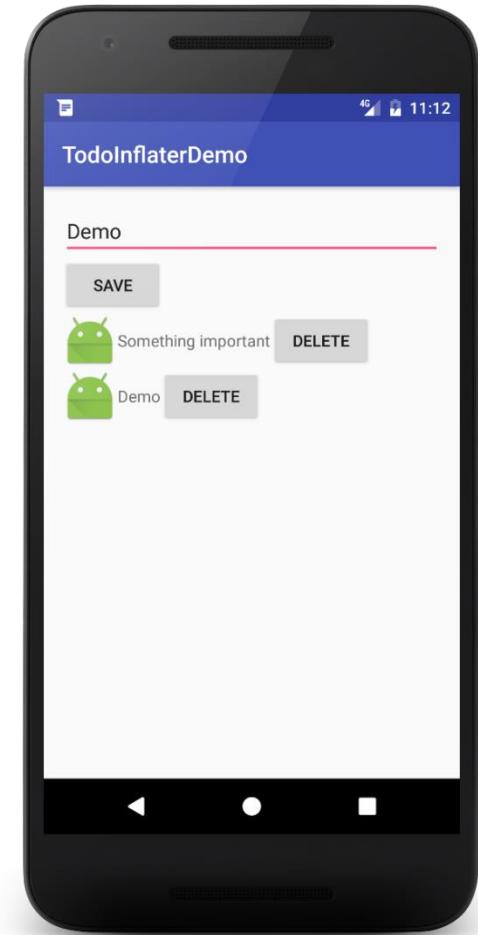
Dynamic UI management

- Inflaters
 - > Allows to create Views from XML code
- Android allows to create and modify the UI dynamically from Kotlin/Java Code
- E.g.: Inflating a *view* and adding it to a *LinearLayout* called *container*.

```
val myView = getLayoutInflater().inflate(  
    R.layout.activity_main, null)
```

Challenge

- Let's create a simple Todo application!
- Implement the delete button
- Add ScrollView
- Apply scrolling only for the list and do not scroll the header



THREAD, TIMER, TIMERTASK

Scheduling operations

- *Thread*
- *TimerTask* describes the task that should be executed on timer tick
- *Timer* object is used for scheduling *TimerTasks*
- *TimerTask*'s *run()* function runs on a separate thread
- User interface can be modified only for the main thread therefore we have to join back to the UI thread:
 - > Activity's *runOnUIThread(Runnable runnable)*
- **Do not forget** to cancel the *Timer* when it is not needed

Timer, TimerTask example

```
private inner class RunTimeTimerTask : TimerTask() {  
    override fun run() {  
        runOnUiThread {  
            val elapsedTime = (System.currentTimeMillis() -  
                startTime.getTime()) as Int / 1000  
            tvRunTime.setText("" + elapsedTime)  
        }  
    }  
}
```

```
var timerStopWatch = Timer()  
  
timerStopWatch.schedule(RunTimeTimerTask(), 0, 1000)  
  
// for example cancel Timer in the onPause() function:  
override fun onPause() {  
    super.onPause()  
    if (timerStopWatch != null) {  
        timerStopWatch.cancel()  
    }  
}
```

Summary

- Resource qualifiers
- Layout
 - > LinearLayout
 - Weight, Padding/margin, Gravity
 - > ScrollView, RelativeLayout, ConstraintLayout
- View
 - > ImageView, AutoCompleteTextView, Radiobutton, Spinner
- Dynamic UI management
- Thread, Timer, TimerTask

Codelabs

- Use **ConstraintLayout** to design your Android views
 - > <https://codelabs.developers.google.com/codelabs/constraint-layout/index.html?index=..%2F..io2018#0>
- Clickable images
 - > <https://codelabs.developers.google.com/codelabs/android-training-clickable-images/index.html?index=..%2F..android-training#0>
- Adaptive layouts:
 - > <https://codelabs.developers.google.com/codelabs/android-training-adaptive-layouts/index.html?index=..%2F..android-training#0>



Homework – Readings

- UI Overview:
 - > <https://developer.android.com/guide/topics/ui/overview.html>
- Layouts:
 - > <https://developer.android.com/guide/topics/ui/declaring-layout.html>
- LinearLayout:
 - > <https://developer.android.com/guide/topics/ui/layout/linear.html>



Homework – Readings

- ConstraintLayout:
 - > <https://developer.android.com/training/constraint-layout/>
- ScrollView:
 - > <https://guides.codepath.com/android/Working-with-the-ScrollView>
- Input Controls:
 - > <https://developer.android.com/guide/topics/ui/controls.html>
- Input Events:
 - > <https://developer.android.com/guide/topics/ui/ui-events.html>
- Custom Components:
 - > <https://developer.android.com/guide/topics/ui/custom-components.html>



Homework – readings

- Optimizing Layout Hierarchies
 - > <https://developer.android.com/training/improving-layouts/optimizing-layout.html>
- Re-using Layouts with <include/>
 - > <https://developer.android.com/training/improving-layouts/reusing-layouts.html>
- Delayed loading of views
 - > <https://developer.android.com/training/improving-layouts/loading-ondemand.html>



Questions?



Mobile- and Web-based Software

Lecture 6: Persistance

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

Data storage intro

Many kinds of applications needs to store some data

- > Almost all apps have **settings**
- > Camera applications: **save photos**
- > Apps that communicate with a server: **local cache**
- > Email apps: **indexed database of mails**
- > Apps that can be rotated (port/land): **state of the UI**
- > Dropbox, Picasa: **primary storage in a cloud but subset of files available on the device**

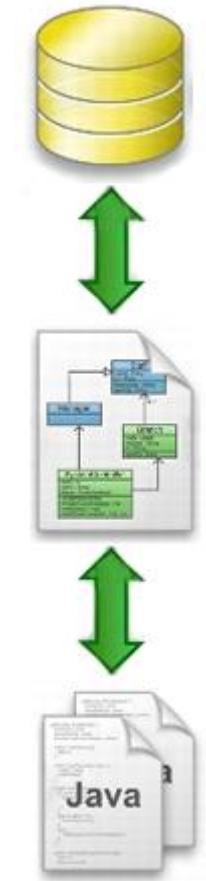
Data storage options

- SQLite DB: structured data in a relational DB
- SharedPreferences: primitive types in key-value pairs
- Saving state of the activity
- File management:
 - > Private storage: non-public data in filesystem
 - > Public storage: user-accessible data in filesystem
- Network: data stored in a cloud or own webserver

OBJECT RELATION MAPPING (ORM)

What is Object Relation Mapping?

- Persisting Java objects to SQL databases
- Basic principles:
 - > Class name ->
 - Table name
 - > Object ->
 - Table row
 - > Field ->
 - Table column, cell
 - > Etc.



Object Relation Mapping libraries on Android

- Room – Google Official
 - > <https://developer.android.com/topic/libraries/architecture/room.html>
- Realm:
 - > <http://realm.io/>
- Sugar-ORM
 - > <http://satyan.github.io/sugar/index.html>
- ORMLite
 - > <http://ormlite.com/>
- GreenDAO
 - > <http://greendao-orm.com/>
- MapDB:
 - > <http://www.mapdb.org/>

ROOM

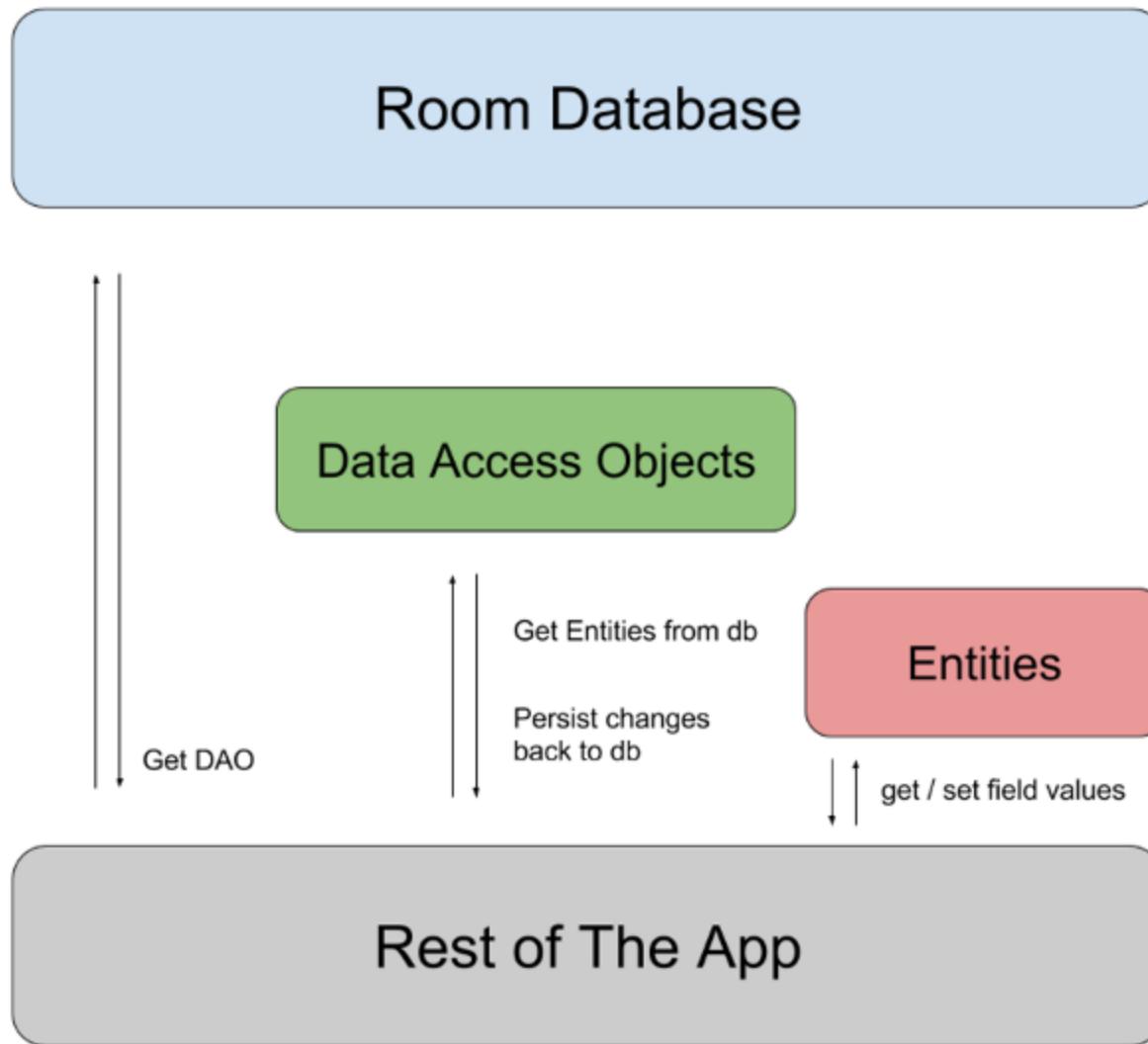
Room

- Persistent Data Storage Library with ORM
- Abstraction layer above SQLite
- Quick database access with full power of SQLite
- Every Database operation should be in a separate *Thread*
- Usage:

```
plugins{ ...
    id("kotlin-kapt")}

...
implementation("androidx.room:room-runtime:2.5.2")
annotationProcessor("androidx.room:room-compiler:2.5.2")
kapt("androidx.room:room-compiler:2.5.2")
```

Room components



Room example - Entity

```
@Entity(tableName = "grade")
data class Grade(
    @PrimaryKey(autoGenerate = true) var gradeId: Long?,
    @ColumnInfo(name = "studentid") var studentId: String,
    @ColumnInfo(name = "grade") var grade: String
)
```

Room example - DAO

```
@Dao
interface GradeDAO {
    @Query("""SELECT * FROM grade WHERE grade="B""""")
    fun getBGrades(): List<Grade>

    @Query("SELECT * FROM grade")
    fun getAllGrades(): List<Grade>

    @Query("SELECT * FROM grade WHERE grade = :grade")
    fun getSpecificGrades(grade: String): List<Grade>

    @Insert
    fun insertGrades(vararg grades: Grade)

    @Delete
    fun deleteGrade(grade: Grade)
}
```

RoomDatabase

```
@Database(entities = arrayOf(Grade::class), version = 1)
abstract class AppDatabase : RoomDatabase() {

    abstract fun gradeDao(): GradeDAO

    companion object {
        private var INSTANCE: AppDatabase? = null

        fun getInstance(context: Context): AppDatabase {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.applicationContext,
                    AppDatabase::class.java, "grade.db").build()
            }
            return INSTANCE!!
        }

        fun destroyInstance() {
            INSTANCE = null
        }
    }
}
```

Room usage 1/2

- Database:

```
@Database(entities = {User.class}, version = 1)  
public abstract class AppDatabase extends RoomDatabase {  
    public abstract UserDao userDao();  
}
```

- Database instance:

```
AppDatabase db = Room.databaseBuilder(  
        getApplicationContext(),  
        AppDatabase.class, "database-name").build();
```

Hint: use the RoomDatabase object as a Singleton, as it is a „heavy” object.

Room usage 2/2

- Insert

```
val grade = Grade(null, etStudentId.text.toString(),
    etGrade.text.toString())

val dbThread = Thread {
    AppDatabase.getInstance(this@MainActivity).gradeDao().insertGrades(grade)
}
dbThread.start()
```

- Query

```
val dbThread = Thread {
    val grades = AppDatabase.getInstance(this@MainActivity).gradeDao()
        .getSpecificGrades("A+")
    runOnUiThread {
        binding.tvResult.text = ""
        grades.forEach {
            binding.tvResult.append("${it.studentId} ${it.grade}\n")
        }
    }
}
dbThread.start()
```

Delete all feature - ShoppingList

- DAO:

```
@Query("DELETE FROM [YOUR_TABLE_NAME_HERE]")
fun deleteAll()
```

- Adapter:

```
fun removeAll() {
    todoItems.clear()
    notifyDataSetChanged()
}
```

- Use from Activity:

```
binding.btnDeleteAll.setOnClickListener{
    Thread {
        AppDatabase.getInstance(this@ScrollingActivity).todoDao().deleteAll()
        runOnUiThread {
            todoAdapter.removeAll()
        }
    }.start()
}
```

Enable default migration policy in DB

- Recreates tables with empty data.
- Add this in *AppDatabase* class:

```
...
fun getInstance(context: Context): AppDatabase {
    if (INSTANCE == null) {
        INSTANCE =
            Room.databaseBuilder(context.getApplicationContext(),
                AppDatabase::class.java, "todo.db")
                .fallbackToDestructiveMigration()
                .build()
    }
    return INSTANCE!!
}
```

Agenda

- ORM
- Room
- Shared Preferences
- File
- SQLite

PERSISTENCE STORAGE

Shared Preferences

SharedPrefs

- For saving basic types as key-value pairs
 - > Like a *Dictionary* in .NET
- Physically stored in a file, but OS hides it
- There can be more than one for each application
 - > Distincted by their names
 - > `getSharedPreferences(name, visibility)`
 - > If not found, system creates it (with the given visibility)

SharedPreferences

- Ideal for saving data that can be represented using **primitive types**, e.g.:
 - Default values
 - Username/password for auto-login feature
 - Any value that must be reached from multiple Activities
 - Preferences (this is why it's called *SharedPreferences*)
- Preserve its content on app and phone restarts (why?)
 - It is stored as a File

SharedPreferences types

- Primitive types:
 - int
 - long
 - float
 - String
 - boolean
- This is not a *Bundle*, it cannot store arbitrary types
 - > Even if they implement *Serializable* or *Parcelable*
- How can we still force our custom objects to fit in SP?
 - > Serialize manually to String (JSON, XML, ...)

SharedPreferences write

- An **Editor** object is required for writing:

```
val PREF_NAME: String = "MySettings"  
val sp: SharedPreferences =  
    getSharedPreferences(PREF_NAME, MODE_PRIVATE)  
val editor: Editor = sp.edit() ← Open for edit  
editor.putLong("lastSyncTimestamp",  
    Calendar.getInstance().getTimeInMillis())  
editor.putBoolean("KEY_FIRST", false)  
editor.apply()
```

Type of
the
data

Save changes

Key

Value

Private mode

Persistent file name

SharedPreferences reading

- Read directly from SharedPreferences
- Key name and type

```
PREF_NAME = "MySettings"  
val sp =  
    getSharedPreferences(PREF_NAME, MODE_PRIVATE)  
val lastSaved: Long = sp.getLong("lastSaved", 0)  
val isFirstRun: Boolean =  
    sp.getBoolean("KEY_FIRST", true)
```

The code demonstrates reading values from a SharedPreferences instance named 'MySettings'. It retrieves a long value for 'lastSaved' and a boolean value for 'isFirstRun'. Annotations highlight the key name ('lastSaved'), its type ('Long'), and its default value ('0').

- Useful method to list all key values:
 - > **sp.getAll()** – list all key values in a Map

Let's practice!

- Create an application that stores the first time when it was started!

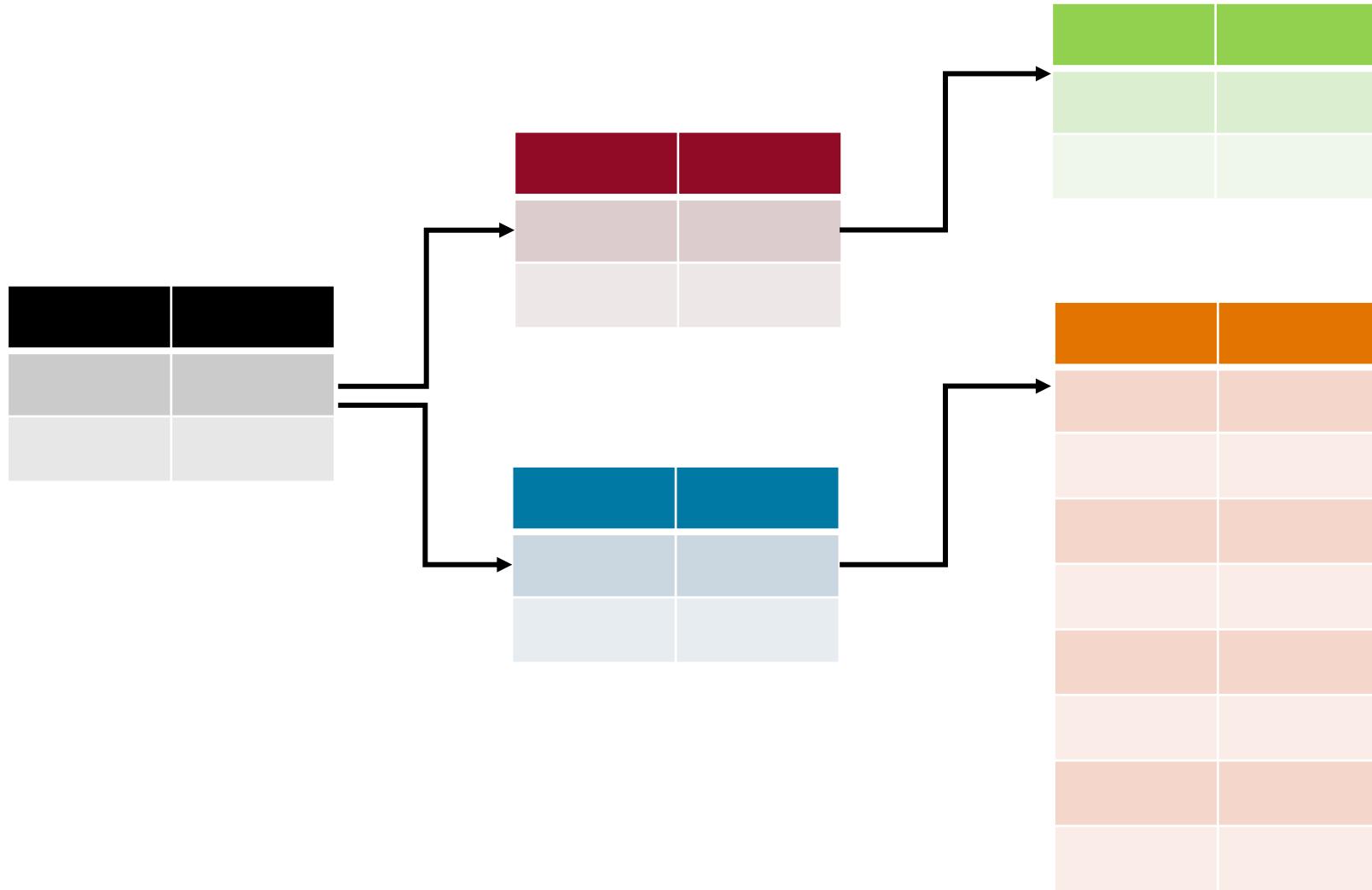
File management

File management on Android

- Standard Java API
- Special storage options
 - > **Internal storage**: protected storage, only the owner application can access to it
 - Cache storage
 - > **External storage**: general storage on the phone, fully accessible (~external storage/SD card)
- WRITE_EXTERNAL_STORAGE permission is required
- Runtime permissions later

RELATIONAL DATABASE - SQLITE

Relation database



SQLite

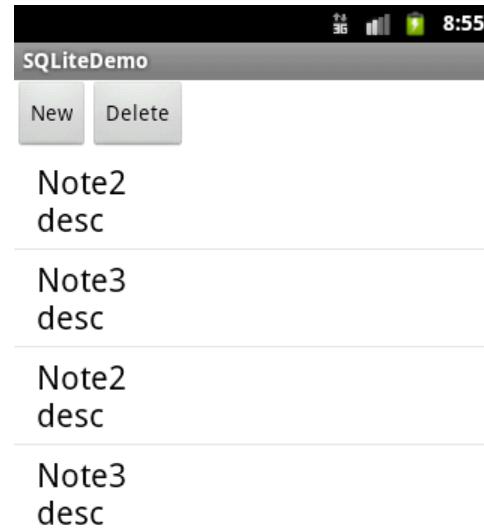
- Very similar to MySQL but there are some limitations
- Features:
 - > SQL syntax
 - > Transaction management
 - > Support for PreparedStatements

SQLITE EXAMPLE (JAVA)

Example

- Create an application where we can store Notes.
- Note fields:
 - > _id: long
 - > title: String
 - > desc: String
- Make it possible to create new note and delete notes!

SQLite example – Note - Java



Note POJO (Plain Old Java Object)

```
public class Note {  
  
    private long id;  
  
    private String title;  
  
    private String desc;  
  
    // Getters and setters  
    // ...  
  
    @Override  
  
    public String toString() {  
  
        return title+"\n"+desc;  
    }  
}
```

Note table in a separate tab

```
public class NoteTable {  
  
    public static final String TABLE_NOTE = "note";  
  
    public static final String COLUMN_ID = "_id";  
  
    public static final String COLUMN_TITLE = "title";  
  
    public static final String COLUMN_DESC = "desc";  
  
    private static final String DATABASE_CREATE = "create table " +  
TABLE_NOTE  
        + "(" + COLUMN_ID + " integer primary key autoincrement, "  
        + COLUMN_TITLE + " text not null, " + COLUMN_DESC  
        + " text not null" + ");";  
  
    public static void onCreate(SQLiteDatabase database) {  
        database.execSQL(DATABASE_CREATE);  
    }  
}
```

SQLite supports
AutoIncrement

Create new table for
simplicity

Own SQLiteOpenHelper

```
public class NoteDBHelper extends SQLiteOpenHelper {  
    private static final String DATABASE_NAME = "notetable.db";  
    private static final int DATABASE_VERSION = 1;  
  
    public NoteDBHelper(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase database) {  
        NoteTable.onCreate(database);  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase database, int oldVersion,  
        int newVersion) {  
        NoteTable.onUpgrade(database, oldVersion, newVersion);  
    }  
}
```

Database file name and version

Table creation

Table refresh, when the version changes

DAO class for reaching the data 1/3

```
public class NoteDataSource {  
    // Database fields  
    private SQLiteDatabase database;  
    private NoteDBHelper dbHelper;  
    private String[] allColumns = { NoteTable.COLUMN_ID,  
        NoteTable.COLUMN_TITLE, NoteTable.COLUMN_DESC };
```

```
public NoteDataSource(Context context) {  
    dbHelper = new NoteDBHelper(context);  
}
```

SQLiteDatabase
instance

```
public void open() throws SQLException {  
    database = dbHelper.getWritableDatabase();  
}
```

Database open
and close

```
public void close() {  
    dbHelper.close();  
}
```

DAO class for reaching the data 2/3

```
public Note createNote(String aTitle, String aDesc) {  
    ContentValues values = new ContentValues();  
    values.put(NoteTable.COLUMN_TITLE, aTitle);  
    values.put(NoteTable.COLUMN_DESC, aDesc);  
    long insertId = database.insert(NoteTable.TABLE_NOTE, null,  
        values);  
    Cursor cursor = database.query(NoteTable.TABLE_NOTE,  
        allColumns, NoteTable.COLUMN_ID + " = " + insertId, null,  
        null, null, null);  
    cursor.moveToFirst();  
    Note newNote = cursorToNote(cursor);  
    cursor.close();  
    return newNote;  
}  
  
private Note cursorToNote(Cursor cursor) {  
    Note note = new Note();  
    note.setId(cursor.getLong(0));  
    note.setTitle(cursor.getString(1));  
    note.setDesc(cursor.getString(2));  
    return note;  
}
```

ContentValues usage
for inserting

DAO class for reaching the data 3/3

```
public void deleteNote(Note note) {  
    long id = note.getId();  
    database.delete(NoteTable.TABLE_NOTE, NoteTable.COLUMN_ID  
        + " = " + id, null);  
}  
  
public List<Note> getAllNotes() {  
    List<Note> notes = new ArrayList<Note>();  
    Cursor cursor = database.query(NoteTable.TABLE_NOTE,  
        allColumns, null, null, null, null, null);  
    cursor.moveToFirst();  
    while (!cursor.isAfterLast()) {  
        Note note = cursorToNote(cursor);  
        notes.add(note);  
        cursor.moveToNext();  
    }  
    // Do not forget to close the Cursor!  
    cursor.close();  
    return notes;  
}  
}
```

Iteration via a Cursor

ListActivity 1/2

```
public class MainActivity extends ListActivity {  
    private NoteDataSource datasource;  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        datasource = new NoteDataSource(this);  
        datasource.open(); ← Open Database  
        List<Note> values = datasource.getAllNotes();  
        // SimpleCursorAdapter usage for simple displaying the elements  
        ArrayAdapter<Note> adapter = new ArrayAdapter<Note>(this,  
            android.R.layout.simple_selectable_list_item, values);  
        setListAdapter(adapter); ← Load list  
    }  
    @Override  
    protected void onResume() {  
        datasource.open();  
        super.onResume();  
    }  
    @Override  
    protected void onPause() {  
        datasource.close(); ← Close database connection  
        super.onPause();  
    }  
}
```

ListActivity 2/2

```
// Create event handler form XML
public void onClick(View view) {
    @SuppressWarnings("unchecked")
    ArrayAdapter<Note> adapter = (ArrayAdapter<Note>) getListAdapter();
    Note note = null;
    switch (view.getId()) {
        case R.id.newnote:
            String[] notes = new String[] { "Note1", "Note2", "Note3" };
            int nextInt = new Random().nextInt(3);
            note = datasource.createNote(notes[nextInt], "desc");
            adapter.add(note);
            break;
        case R.id.delnote:
            if (getListAdapter().getCount() > 0) {
                note = (Note) getListAdapter().getItem(0);
                datasource.deleteNote(note);
                adapter.remove(note);
            }
            break;
    }
    adapter.notifyDataSetChanged();
}
```

Notify list
about the
change

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/group"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <Button
            android:id="@+id/newnote"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New"
            android:onClick="onClick"/>
        <Button
            android:id="@+id/delnote"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Delete one"
            android:onClick="onClick"/>
    </LinearLayout>
    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="empty" />
</LinearLayout>
```

Event handler
from XML

ListActivity identifies the
ListView by this special ID

Homework – readings

- Storage options
 - > <https://developer.android.com/guide/topics/data/data-storage.html>
- SharedPreferences:
 - > <https://developer.android.com/training/data-storage/shared-preferences>
 - > <https://developer.android.com/guide/topics/ui/settings/>



Homework – readings

- Room
 - > <https://developer.android.com/topic/libraries/architecture/room.html>
 - > <https://developer.android.com/training/data-storage/room/index.html>



Questions?



Mobile- and Web-based Software

Lecture 7: HTTP, Retrofit

David Sik

david.sik@aut.bme.hu



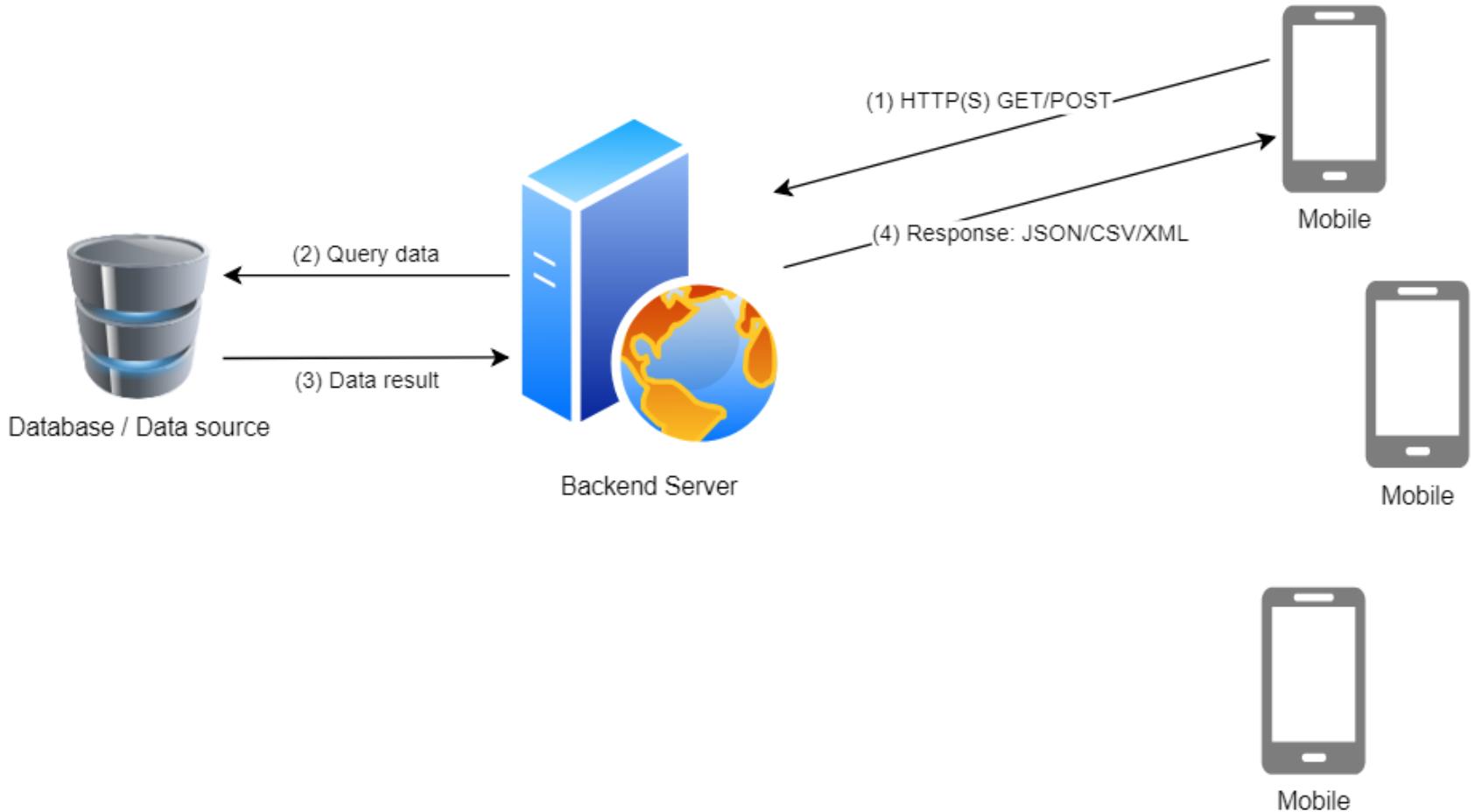
Department of
Automation and
Applied Informatics

Communication types

- Long-range communication and Internet protocols
 - > Cloud
 - > HTTP (REST Services - Representational State Transfer)
 - > TCP Socket
 - > UDP
- Short-range communication
 - > WiFi Direct
 - > Bluetooth
 - > NFC

HTTP COMMUNICATION

Typical Architecture



HTTP communication on Android

- One of the most commonly used technologies
- HTTP methods
 - GET, POST, PUT, DELETE
- Full HTTPS support and certificate management
- REST communications (Representational State Transfer)

Example HTTP Query

- Query:
 - > `http://api.openweathermap.org/data/2.5/weather?q=Budapest&units=metric&appid=f3d694bc3e1d44c1ed5a97bd1120e8fe`
- Query parts:
 - > Host:
 - `http://api.openweathermap.org/`
 - > Path:
 - `data/2.5/weather`
 - > Query parameters:
 - `?q=Budapest,hu&units=metric&appid=f3d694bc3e1d44c1ed5a97bd1120e8fe`

HTTP(S) queries on Android

- Standard Java API
 - > *HTTPURLConnection, InputStream, AsyncTask*
 - > ... we check only a demo for this
- External library
 - > *Retrofit*
 - > ... we use this in HW ☺

HTTP GET – HttpURLConnection

```
fun httpGet(urlAddr: String) {
    var reader: BufferedReader? = null
    try {
        val url = URL("http://myserver.com/api/getitems")
        val conn = url.openConnection() as HttpURLConnection
        reader = BufferedReader(InputStreamReader(conn.getInputStream()))
        var line: String?
        do {
            line = reader.readLine()
            System.out.println(line)
        } while (line != null)
    } catch (e: IOException) {
        e.printStackTrace()
    } finally {
        if (reader != null) {
            try {
                reader.close()
            } catch (e: IOException) {
                e.printStackTrace()
            }
        }
    }
}
```

HTTP POST- HttpURLConnection

```
fun httpPost(urlAddr: String, content: ByteArray) {
    // ...
    var os: OutputStream? = null
    try {
        val url = URL("http://mysrver.com/api/refreshitems")
        val conn = url.openConnection() as HttpURLConnection
        conn.requestMethod = "POST"
        conn.doOutput = true
        conn.useCaches = false
        os = conn.getOutputStream()
        os.write(content)
        os.flush()
        // ...
    } catch (e: IOException) {
        e.printStackTrace()
    } finally {
        // ...
        if (os != null) {
            try {
                os.close()
            } catch (e: IOException) {
                e.printStackTrace()
            }
        }
    }
}
```

Timeout values

- Timeout for opening the connection
- Timeout for reading the result
- Example:

...

```
val conn = url.openConnection() as HttpURLConnection
```

...

```
conn.setConnectTimeout(10000)
```

```
conn.setReadTimeout(10000)
```

...

Headers and Cookies

- Header:

```
val conn = url.openConnection() as HttpURLConnection  
conn.setRequestProperty("[KEY]", "[VALUE]")
```

- Cookie:

```
val conn = url.openConnection() as HttpURLConnection  
conn.setRequestProperty("Cookie", "sessionid=abc; age=15")
```

- Header and Cookie:

```
val conn = url.openConnection() as HttpURLConnection  
conn.setRequestProperty("Content-Type", "application/json")  
conn.setRequestProperty("Cookie", "sessionid=abc; age=15")
```

URL Encoding 1/2

- URL GET parameters can not contain “extra” characters
- Those characters require URL encoding
- Example:
 - > `http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John Doe`
 - > Vs.
 - > `http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John%20Doe`
- The error in case of “space” character:
 - > 11-26 18:39:24.417: ERROR/AndroidRuntime(17232):
java.lang.IllegalArgumentException: Illegal character in query at index 53:
`http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John Doe`
- Solution:
 - > `val name = URLEncoder.encode("John Doe")`
 - > `connect("http://avalon.aut.bme.hu/~tyrael/phpget.php?name="+name)`

URL Encoding 2/2

- *URLEncoder*:
 - > Besides ('a'..'z', 'A'..'Z'), numbers ('0'..'9') and '.', '-', '*', '_' characters, all others requires URL encoding (they will be converted to their hexadecimal values), e.g.: '#' -> %23
 - > `encode(String s)`
 - > `encode(String s, String charsetName)`
- *URLDecoder*
 - > *Decoding URL encoded content* (MIME type: *application/x-www-form-urlencoded*)
 - > `decode(String s)`
 - > `decode(String s, String encoding)`

How to modify the UI from separate thread?

- Application runs on a main thread
- Long operations can block the User Interface
- Android allows to modify the UI only from the main thread
- Network communication runs on a separate thread, therefore somehow we have to return on the main thread:
 - > `Activity.runOnUiThread(Runnable)`
 - > `View.post(Runnable)`
 - > `View.postDelayed(Runnable, long)`
 - > `Handler`
 - > `AsyncTask` and `LocalBroadcast`
 - > External libs: `EventBus`, `Otto`
 - > **Best REST library:** `Retrofit`

AsyncTask example

```
class AsyncTaskUploadVote : AsyncTask<String, Void, String>() {  
  
    override fun onPreExecute() {  
        // This runs on the UI thread  
    }  
  
    override fun doInBackground(vararg params: String): String? {  
        // This runs on the Background Thread  
        val result = null  
        // save network response in result  
        return result  
    }  
  
    override fun onPostExecute(result: String?) {  
        // This runs on the UI Thread  
        // use the result on the UI Thread  
        Log.d("RESULT", result)  
    }  
  
}  
  
// usage e.g. from an Activity:  
AsyncTaskUploadVote("Yes").execute()
```

Sending data and processing responses

- Communication is done via a specific protocol with the server
- Messages are in well-known formats
- Typical formats:
 - > CSV (Comma Separated Value(s))
 - > XML (Extensible Markup Language)
 - > JSON (JavaScript Object Notation)
- Security is very important
- In some cases unique protocol can also be used

JSON

- Structural characters: '{', '}', '[', ']', '.', ','
- Example:

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "age": 25,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": 10021  
    },  
    "phoneNumber": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "fax",  
            "number": "646 555-4567"  
        }  
    ]  
}
```

Processing JSON

- **JSONObject**
 - > Parsing JSON objects
 - > Get elements with key
 - `getString(String name)`
 - `getJSONObject(String name)`
 - `getJSONArray(String name)`
 - > Create JSON objects from String or Map
- **JSONArray**
 - > For JSON arrays
 - > Parse and query by index
 - > Create from Collection

JSON/XML API examples

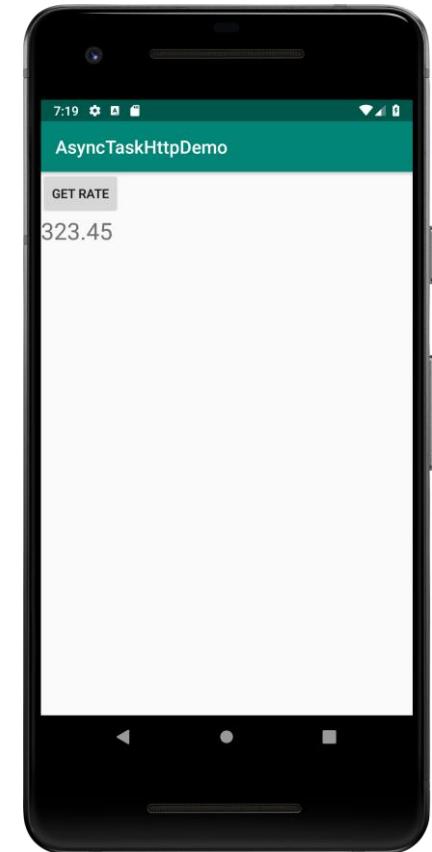
- Currency Exchange:
 - > <https://api.frankfurter.app/latest?from=USD>
- OpenWeather
 - > <http://api.openweathermap.org/data/2.5/weather?q=Budapest,hu&units=metric&appid=f3d694bc3e1d44c1ed5a97bd1120e8fe>
- TV Show Data API:
 - > <http://api.tvmaze.com/search/shows?q=stargate>
- Making JSON more readable:
 - > <https://jsonformatter.curiousconcept.com/>

Demo: Money Converter API

- Money exchange rate converter application

- > <https://api.frankfurter.app/latest?from=USD>

- > Technology:
 - AsyncTask
 - HttpURLConnection



JSON POJO example

JSON: Generates objects from JSON string based on class tempates

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "age": 25,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": 10021  
    },  
    "phoneNumber": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "fax",  
            "number": "646 555-4567"  
        }  
    ]  
}
```



```
data class Person(  
    val firstName: String?,  
    val lastName: String?,  
    val age: Number?,  
    val address: Address?,  
    val phoneNumber: List<PhoneNumber>?)  
  
data class Address(  
    val streetAddress: String?,  
    val city: String?,  
    val state: String?,  
    val postalCode: Number?)  
  
data class PhoneNumber(  
    val type: String?,  
    val number: String?)
```

MoneyConverter - GSON POJO

```
{  
    "amount": 1,  
    "base": "USD",  
    "rates": {  
        ...  
        "HUF": 289.8969072165,  
        "EUR": 0.896458987  
    },  
    "date": "2019-05-04"  
}
```

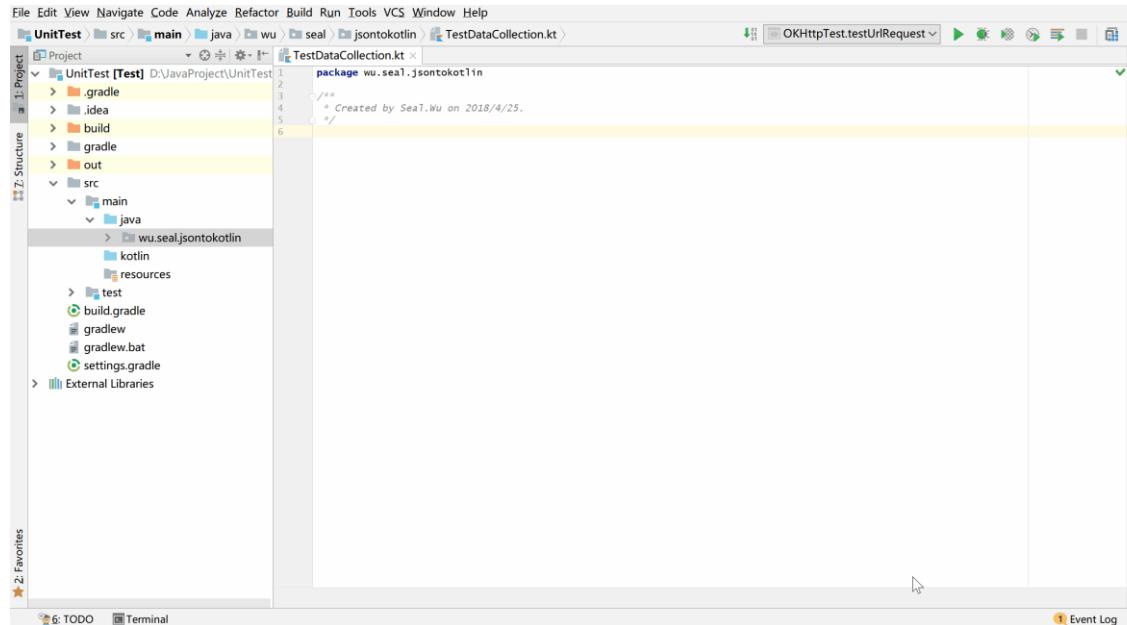


```
data class MoneyResult(  
    val date: String?,  
    val rates: Rates?,  
    val base: String?  
)
```

```
data class Rates(  
    ...  
    val HUF: Number?,  
    val EUR: Number?  
)
```

Entity or data class generation from JSON

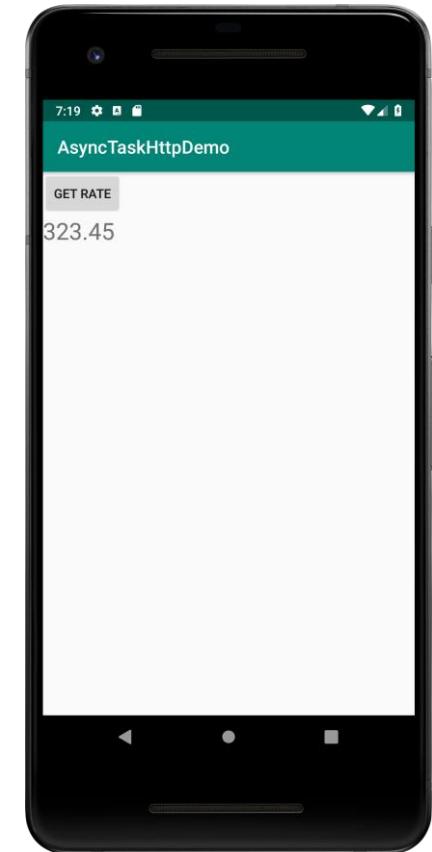
- <https://jsonformatter.org/json-to-kotlin>
- <https://transform.tools/json-to-kotlin>
- <https://json2kt.com/>
- <https://github.com/wuseal/JsonToKotlinClass>



Code -> Generate -> Convert Json into Kotlin Class

Demo: Money Converter API with GSON

- Money exchange rate converter application
 - > <https://api.frankfurter.app/latest?from=USD>
 - > Technology:
 - AsyncTask
 - HttpURLConnection
 - **GSON**



External Libs for XML and JSON parsing

- XML:
 - > TikXML
 - > <https://github.com/Tickaroo/tikxml>
- JSON:
 - > Gson
 - > <https://github.com/google/gson>
- REST API testing:
 - > Postman application
 - > <https://www.getpostman.com/>

Retrofit

- Best HTTP(s) library on Android
 - > Simple API definition and network calls
 - > GSON integration for data parsing

- Define HTTP API in an interface:

```
interface ItemsService {  
    @GET("/items/{item}/details")  
    fun listItems(@Path("item") item: String): Call<List<Item>>  
}
```

- Retrofit object and HTTP service object:

```
val retrofit = Retrofit.Builder()  
    .baseUrl("https://api.myshop.com")  
    .build()  
val service = retrofit.create(ItemsService::class.java)
```

- Call example:

```
val items: Call<List<Item>> = service.listItems("myItem")
```

Retrofit

- HTTP queries with annotations:
 - > URL and query parameters
 - > Body – object conversion (JSON to object, protocol buffers)
 - > Multipart request and file upload is also supported
- Gradle:
 - > Implementation("com.squareup.retrofit2:retrofit:2.9.0")
- More details:
 - > <http://square.github.io/retrofit/>

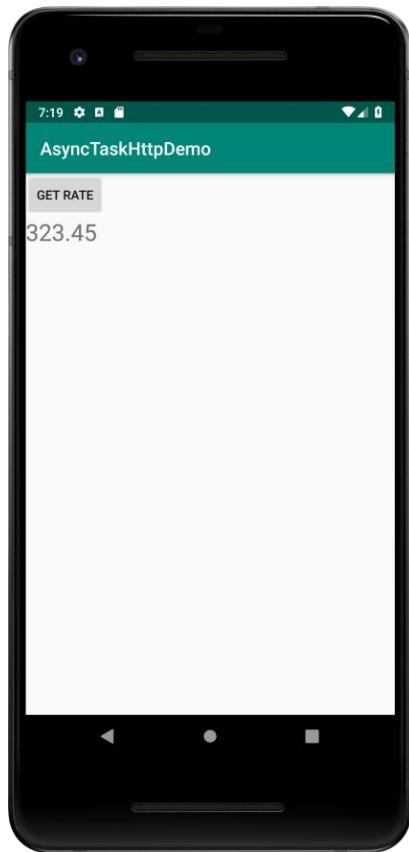
Retrofit – JSON support

- Auto conversion in the background (JSON – Object)
 - > Conversion algorithm can be customized
 - >

```
val retrofit=Retrofit.Builder()
    .baseUrl("http://api.myserver.com/")
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```
- Gradle:
 - > Implementation("com.google.code.gson:gson:2.10.1")
 - > Implementation("com.squareup.retrofit2:converter-gson:2.9.0")
- More details:
 - > <http://square.github.io/retrofit/>

Example - Retrofit

- <https://api.frankfurter.app/latest?from=USD>
- Retrofit 2 + GSON



Retrofit - Entity / data class

Use the generator:

<https://api.frankfurter.app/latest?from=USD>



<https://jsonformatter.org/json-to-kotlin>

<https://transform.tools/json-to-kotlin>

<https://json2kt.com/>

```
data class MoneyResult(  
    var date: String,  
    var rates: Rates,  
    var base: String  
)
```

```
data class Rates(  
    var BGN: Double,  
    var CAD: Double,  
    ...  
)
```

Retrofit – API interface

```
import retrofit2.Call
import retrofit2.Callback
import retrofit2.http.GET
import retrofit2.http.Query

interface MoneyAPI {
    @GET("/latest")
    fun getRates(@Query("from") base: String): Call<MoneyResult>
}
```

Retrofit – usage from Activity

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

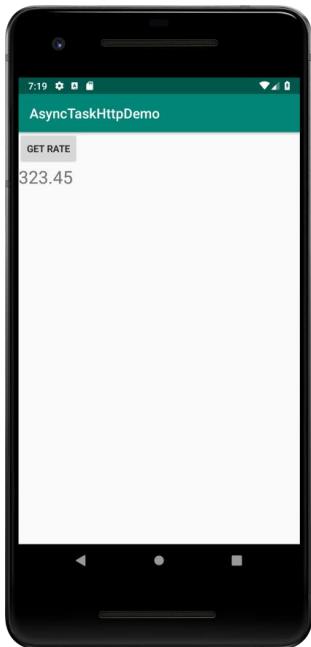
    val retrofit = Retrofit.Builder()
        .baseUrl("https://api.frankfurter.app/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val currencyAPI = retrofit.create(MoneyAPI::class.java)

    binding.btnGetRate.setOnClickListener {
        val ratesCall = currencyAPI.getRates("EUR")
        ratesCall.enqueue(object: Callback<MoneyResult> {
            override fun onFailure(call: Call<MoneyResult>, t: Throwable) {
                binding.tvResult.text = t.message
            }
            override fun onResponse(call: Call<MoneyResult>,
                                  response: Response<MoneyResult>) {
                binding.tvResult.text = response.body()?.rates?.HUF?.toString()
            }
        })
    }
}
```

Engagement

- Let's implement a demo HTTP Query app with *Retrofit* to query currency rates:
 - > <https://api.frankfurter.app/latest?from=USD>



URL analysis

- <https://api.frankfurter.app/latest?from=USD>
 - > Must be https
 - > Host URL: <https://api.frankfurter.app/>
 - > Path: latest
 - > Query params:
 - from: String

REST API collections

- <https://github.com/toddmotto/public-apis>
- <https://github.com/abhishekbanthia/Public-APIs>
- <https://github.com/Kikobeats/awesome-api>

Summary

- Network communication
- HTTP communication
 - > Typical architecture
- Standard HTTP API
- Typical data formats
 - > JSON
 - > CSV
 - > XML
- Using HTTP with Retrofit
- Example APIs

Homework – readings

- Using Retrofit 2.x as REST client
 - > <http://square.github.io/retrofit/>
 - > <http://www.vogella.com/tutorials/Retrofit/article.html>



Questions?



Mobile- and Web-based Software

Lecture 8: List, RecyclerView

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

Agenda

- Efficient list handling
- RecyclerView

WORKING WITH LISTS

ListActivity – displaying lists – old API

- Contains already a *ListView* component that can be queried by the `getListView()` function
- Contains a *ListAdapter*, that is responsible to manage the store elements and display them
- Further objectives:
 - Add, delete and modify elements
 - By overriding the `getView(int position, View convertView, ViewGroup parent)` function the rendering of an element can be set
 - `convertView` is a previously rendered object (not defined which one) that can be re-used for rendering the other line, therefore we do not have to recreate the whole view again
- Later: *ListFragment*

ListAdapter example (Java)

```
public class MyContactsAdapter extends ArrayAdapter {
    private List myContacts;
    private Context context;

    public MyContactsAdapter(Context context, int textViewResourceId,
                           List contacts) {
        super(context, textViewResourceId, data);
        this.context = context;
        this.myContacts = contacts;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View v = convertView;
        if(v == null) {
            LayoutInflator inflater = LayoutInflator.from(context);
            v = inflater.inflate(R.layout.layout_list_item, null);
        }
        TextView txtName = (TextView)v.findViewById(R.id.txtName);
        TextView txtMail = (TextView)v.findViewById(R.id.txtMail);
        MyContact entry = myContacts.get(position);
        txtName.setText(entry.getName());
        txtMail.setText(entry.getMail());
        return v;
    }
}
```

getView(...) usage – simple case (Java)

```
@Override  
  
public View getView(int position, View convertView, ViewGroup parent) {  
  
    View v = convertView;  
  
    if(v == null) {  
  
        LayoutInflator inflater = LayoutInflator.from(mContext);  
  
        v = inflater.inflate(R.layout.layout_list_item, null);  
  
    }  
  
    TextView txtName = (TextView)v.findViewById(R.id.txtName);  
    TextView txtMail = (TextView)v.findViewById(R.id.txtMail);  
  
    Contact entry = mList.get(position);  
  
    txtName.setText(entry.getName());  
    txtMail.setText(entry.getMail());  
  
    return v;  
}
```

getView(...) with ViewHolder pattern (much faster!) (Java)

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View v = convertView;
    if(v == null) {
        LayoutInflator inflater = LayoutInflator.from(mContext);
        v = inflater.inflate(R.layout.layout_list_item, parent);
        ViewHolder holder = new ViewHolder();
        holder.txtName = (TextView)v.findViewById(R.id.txtName);
        holder.txtMail = (TextView)v.findViewById(R.id.txtMail);
        v.setTag(holder);
    }

    Contact entry = mList.get(position);
    if(entry != null) {
        ViewHolder holder = (ViewHolder)v.getTag();
        holder.txtName.setText(entry.getName());
        holder.txtMail.setText(entry.getMail());
    }
    return v;
}

static class ViewHolder {
    TextView txtName;
    TextView txtMail;
}
```

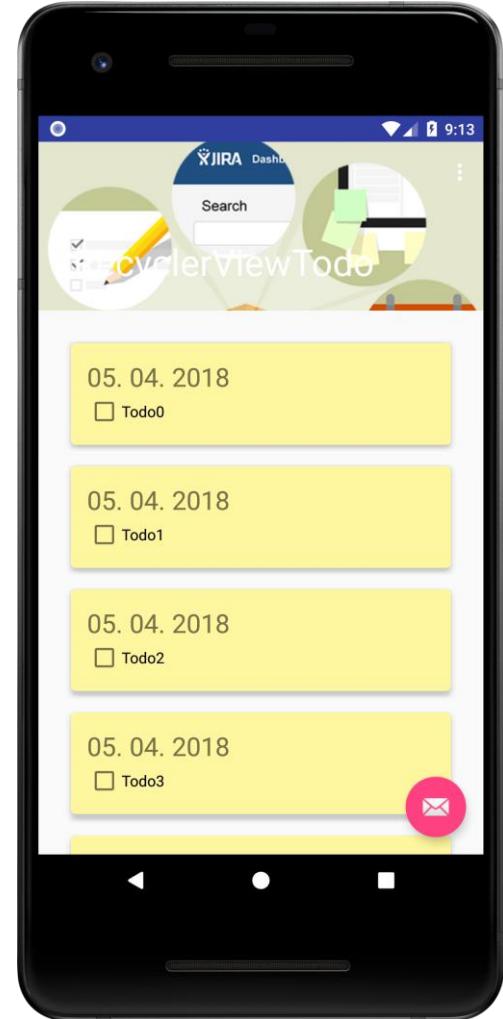
Advantages of ViewHolder pattern

- Static *ViewHolder* object, cache support
- No need for continuous *findViewById(...)* call
- Fast operation, smooth scrolling

RECYCLERVIEW

RecyclerView

- Advanced ListView
- Recommended to use instead of ListView
- Flexible list
- *ViewHolder* pattern by default for efficiency



RecyclerView.Adapter<ViewHolder> 1/3

- Initialization and constructor

```
private val context: Context
private val items: MutableList<ShoppingItem> = mutableListOf<ShoppingItem>(
    ShoppingItem("milk", 200, false),
    ShoppingItem("car", 4000, false),
    ShoppingItem("beer", 1, false)
)

constructor(context: Context) : super() {
    this.context = context
}
```

- onCreateViewHolder(...), define the layout of one row in the list

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {

    return ViewHolder(TodoRowBinding.inflate(LayoutInflater.from(parent.context),
        parent, false))
}
```

ViewHolder implementation

```
class class ViewHolder(val binding: TodoRowBinding) :  
    RecyclerView.ViewHolder(binding.root) {  
}
```

RecyclerView.Adapter<ViewHolder> 2/3

- Binding the ViewHolder
- Setting the values in one row
- Set event handlers for items on a row

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
    //val shopItem = items[holder.adapterPosition]  
    //holder.tvName = shopItem.name  
  
    val (name, price, bought) = items[holder.adapterPosition]  
    holder.tvName.text = name  
    holder.tvPrice.text = price.toString()  
    holder.cbBought.isChecked = bought  
  
    holder.btnDelete.setOnClickListener{  
        (context as MainActivity).showEditTodoDialog(items[holder.adapterPosition])  
    }  
}
```

RecyclerView.Adapter<ViewHolder> 3/3

- Number of items, adding, deleting functions

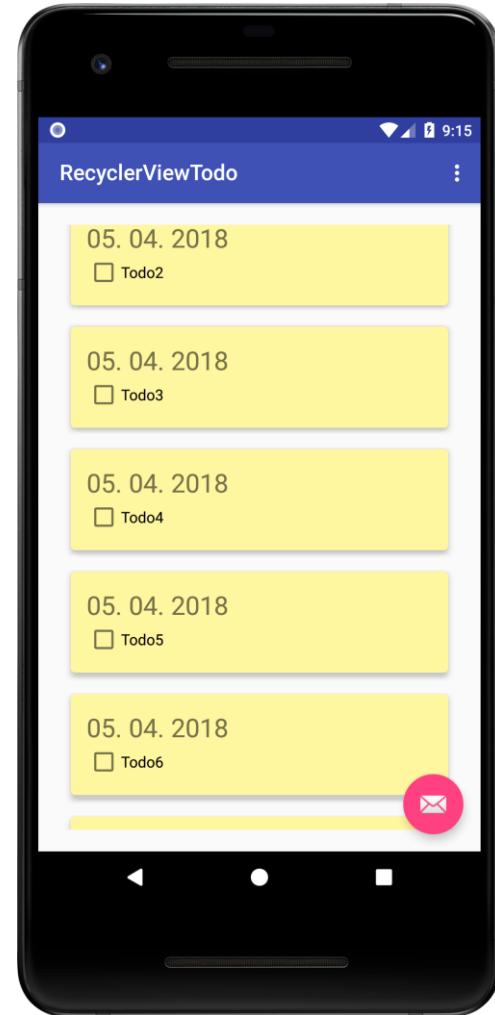
```
override fun getItemCount() = items.size

fun addItem(item: ShoppingItem) {
    items += item
    notifyItemInserted(items.lastIndex)
}

private fun deleteItemBasedOnPosition(position: Int) {
    items.removeAt(position)
    notifyItemRemoved(position)
}
```

Engagement!

- Create a Todo application with RecyclerView
- Show a checkbox before each todo elements
- Implement quick „undo“ action when a Todo is created



TOUCH GESTURES IN RECYCLERVIEW

Move and swipe gestures

- Items are removed by swipe gestures
- Items order can be changed by drag&drop
- *RecyclerView* support:
 - > `ItemTouchHelper.Callback`

ItemTouchHelper.Callback 1/2

- *isLongPressDragEnabled():*
 - > Returns true if drag&drop is enabled
- *isItemViewSwipeEnabled():*
 - > Returns true if swipe is enabled
- *onMove(...):*
 - > Called when move event detected
- *onSwipe(...):*
 - > Called when swipe is detected

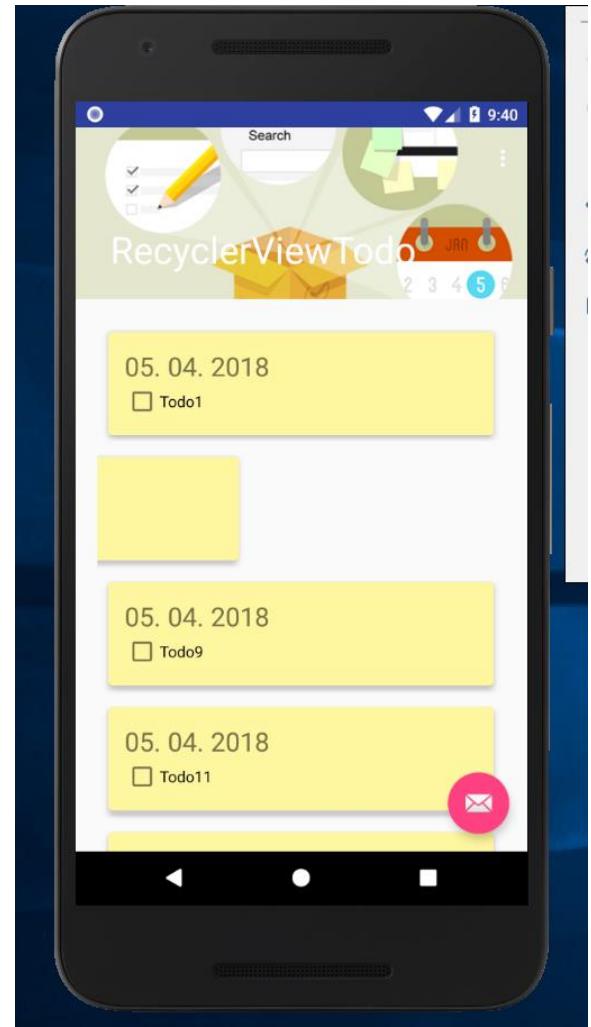
ItemTouchHelper.Callback 2/2

- Set drag and swipe allowed directions:

```
override fun getMovementFlags(recyclerView: RecyclerView,  
    viewHolder: RecyclerView.ViewHolder): Int {  
    val dragFlags = ItemTouchHelper.UP or ItemTouchHelper.DOWN  
    val swipeFlags = ItemTouchHelper.START or ItemTouchHelper.END  
    return ItemTouchHelper.Callback.makeMovementFlags(dragFlags, swipeFlags)  
}
```

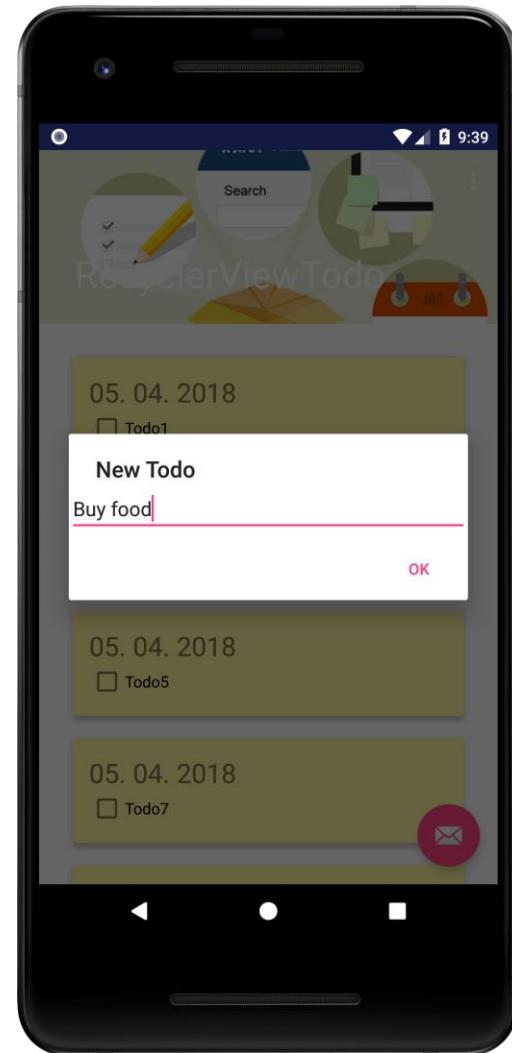
Engagement!

- Extend the application with swipe gesture for removing and reordering items!



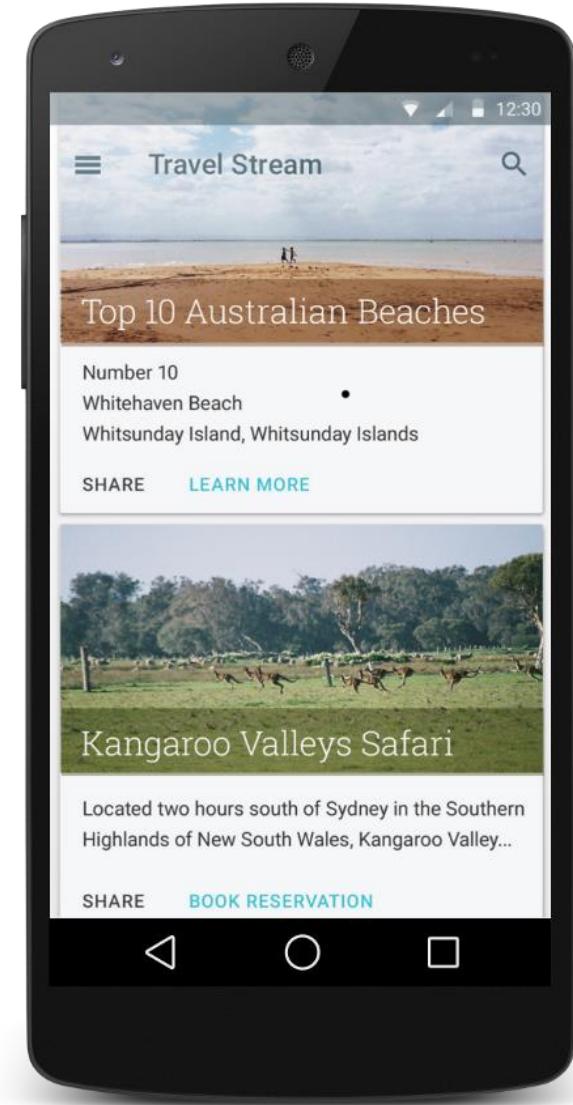
Use AlertDialog with DialogFragment – New Todo Dialog

```
val builder = AlertDialog.Builder(this@MainActivity)
builder.setTitle("Enter Todo")
val input = EditText(this@MainActivity)
builder.setView(input)
builder.setPositiveButton("OK", { dialog, which ->
    todoRecyclerAdapter.addTodo(
        Todo(input.text.toString(), false))
    binding.recyclerTodo.scrollToPosition(0)
})
builder.setNegativeButton("Cancel") {
    dialog, which -> dialog.cancel()
}
builder.show()
```



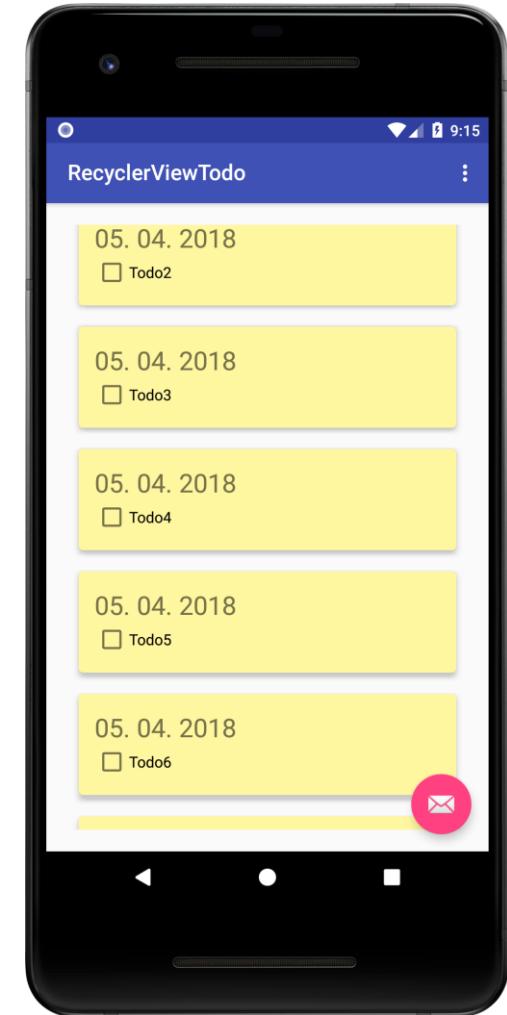
Introducing CardView

- Display data in similarly styled containers
- Often used in lists to hold each item's information
- CardView API: an easy way for you show information inside cards that have a consistent look across the platform



CardView demo

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_margin="10dp"
    card_view:cardBackgroundColor="#ffff6a0"
    card_view:cardCornerRadius="4dp"
    card_view:cardElevation="4dp">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:id="@+id/tvDate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="date"
            android:textSize="24sp"/>
        ...
    </LinearLayout>
</android.support.v7.widget.CardView>
```



Edit Todo feature

- Separate dialog for todo create/edit
- Edit Todo feature
 1. Create a separate Dialog class
 2. Edit button
 3. Show edit dialog

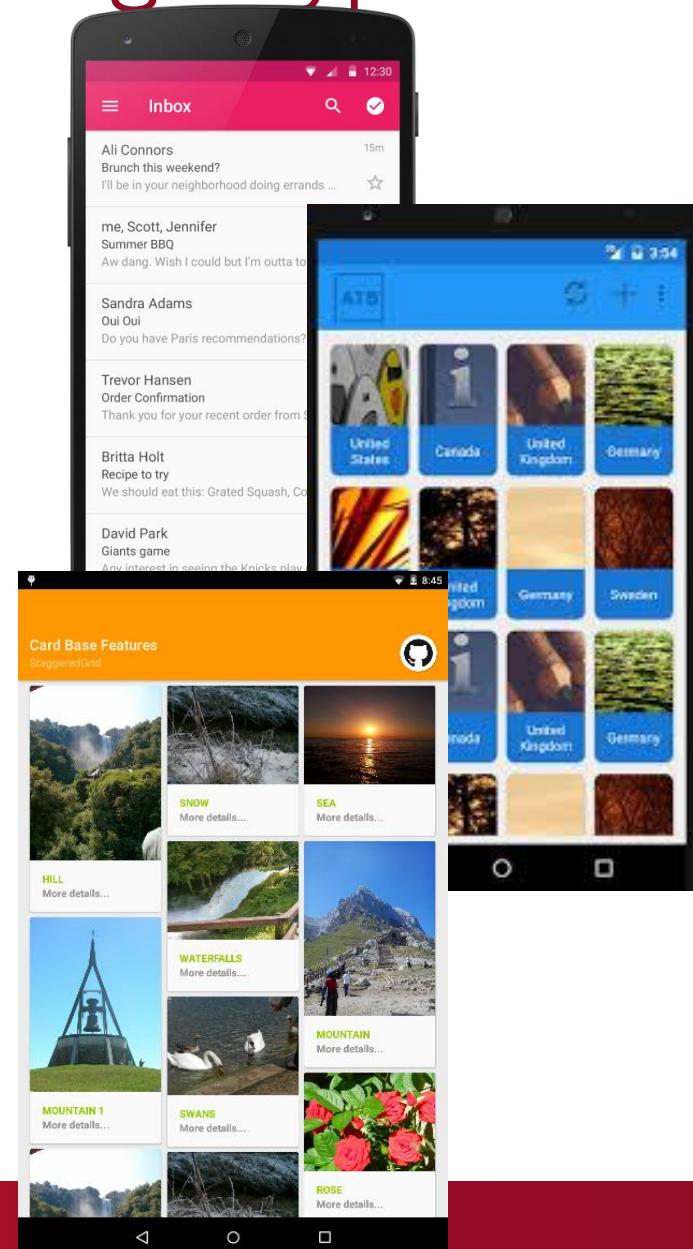
Adding Divider lines

```
val itemDecoration = DividerItemDecoration(this,  
    DividerItemDecoration.VERTICAL)  
recyclerTodo.addItemDecoration(itemDecoration)
```



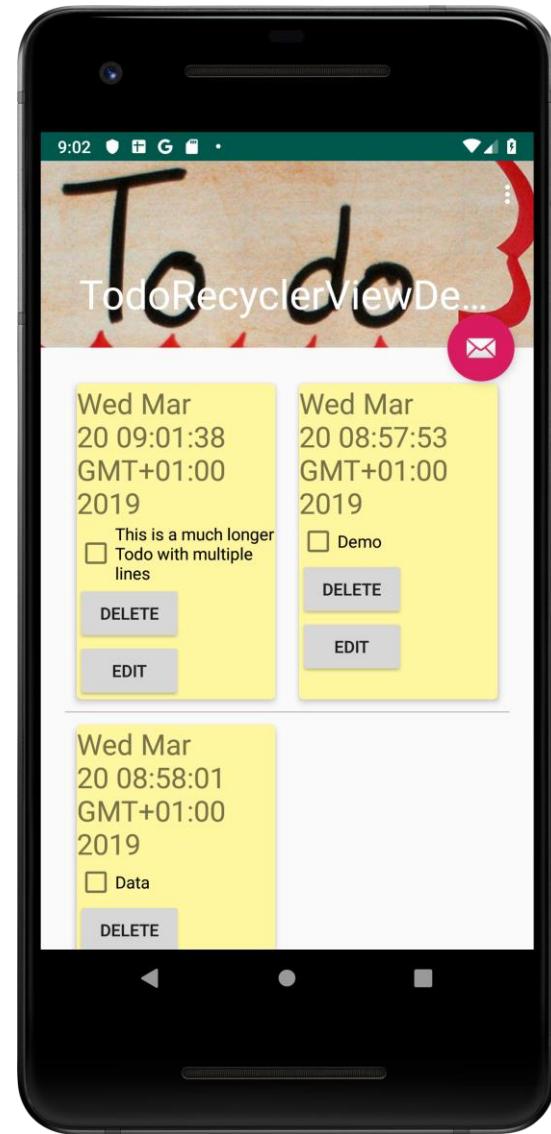
RecyclerView – LayoutManager types

- **LinearLayoutManager:**
 - > Positions items to look like a standard ListView
- **GridLayoutManager:**
 - > Positions items in a grid format similar to a GridView
- **StaggeredGridLayoutManager:**
 - > Positions items in a staggered grid format.



GridLayoutManager demo

```
recyclerTodo.setLayoutManager(  
    GridLayoutManager(this, 2)
```

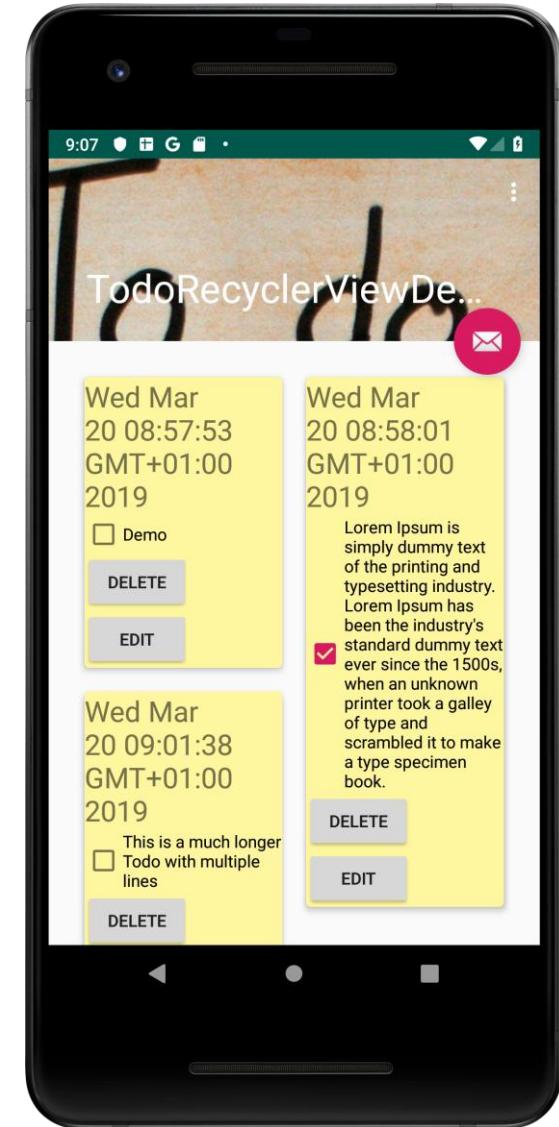


StaggeredGridLayoutManager demo

```
/*
public StaggeredGridLayoutManager (int
spanCount, int orientation)
Creates a StaggeredGridLayoutManager with given
parameters.

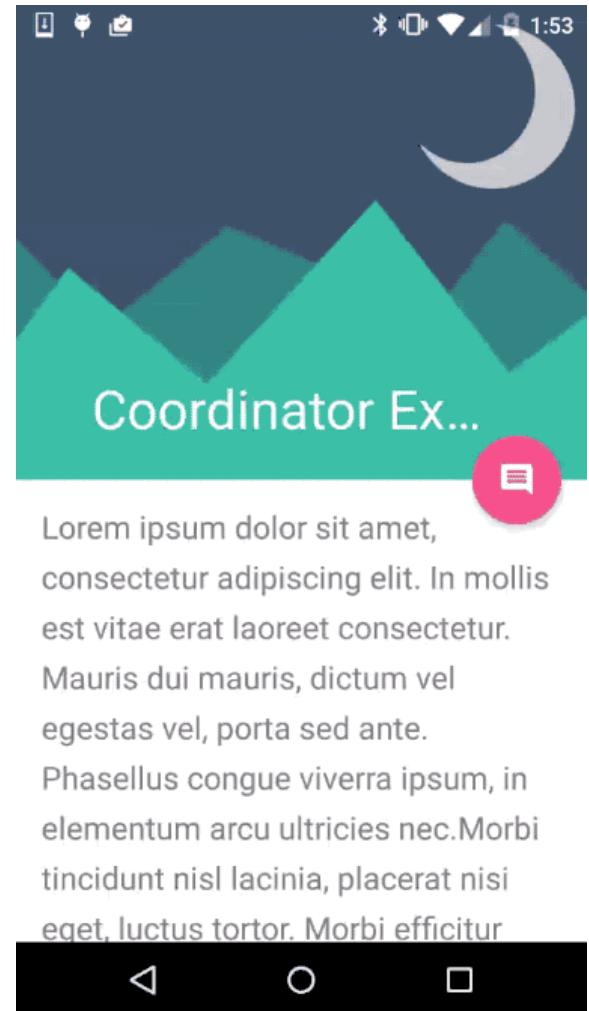
Parameters

```



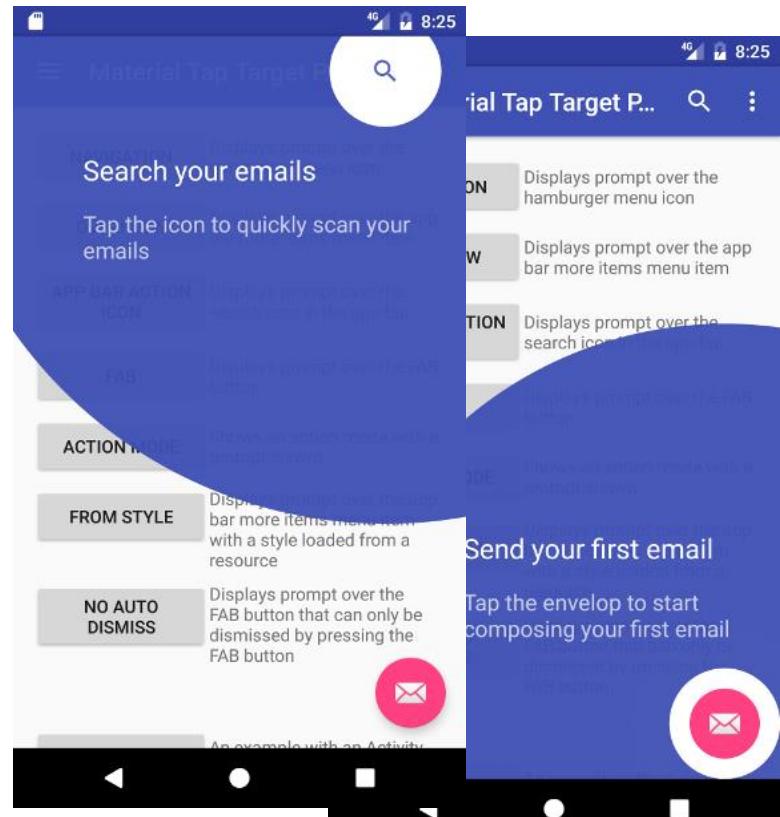
CoordinatorLayout, AppBarLayout

- *CoordinatorLayout* is a super-powered *FrameLayout*.
- *CoordinatorLayout* is intended for two primary use cases:
 - > As a top-level application decor or chrome layout
 - > As a container for a specific interaction with one or more child views
- By specifying Behaviors for child views of a *CoordinatorLayout* you can provide many different interactions within a single parent and those views can also interact with one another.
- *AppBarLayout* is a vertical *LinearLayout* which implements many of the features of material designs app bar concept, namely scrolling gestures.
 - > This view depends heavily on being used as a direct child within a *CoordinatorLayout*.



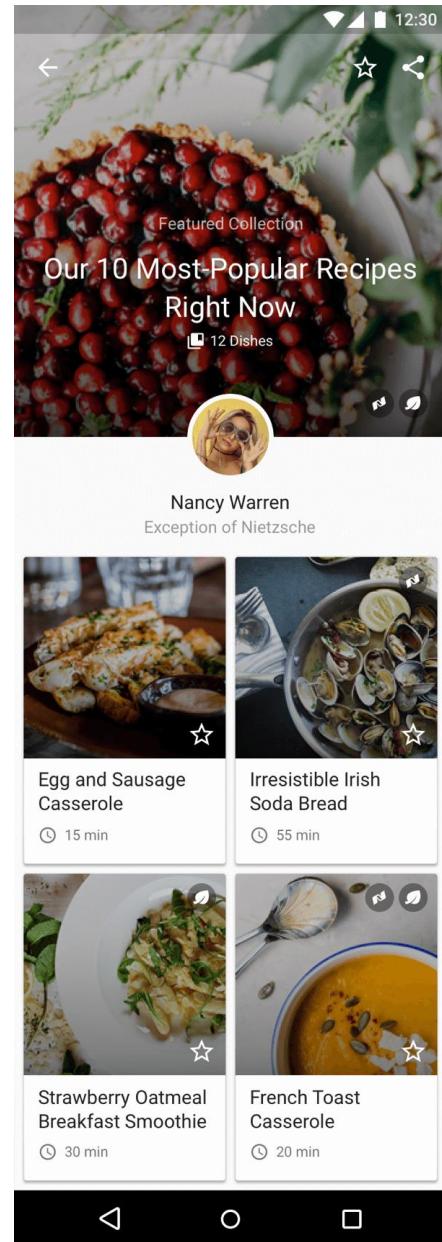
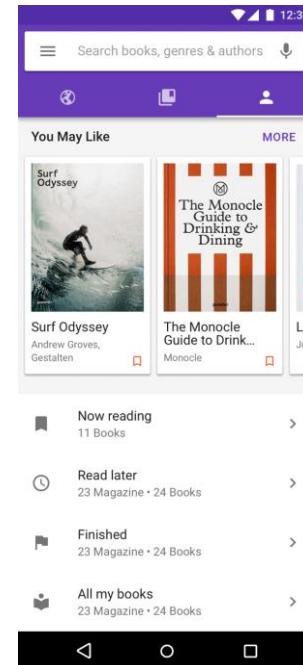
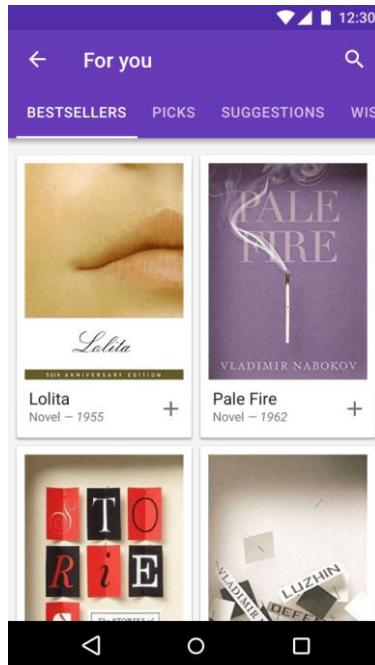
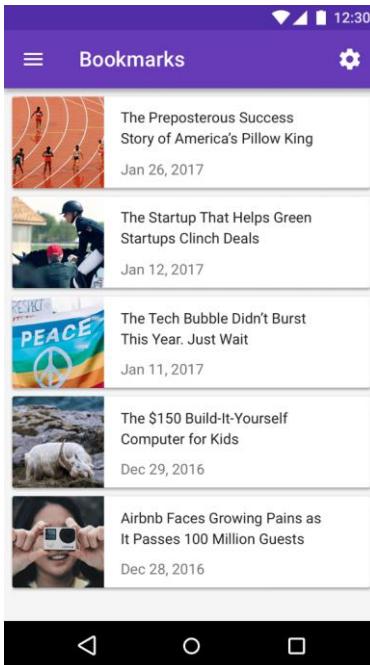
Tutorial view

- <https://github.com/sjwall/MaterialTapTargetPrompt>
- Gradle:
 - > implementation 'uk.co.samuelwall:material-tap-target-prompt:2.3.0'



Design ideas

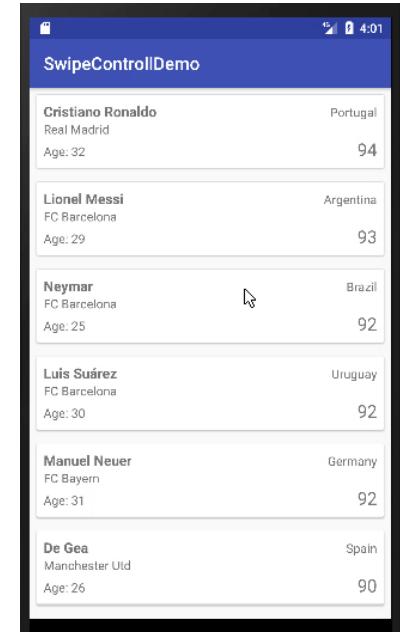
- <https://materialdesignkit.com/templates/>
- <https://www.materialpalette.com/colors>
- <https://material.io/resources/color/#!/?view.left=0&view.right=0>
- <https://material.io/design/color/#color-usage-palettes>



Other RecyclerView solutions

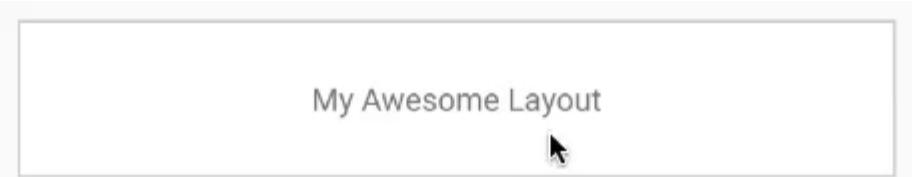
- Swipe tutorial

> <https://codeburst.io/android-swipe-menu-with-recyclerview-8f28a235ff28>



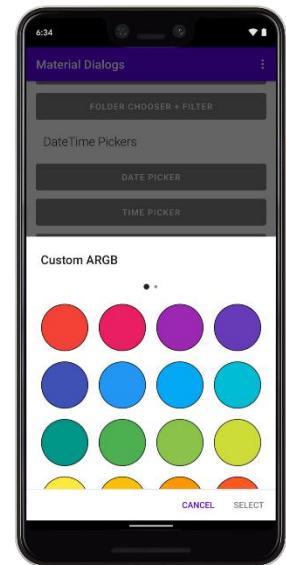
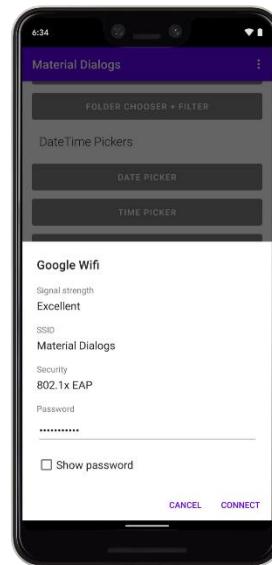
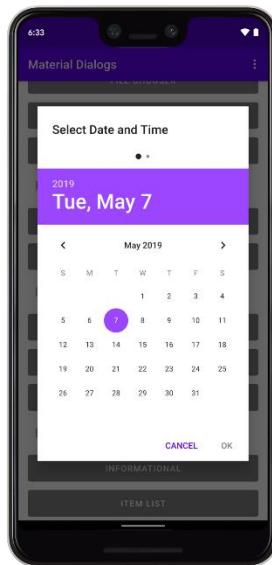
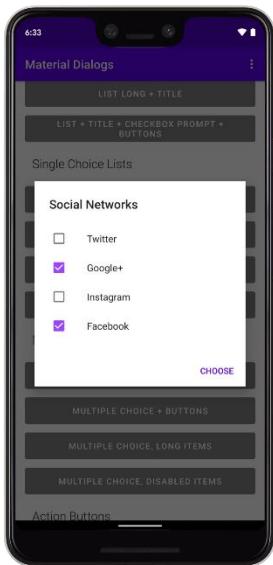
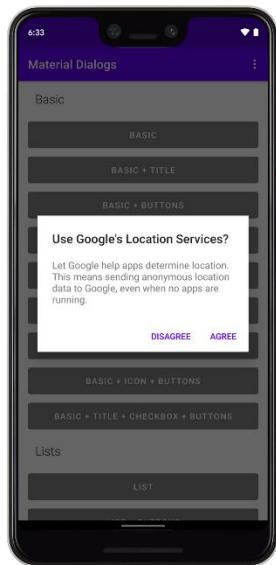
- SwipeToReveal library:

> <https://github.com/chthai64/SwipeRevealLayout>



Alternative dialog library

- Material Dialogs:
 - > <https://github.com/afollestad/material-dialogs>

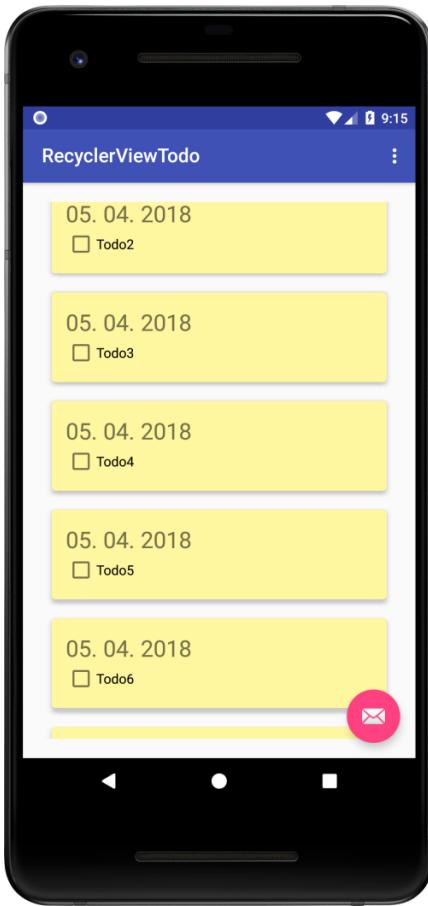


Todo App so far...

- RecyclerView
- Adapter
 - > ViewHolder
 - > onCreateViewHolder(...)
 - > onBindViewHolder(...)
 - > getItemCount()
- CardView
- Add & Edit Todo
- First run prompt
- Room
 - > Entity
 - > DAO class
 - > AppDatabase

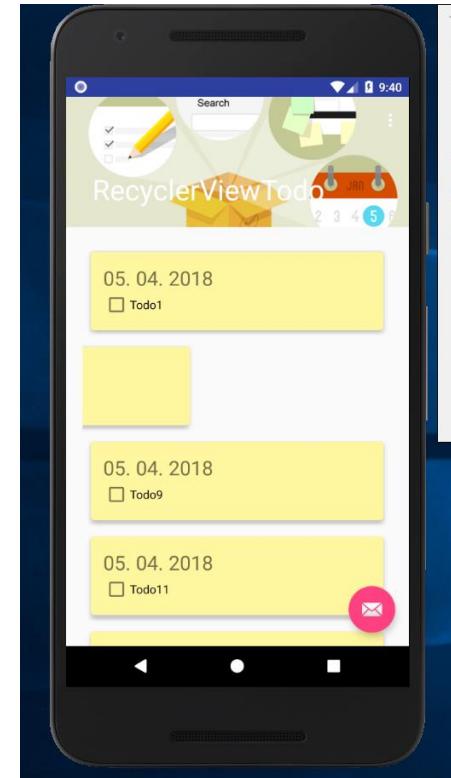
Challenge: Room

- Add persistence storage to the Todo application with Room



Engagement

- Remove Todo from database as well
- Do not forget about the delete button that removes the Todo



Summary

- List
- RecyclerView
 - > Adapter
 - > ViewHolder
 - > Touch
- Room
 - > List
 - > Add
 - > Edit
- Extensions

Codelabs

- Input controls:
 - > <https://codelabs.developers.google.com/codelabs/android-training-input-controls/index.html?index=..%2F..android-training#5>
- RecyclerView:
 - > <https://codelabs.developers.google.com/codelabs/android-training-create-recycler-view/index.html?index=..%2F..android-training#0>
- Cards & colors:
 - > <https://codelabs.developers.google.com/codelabs/android-training-cards-and-colors/index.html?index=..%2F..android-training#0>



Homework – readings

- RecyclerView
 - > <https://developer.android.com/guide/topics/ui/layout/recyclerview.html>
- RecyclerView tutorials:
 - > <https://android.jlelse.eu/using-recyclerview-in-android-kotlin-722991e86bf3>
 - > <https://www.youtube.com/watch?v=hyyX3g57Ms8>
 - > https://www.techotopia.com/index.php/A_Kotlin_Android_RecyclerView_and_CardView_Tutorial
- Dialogs
 - > <https://developer.android.com/guide/topics/ui/dialogs.html>
- Dialogs
 - > <https://developer.android.com/guide/topics/ui/dialogs.html>
- DialogFragment
 - > https://github.com/codepath/android_guides/wiki/Using-DialogFragment
 - > <https://guides.codepath.com/android/using-dialogfragment>



Questions?



Mobile- and Web-based Software

Lecture 9: Animations, Styles, WebView, Permission, Fragment

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

Animations

Animations

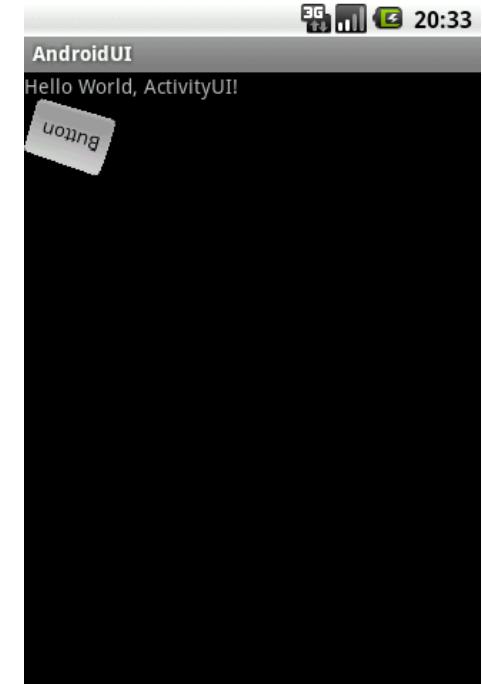
- Animation support
 - > XML based animation description (*res/anim*)
 - > Programmatically
- General animations
 - > *Scale, rotate, translate, alpha, etc.*
- Types:
 - > View animation
 - > Drawable animation
 - > Object/property animation

Simple animation XML

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator=
            "@android:anim/accelerate_interpolator"
        android:fromXScale="0.0"
        android:toXScale="1.0"
        android:fromYScale="0.0"
        android:toYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="1000" />

    <alpha android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="5000"/>

    <rotate
        android:interpolator=
            "@android:anim/accelerate_interpolator"
        android:fromDegrees="0.0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="5000" />
</set>
```



Playing the animation

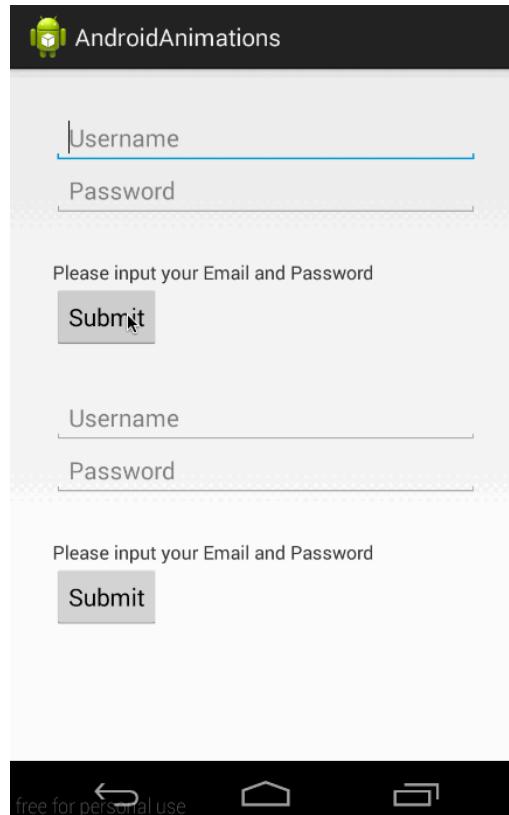
```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    binding = ActivityMainBinding.inflate(layoutInflater)  
    setContentView(binding.root)  
  
    val showAnim = AnimationUtils.loadAnimation(this,  
        R.anim.btnanim)  
    binding.btnAnim.startAnimation(showAnim)  
}
```

Let's practice!

- Animation demo
- Implement an application that animates the whole root layout in case of button press
- Notes:
 - > AnimationListener
 - > fillEnabled/fillAfter
 - > repeatCount

Animations library

- AndroidViewAnimations:
 - > <https://github.com/daimajia/AndroidViewAnimations>



Graphical resources in XML

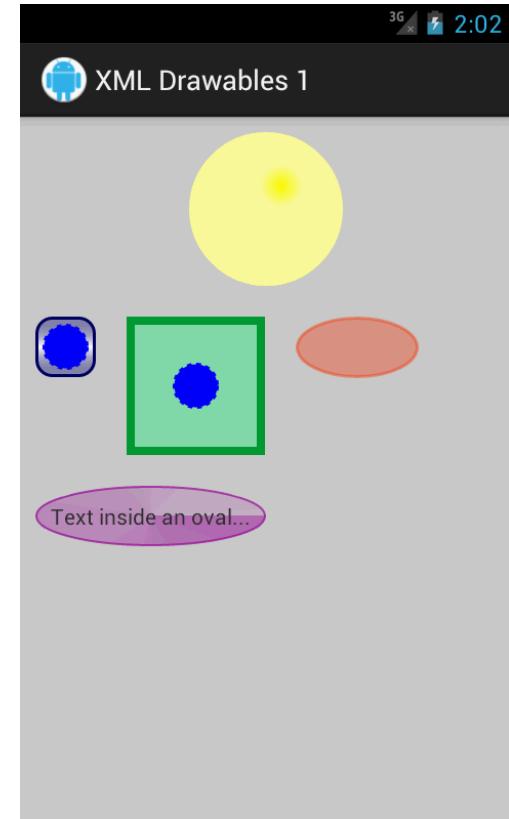
Graphical resources 1/2

- Graphical resources and backgrounds can be described in XML

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">

    <gradient
        android:type="radial"
        android:gradientRadius="20"
        android:centerX=".6"
        android:centerY=".35"
        android:startColor="#FFFF00"
        android:endColor="#FFFF99" />

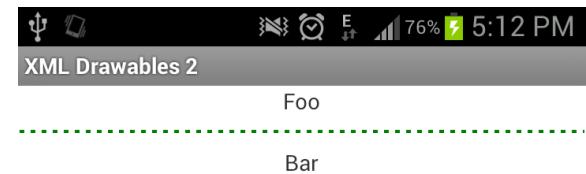
    <size
        android:width="100dp"
        android:height="100dp"/>
</shape>
```



Graphical resources 2/2

- Green dashed line:

```
<?xml version="1.0" encoding="utf-8"?>  
  
<shape xmlns:android=  
    "http://schemas.android.com/apk/res/android"  
        android:shape="line">  
  
    <stroke  
        android:width="2dp"  
        android:color="#008000"  
        android:dashWidth="3dp"  
        android:dashGap="4dp"/>  
  
    <size android:height="20dp" />  
  
</shape>
```



State Definition in Graphical XML

- Button text color based on state:

 > res/color/button_text.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:state_pressed="true"
          android:color="#ffff0000"/><!-- pressed -->

    <item android:state_focused="true"
          android:color="#ff0000ff"/><!-- focused -->

    <item android:color="#ff000000"/><!-- default -->

</selector>
```

- Usage:

```
<!-- use android:layerType="software" only if needed!>

<Button

    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/button_text,"
    android:layerType="software"
    android:textColor="@color/button_text" />
```

Styles and themes

Creating styles

- Style file: res/values/styles.xml

```
<style name="ExampleStyle">  
    <item name="android:textSize">22sp</item>  
    <item name="android:textColor">#0000EE</item>  
</style>
```

- Using a style:

```
<TextView  
    android:id="@+id/tvHello"  
    android:text="@string/hello_world"  
    style="@style/ExampleStyle" />
```

Using themes

- Definition like in styles:

```
<style name="CustomTheme" parent="android:Theme">  
    <item name="android:windowTitleSize">50dip</item>  
    <item name="android:textColor">#000000</item>  
    <item name=  
        "android:windowBackground">@color/white_color</item>  
</style>
```

- Themes can be inherited from each other
- Can set in the Manifest:

```
<activity android:label="" android:name=".MainActivity"  
        android:screenOrientation="portrait"  
        android:theme="@style/CustomTheme"/>
```

Take away App ☺

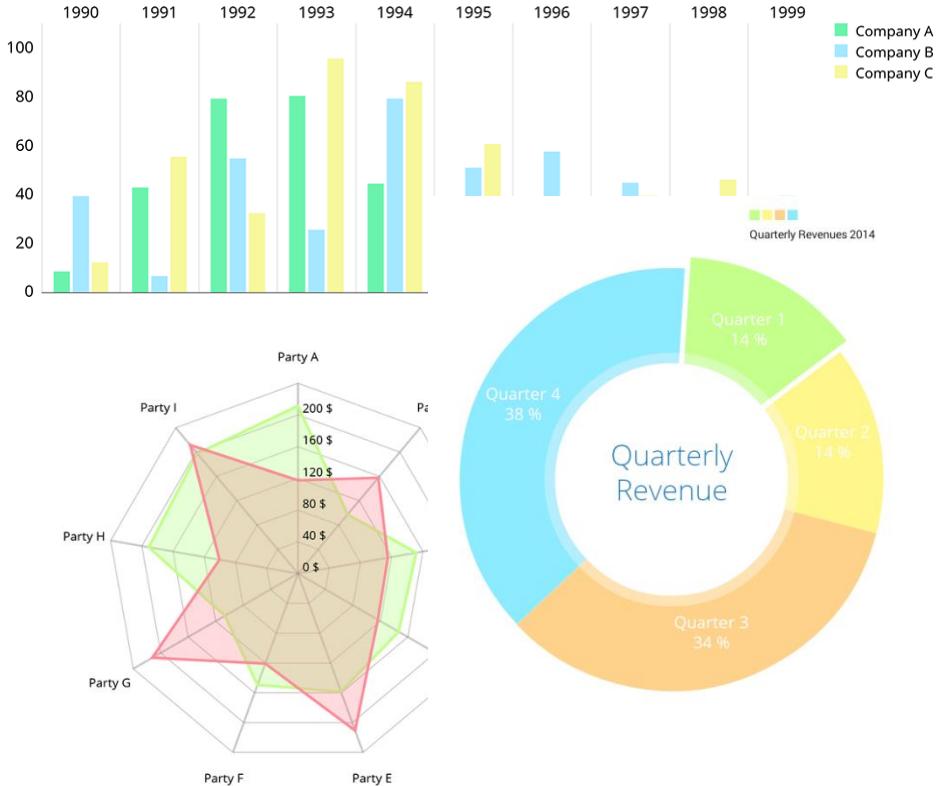
Chart Demo

- Show income and expense on a PieChart



MPAndroidChart

- Powerful charting library
- Complex chart support
 - > LineChart
 - > BarChart
 - > PieChart
 - > CandleStickChart
 - > BubbleChart
 - > ...
- <https://github.com/PhilJay/MPAndroidChart>



MPAndroidChart usage

1. build.gradle and settings.gradle
2. Chart in layout

```
<com.github.mikephil.charting.charts.PieChart  
    android:id="@+id/chartBalance"  
    android:layout_width="0dp"  
    android:layout_height="0dp"  
    .../>
```

3. Setup chart: rotation, gestures, texts, etc.
4. DataSet and entries
5. Color scheme for values
6. Chart and DataSet assignment

Homework – readings

- Animation tutorial:
 - > <http://www.vogella.com/tutorials/AndroidAnimation/article.html>
- View animation:
 - > <https://developer.android.com/guide/topics/graphics/view-animation.html>
- Drawable animation:
 - > <https://developer.android.com/guide/topics/graphics/drawable-animation.html>
- Property animation:
 - > <https://www.raywenderlich.com/128105/introduction-android-animations>
 - > <https://developer.android.com/guide/topics/graphics/prop-animation.html>
- Resource drawables
 - > <https://developer.android.com/guide/topics/resources/drawable-resource.html>



Homework – readings

- Resource drawables
 - > <https://developer.android.com/guide/topics/resources/drawable-resource.html>
- Styles and themes
 - > <https://developer.android.com/guide/topics/ui/look-and-feel/themes.html>



Kotlin Style Guide

- <https://developer.android.com/kotlin/style-guide>

WebView

WebView

- Easy way to embedd website/HTML into your application
- Permission:
 - > <uses-permission android:name="android.permission.INTERNET"/>
- XML resource:

```
<?xml version="1.0" encoding="utf-8"?>  
<WebView xmlns:android=  
         "http://schemas.android.com/apk/res/android"  
         android:id="@+id/webview"  
         android:layout_width="match_parent"  
         android:layout_height="match_parent" />
```

WebView

- Control from Activity:

```
override fun void onCreate( savedInstanceState:  
Bundle? ) {  
  
    super.onCreate(savedInstanceState)  
  
    binding = ActivityMainBinding.inflate(layoutInflater)  
    setContentView(binding.root)  
  
    binding.webView.getSettings() .  
        setBuiltInZoomControls(true)  
  
    binding.webView.loadUrl(  
        "https://www.google.com")  
}
```

Runtime permissions

What are the permissions?

- Dangerous operations require permissions
- Permissions earliery only in the Manifest:

```
<uses-permission android:name=
    "android.permission.WRITE_EXTERNAL_STORAGE"/>
```

- New permission model after Android 6:
 - Dangerous permissions must be requested during runtime

Check runtime permission

- Check permission:

```
ContextCompat.checkSelfPermission(thisActivity,  
    Manifest.permission.WRITE_CALENDAR)
```

- Ask permission from Activity:
 - > Check if we already have the permission
 - > Notify user why we need the permission

Permission types

- Types
 - > Normal permissions
 - > Dangerous permissions
 - > <https://developer.android.com/guide/topics/security/permissions.html#normal-dangerous>
- More details:
 - > <https://developer.android.com/training/permissions/requesting.html>

Permission request 1/2

```
private fun requestNeededPermission() {
    if (ContextCompat.checkSelfPermission(this,
            android.Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                android.Manifest.permission.CAMERA)) {
            Toast.makeText(this,
                    "I need it for camera", Toast.LENGTH_SHORT).show()
        }

        ActivityCompat.requestPermissions(this,
                arrayOf(android.Manifest.permission.CAMERA),
                PERMISSION_REQUEST_CODE)
    } else {
        // we already have this permission
    }
}
```

Permission request 2/2

```
override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<String>, grantResults: IntArray) {
    when (requestCode) {
        PERMISSION_REQUEST_CODE -> {
            if (grantResults.isNotEmpty() && grantResults[0] ==
                PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "CAMERA perm granted",
                    Toast.LENGTH_SHORT).show()
            } else {
                Toast.makeText(this, "CAMERA perm NOT granted",
                    Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Permissions with QuickPermission library

- build.gradle (project level)

```
maven { url "http://jitpack.io/" }
```

```
classpath 'com.github.Questor:QuickPermissions:0.3.2'
```

- build.gradle (app level):

```
apply plugin: 'com.livinglifetechway.quickpermissions_plugin'
```

- Use annotations for methods that require permission, for example:

```
@WithPermissions(  
    permissions = [android.Manifest.permission.ACCESS_FINE_LOCATION]  
)  
fun startLocation() {...}
```

- More info: <https://github.com/QuickPermissions/QuickPermissions>
 - > <https://github.com/QuickPermissions/QuickPermissions-Kotlin>

QR Code reading

QR and Bar code reading

- Zxing library
- Most popular and stable library
- Real time scanner view
- More details:
 - > <https://github.com/zxing/zxing>
 - > <https://github.com/dm77/barcodescanner>

QR code reader demo

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <me.dm7.barcodescanner.zxing.ZXingScannerView
        android:id="@+id/zxingView"
        android:layout_width="match_parent"
        android:layout_height="300dp"/>

    <TextView
        android:id="@+id/tvScan"
        android:textSize="24sp"
        android:autoLink="all"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"/>

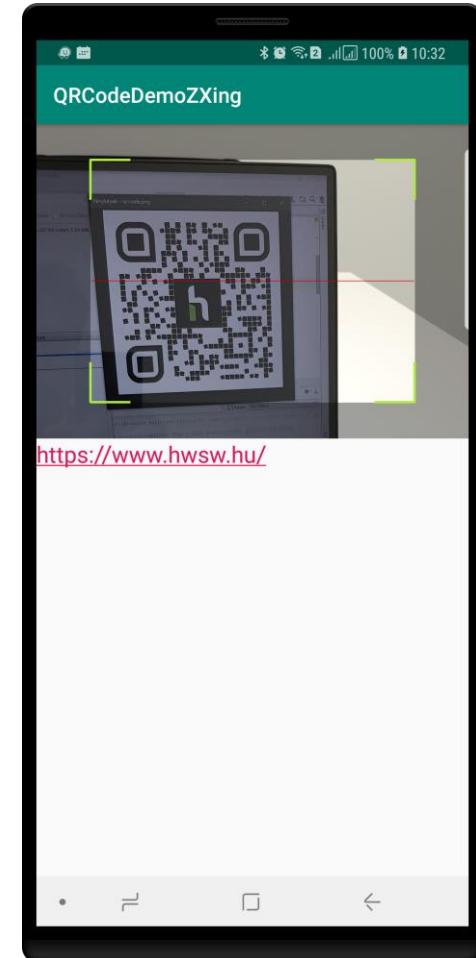
</LinearLayout>
```

QR code reader demo

```
class MainActivity : AppCompatActivity(), ZXingScannerView.ResultHandler {
    private lateinit var binding: ActivityMainBinding
    public override fun onCreate(state: Bundle?) {
        super.onCreate(state)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
    }
    public override fun onResume() {
        super.onResume()
        startCamera()
    }
    @WithPermissions(
        permissions = [Manifest.permission.CAMERA]
    )
    private fun startCamera() {
        binding.zxingView.setResultHandler(this)
        binding.zxingView.startCamera()
    }
    public override fun onPause() {
        super.onPause()
        binding.zxingView.stopCamera()
    }
    override fun handleResult(rawResult: Result) {
        binding.tvScan.text = rawResult.text
        binding.zxingView.resumeCameraPreview(this)
    }
}
```

QR code reader demo

- Generating QR codes:
 - > <https://www.qrcode-monkey.com/>



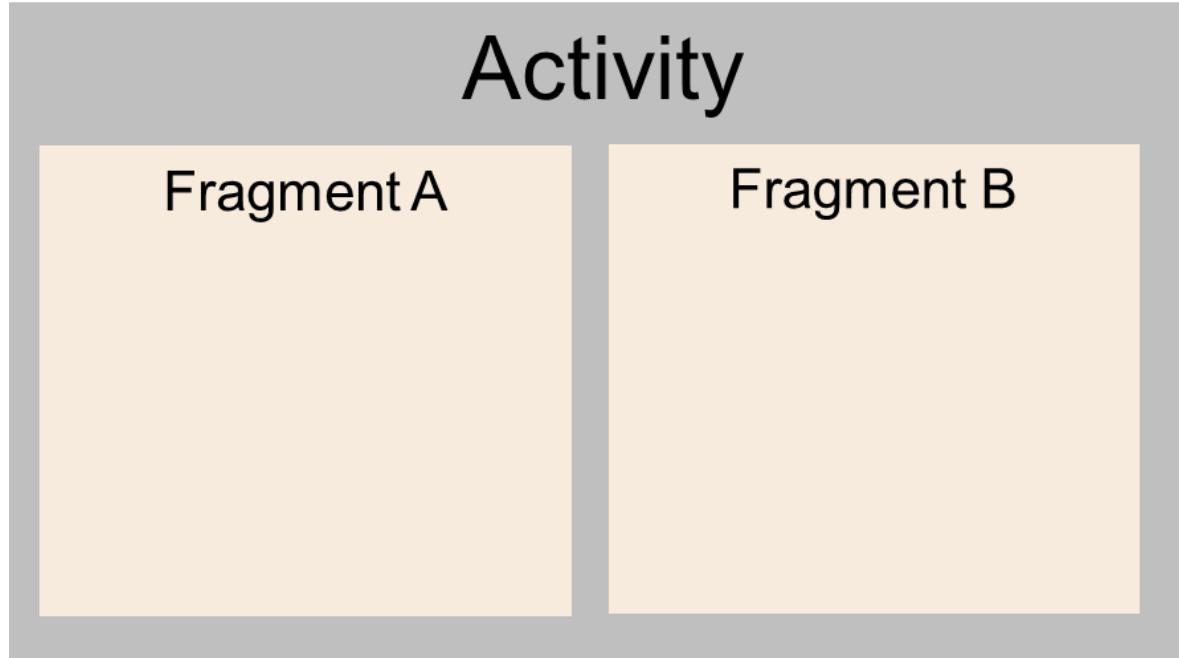
Fragment Framework

Android Fragment Framework

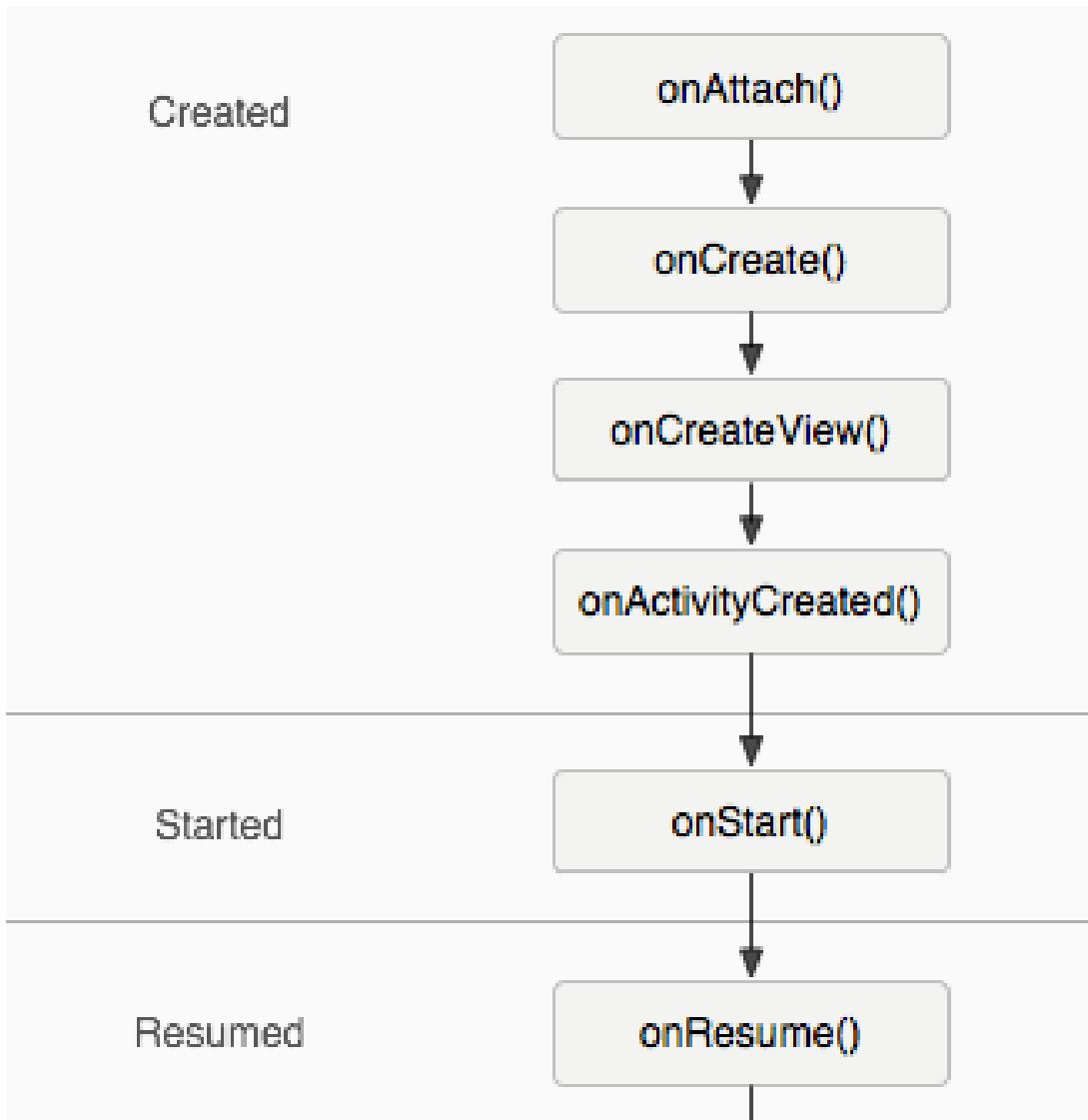
- Fragment: represents a behavior or a portion of user interface in an Activity
- Advantages
 - Modularize the code
 - Reusable: use in multiple activities
 - Adjust the user interface to the screen
 - Multiple layouts based on the size of the display

Fragment and Activity

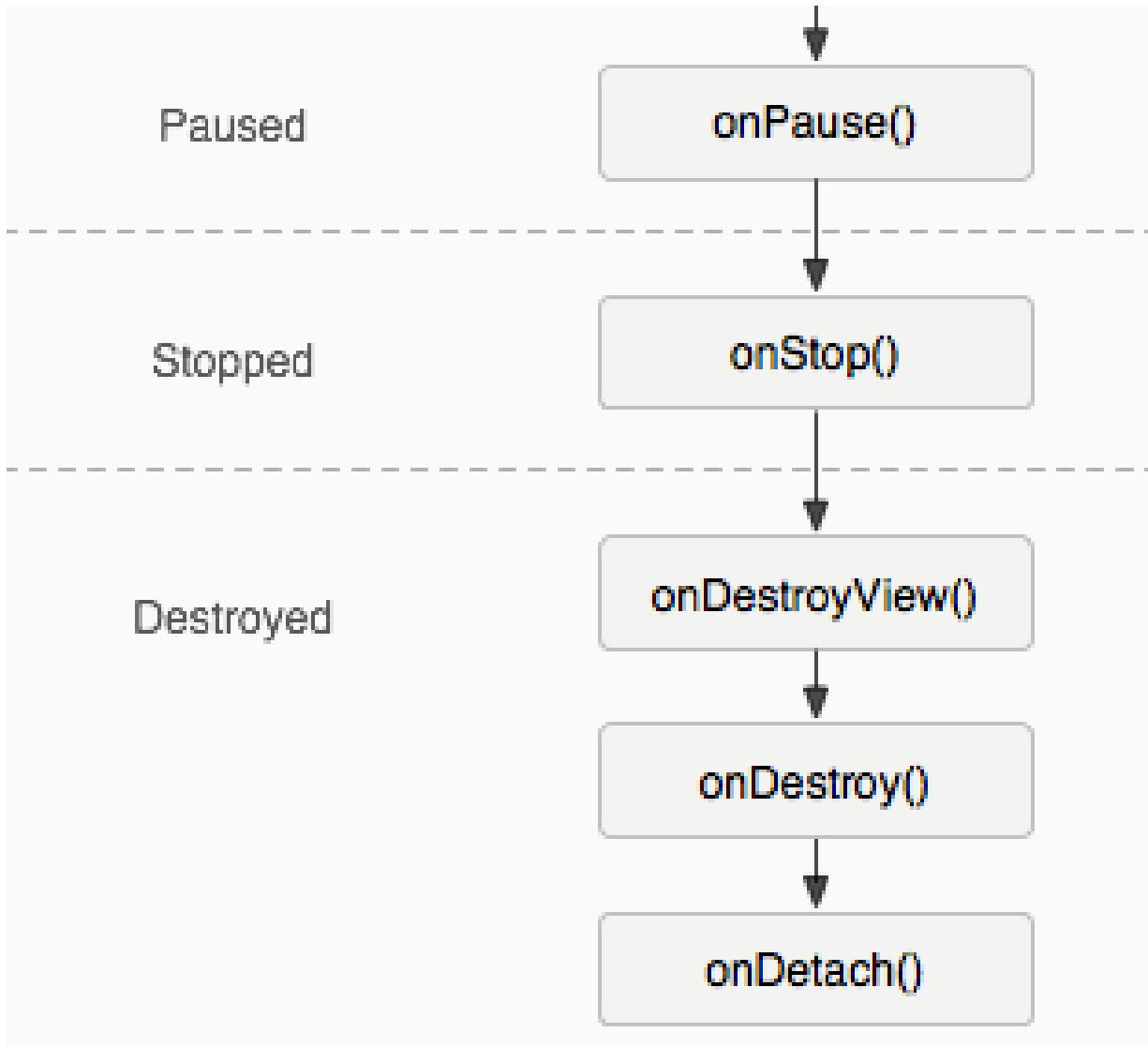
- Fragments are attached to an Activity
- Their lifecycles almost the same

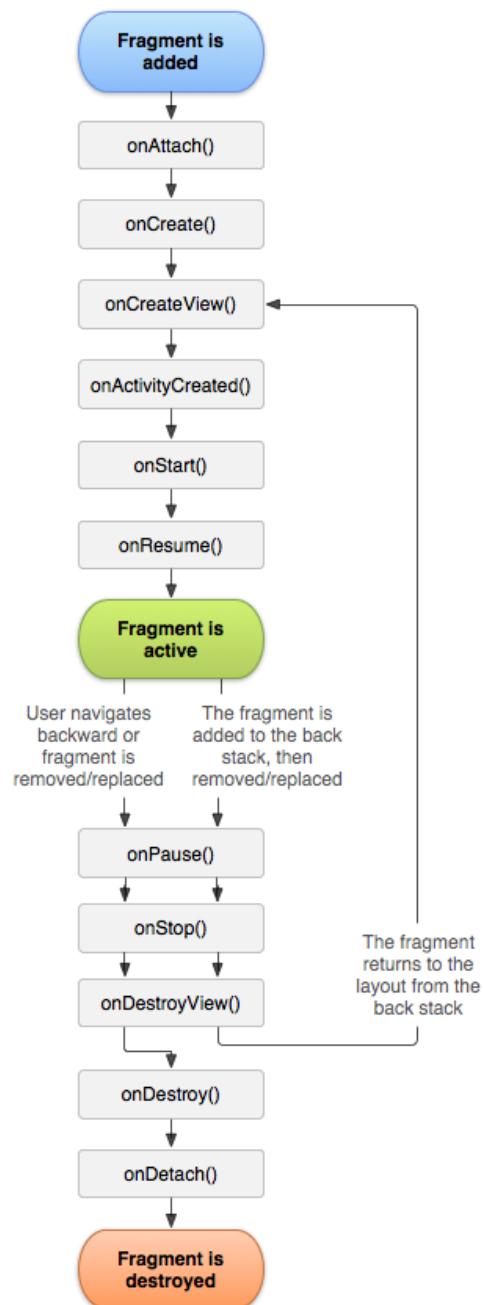


Fragment life cycle 1/2

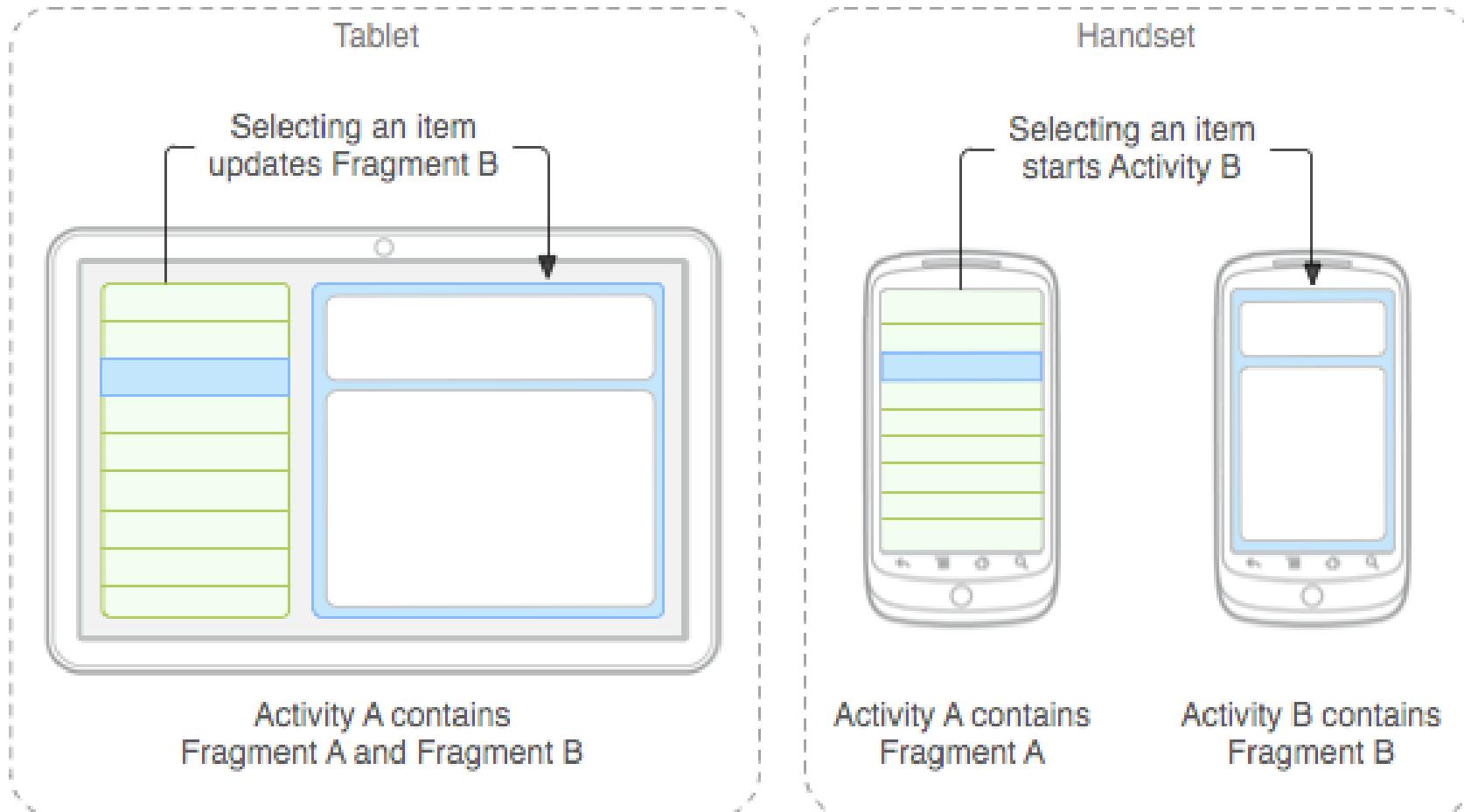


Fragment life cycle 2/2





Different display sizes



UI Fragment

- Create
 - > Extend Fragment class and override onCreateView() method

```
class FragmentOne : Fragment() {  
  
    companion object {  
        const val TAG = "TAG_FRAGMENT_ONE"  
    }  
  
    override fun onCreateView(  
        inflater: LayoutInflater,  
        container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
        val rootView = FragmentOneBinding.inflate(inflater, container, false)  
        return rootView  
    }  
}
```

UI Fragment

➤ Attach

- Dynamic

- Load the proper Fragment in runtime

- Static

- In Layout XML

```
<fragment class="hu.bme.aut.fragment.MenuListFragment"  
    android:tag="MenuListFragment"  
    android:layout_width="0dip"  
    android:layout_height="fill_parent"  
    android:layout_weight="1"/>
```

Dynamic attachment

- Manage fragments with FragmentManager class
 - Activity: .getFragmentManager()
 - Starting FragmentTransaction
 - Find fragments with id or tag
 - **Managing Fragment-stack**

FragmentTransaction

- Modify active fragments through FragmentTransaction
- Starting with the beginTransaction () method in FragmentManager
- Important methods
 - **add () , remove () , replace ()**: add or remove fragments in an Activity
 - **commit ()**: execute transaction
 - **show () , hide ()**: show or hide fragment instance
 - **setTransition () , setCustomAnimations ()**
 - **addToBackStack ()**: transaction can be added to back stack

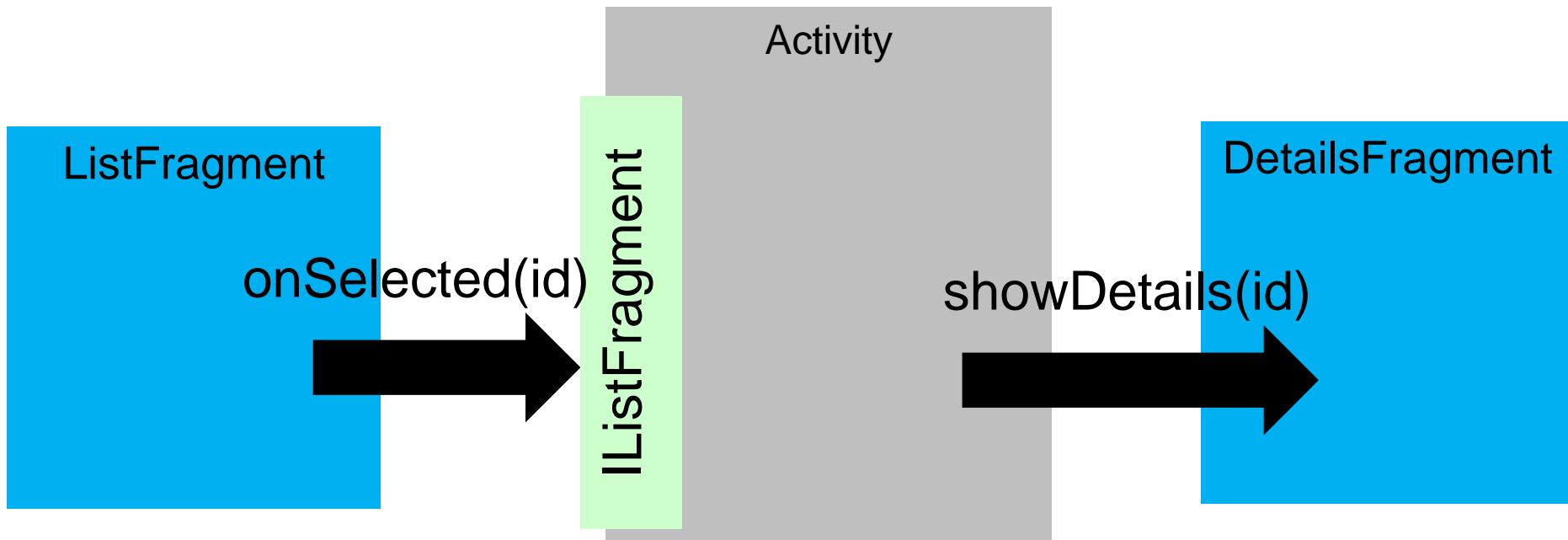
FragmentTransaction example

- Fragment replace:

```
val ft = supportFragmentManager.beginTransaction()  
  
ft.replace(  
    R.id.fragmentContainer,  
    FragmentMain(),  
    FragmentMain.TAG)  
  
ft.commit()
```

Fragment communication 1/2

- The *Activity* handles the communication between *Fragments*



Fragment communication 2/2

- If direct communication is required between Fragments:
 - > `.setTargetFragment(...)`
 - > `.getTargetFragment(...)`
- This connection survives even the screen rotation!

Let's practice!

- Create a simple fragment based application.
 - > Practice the static and dynamic attachment.
- Extend the application to have two fragments!

Fragments without UI

- If the *Fragment* is not attached to a *ViewGroup*, it runs in the background!
- `.setOnRetainInstanceState(true)`
 - > During screen rotation it is detached from the fragment and attached again, but keeps it in the memory!
 - No need for saving the state
 - > Can be used instead of`.onRetainNonConfigurationInstance()`

DialogFragment

- A Fragment can appear as a Dialog
 - > AlertDialog.Builder can be used to build commonly used dialogs
 - > Dialogs can have custom layouts
- DialogFragments have the same Fragment life cycle
- DialogFragments can also be on the Fragment Back Stack
- onCreateDialog()
 - > The method returns the Dialog to be displayed

Fragment parameters

- A Fragment is created with a simple constructor with no parameters (convention)
- When parameters are required for the Fragment:
 - > Send the parameters in a Bundle
 - .setArguments(...)
 - > During initialization this Bundle is queried
 - .getArguments()

Embedded fragments

- A fragment can hold another fragment
 - > From API 17
 - > Implemented also in the Support Library
- Dynamic, re-usable UI fragments

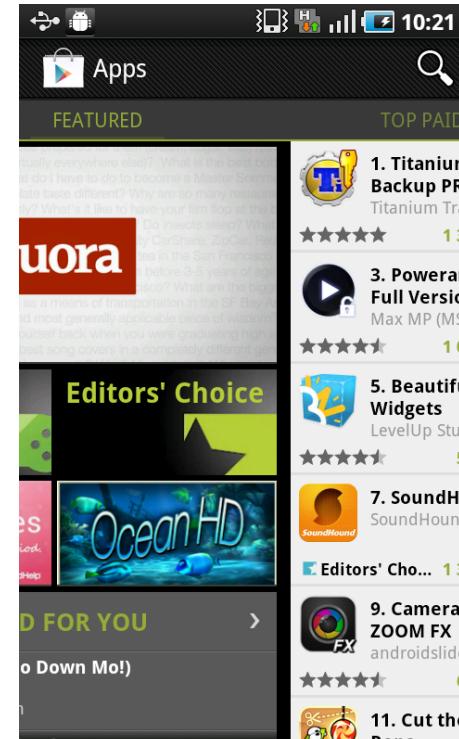
```
val videoFragment = new VideoPlayerFragment();  
  
val transaction =  
getFragmentManager().beginTransaction();  
  
transaction.add(R.id.video_fragment,  
videoFragment).commit();
```

- The embedded fragment can reach the owner FragmentManager:

```
getParentFragment()
```

ViewPager

- *ViewGroup*, that supports swipe animation for changing between *Fragments*
 - > E.g.: Google Play Store application

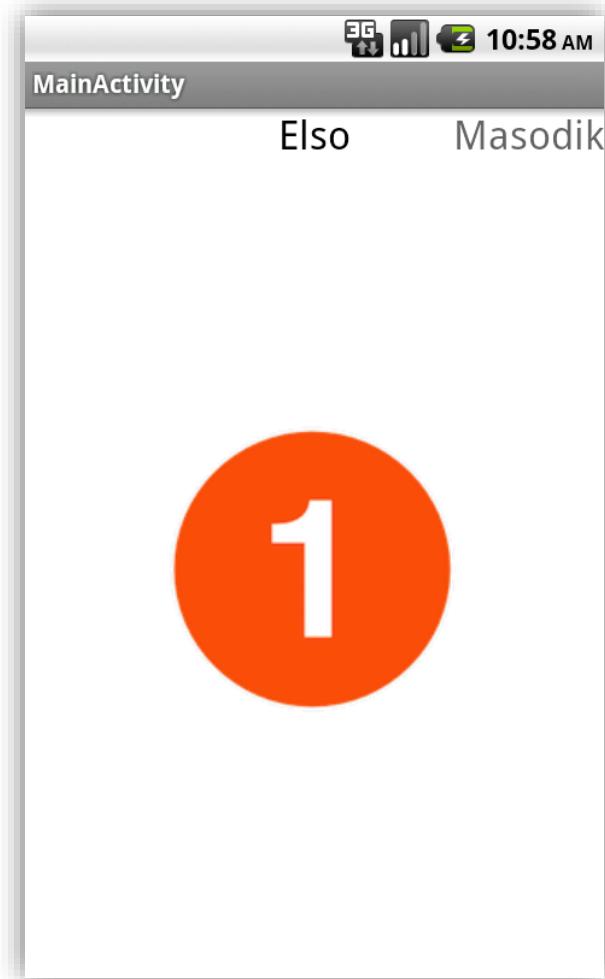


FragmentPagerAdapter

- Typically *ViewPager* sites contain *Fragments*
- A *FragmentPagerAdapter* manages the *Fragments* (“sites”):
 - > Similar to *BaseAdapter*
 - > *Fragment getItem(int position)*:
 - Returns the current *Fragment* instance
 - > *int getCount()*:
 - Returns the number of pages
 - > *String getTitle(int position)*:
 - Returns the title of the page

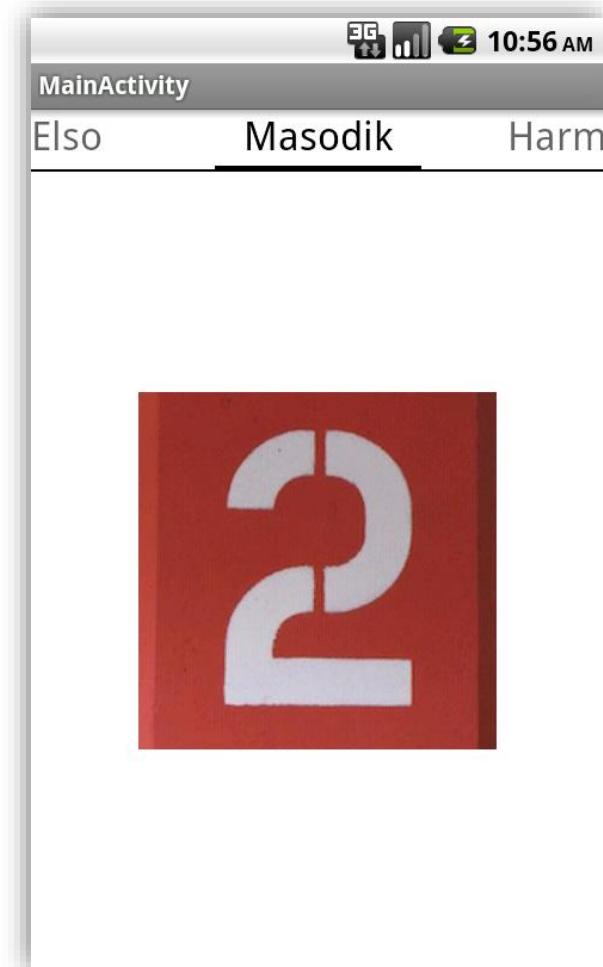
PagerTitleStrip

- Widget, that shows the title of the page on the *ViewPager*
 - > The official *SupportLibrary* contains it
 - > It is not interactive!



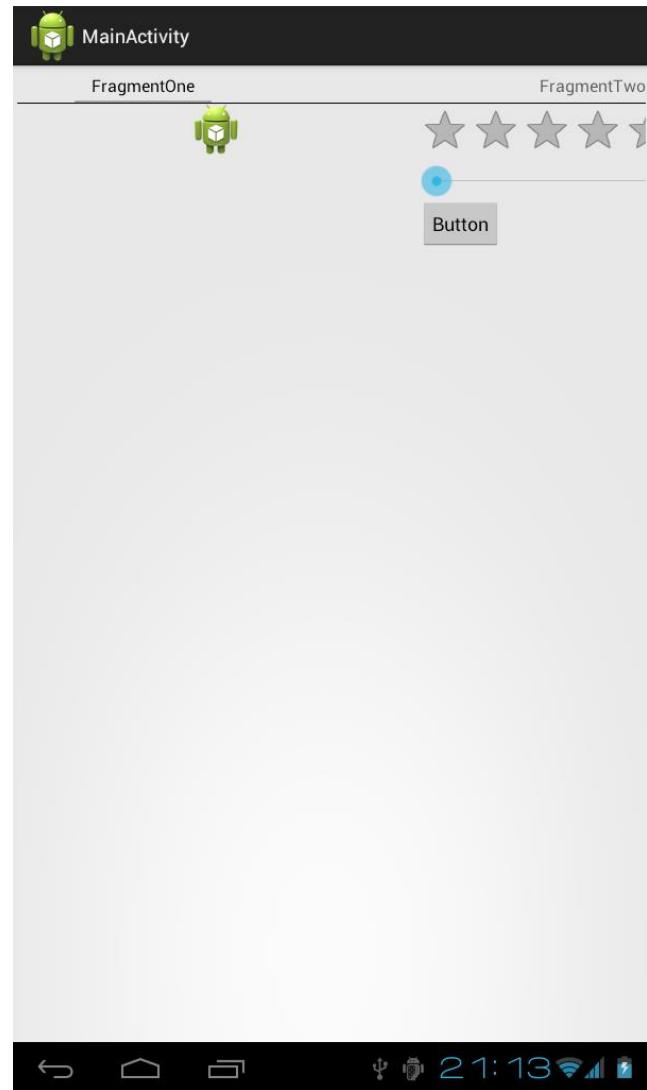
PagerTabStrip

- Widget, that shows the title of the page on the *ViewPager*
 - > The official *SupportLibrary* contains it
 - > It is interactive!



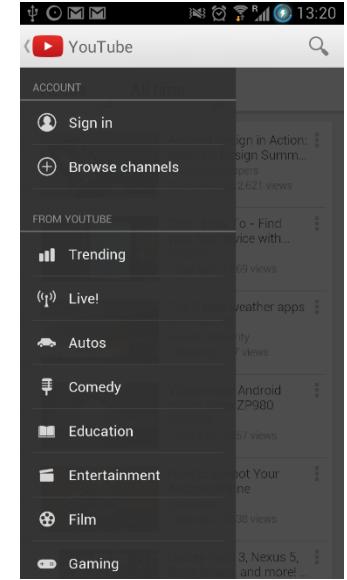
Let's practice!

- Create an application in where we can switch between fragments via a *ViewPager*!
- Try out the PagerTitleStrip and PagerTabStrip elements!



Android Navigation Patterns

- Patterns:
 - > Action Bar
 - > Navigation Drawer
 - > Swipe views
 - > Confirming and Acknowledge
 - > Notifications
 - > More:
<https://developer.android.com/design/patterns/index.html>
- Navigation Drawer:
 - > Common navigation component / Menu
 - > More: <https://developer.android.com/training/implementing-navigation/nav-drawer.html>



Readings

- Fragments:
 - > <http://developer.android.com/guide/components/fragments.html>
- Building a Dynamic UI with Fragments:
 - > <http://developer.android.com/training/basics/fragments/index.html>

Mobile midterm test info

Wednesday 16th October 2024. 18:15-19:45 IB025

Mobile topics only until this slide!

Android Homework Assignment

- Specification Deadline: 3th November 23:59
- Project Deadline: 1th December 23:59
- Instructions on Moodle

Questions?



Mobile- and Web-based Software

Lecture 10: ContentProvider, BroadcastReceiver, Service

David Sik

david.sik@aut.bme.hu



Department of
Automation and
Applied Informatics

Content Provider

ContentProvider

- Main objective of *ContentProvider* is to share a data source with other components
- The data can be stored in different locations: SQLite, file, *SharedPreferences*, remote server, etc.
- Other applications can work on the same data by using *ContentProvider*
- For example: CallLog *ContentProvider*
- Extends from the
`android.content.ContentProvider` class

Reacting to global events

BROADCAST RECEIVER

Broadcast events

- An app can subscribe to certain system events
- Intent is suitable to describe the event
 - > Name: *Broadcast Intent*
- The only component that can react on a broadcast: **Broadcast Receiver**
- More broadcast events with every Android version
- This is still an Intent, it may contain additional data (about the event)

System broadcasts

ACTION_TIME_TICK
ACTION_TIME_CHANGED
ACTION_TIMEZONE_CHANGED
ACTION_BOOT_COMPLETED
ACTION_PACKAGE_ADDED
ACTION_PACKAGE_CHANGED
ACTION_PACKAGE_REMOVED
ACTION_PACKAGE_RESTARTED
ACTION_PACKAGE_DATA_CLEARED

ACTION_UID_REMOVED
ACTION_BATTERY_CHANGED
ACTION_POWER_CONNECTED
ACTION_POWER_DISCONNECTED
ACTION_SHUTDOWN
....

Broadcast Receiver

- Register in Manifest or from code, subscribe to Intent Actions
- Subscription by „*Intent Filter*”, an inner node in the manifest
- BR is not an Activity, no user interface...
- ...but it can start an Activity, display a Toast or Notification...
- ...or do anything else that a Kotlin class can do

Example BroadcastReceiver

```
class OutgoingCallReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val outNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER)
        Toast.makeText(context, outNumber, Toast.LENGTH_LONG).show()
    }
}
```

AndroidManifest.xml:

```
<receiver android:name=".OutgoingCallReceiver">
    <intent-filter>
        <action android:name=
            "android.intent.action.NEW_OUTGOING_CALL"/>
    </intent-filter>
</receiver>
```

Practice!

- Create a Broadcast Receiver which reacts on change of „Airplane mode” and displays a Toast message.

Running in the background

SERVICE

Services

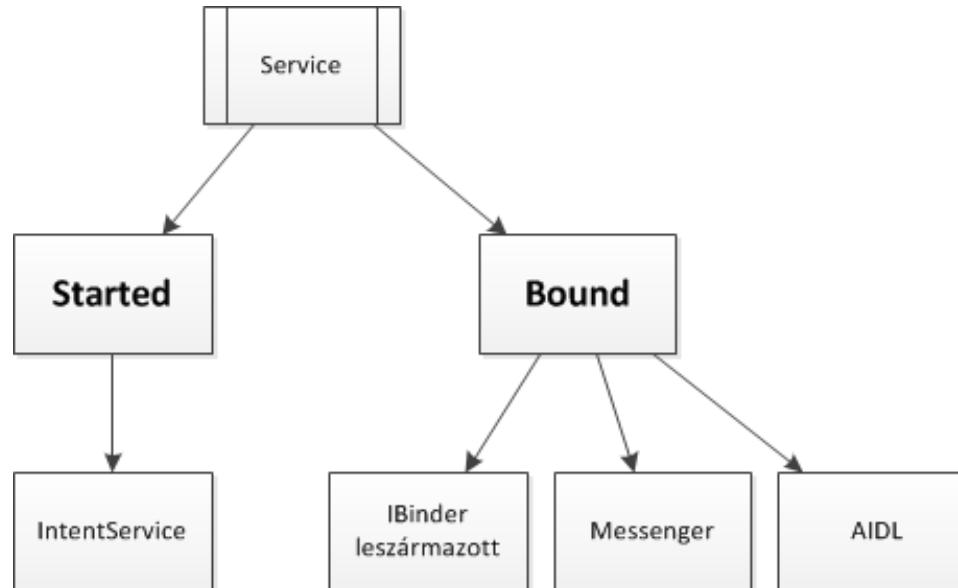
- Applications may need components that run in the background
- Even when the app itself is not in the foreground
- Music player, synchronizing service, torrent, ...
- Android allows developers to create **services**
- The only component that can run (for indefinite time) when app is not in front

Services

- No user interface
- No interaction with the user
- Can serve other components / apps
- The only component that can be declared private
- Started by Intent with `startService()` or `bindService()`
 - > Like starting an Activity with `startActivity()`
- Cannot have a result as an Activity can

Service types

- Two type of operation:
 - > *Started from component / Started*
 - > *Bunded / Bound*



Service types

- Two different types for different use-cases
 - > For a single task: **Started / Unbound Service**
 - (E.g.: download a file)
 - > For continuous service: **Bound Service**
 - (E.g.: music player)

Started Service

- An other component starts it by `startService(intent)`
 - > Usually started from an Activity
- Service runs even after its creator is destroyed
- This Service type is intended for a single action, not continuous running
 - > Download or upload one file, play a single music track
- If the Service has finished the job, it has to shut down itself by `stopSelf()`

Bound service

- Not started directly, it is started by Android if someone tries to connect to it
- It runs until there are connected clients, no need to stop itself
- Several components can bind to it simultaneously
- Intended to be used for long-term tasks
 - > Full-featured music player, Torrent downloader
- All system services are bound

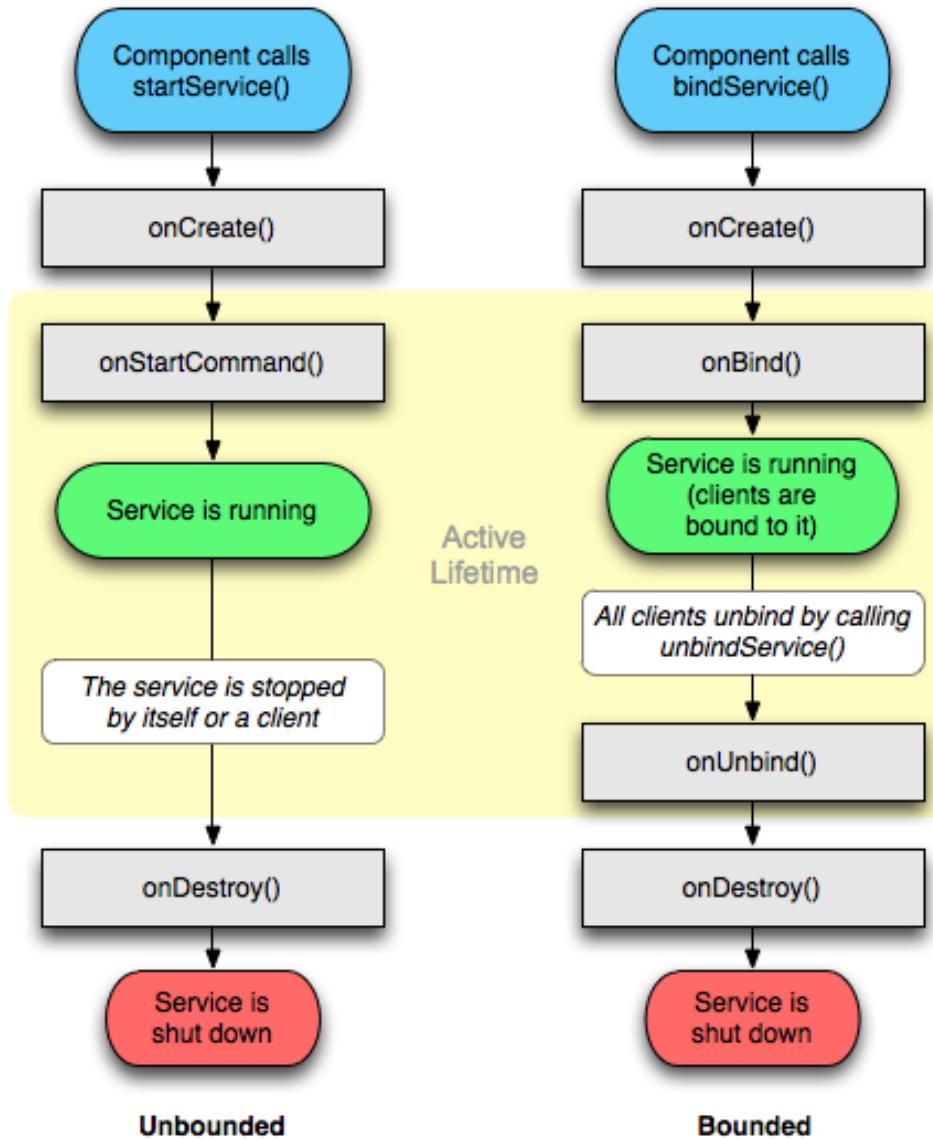
Service operation types

- Service can support both started and bound operation
- If the Service is started we can not get a reference to it (it can be done only for Bound services)
- Example for Service operation with both features
 - > Music Player
 - > Music starts when Started Service starts
 - > Music runs in the background
 - > We connect to the player (bound) to switch music

Service implementation

- Extend from Service class
- Implement the proper callback methods
 - > In case of Started Service:
 - Override *onStartCommand()*
 - > In case of Bound Service:
 - Override *onBind()*

Service Life Cycle



Service handling

- Other application can also use the Service (in both operation types)
- Communication via Intents
- It is also possible to create private Services:
 - > Manifest attributes

Service attributes

- Service is an application component, it must be declared in AndroidManifest.xml

```
<service android:enabled=["true" | "false"]  
         android:exported=["true" | "false"]  
         android:icon="image resource"  
         android:label="Title of the Service"  
         android:name="name"  
         android:permission="permission"  
         android:process="process name" >
```

Service attributes

- *enabled*: Can be used or not
- *exported*: Allow other apps to use the Service
- *icon*: Icon if the service is startable from app launcher
- *label*: Name of the service in *Settings/Running services*
- *name(*)*: Classname of the implementation
- *permission*: What permissions are needed to use the Service (custom permissions can be defined)
- *process*: Which process should the Service run in. If it is started by colon, system will create a new process

Operations inside the Service

- By default Started and Bound Services do not have separate Threads!!!
- For long operations we have to define the Thread:
 - > CPU intensive operations (e.g. encryption, encoding, etc.)
 - > Network communication
- If we do not use Thread -> *Application Not Responding* (ANR) dialog

STARTED SERVICE

Service implementation

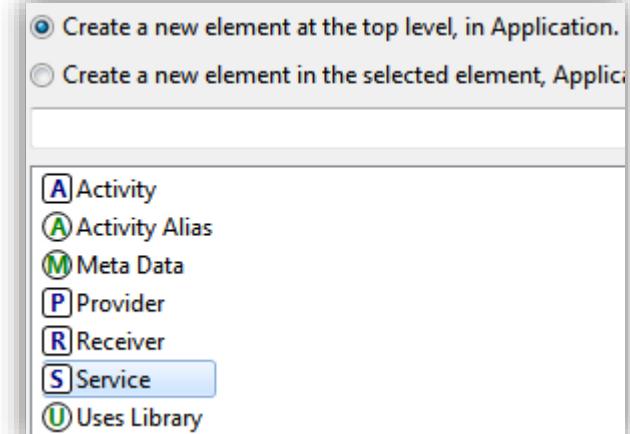
1. Extend from `android.app.Service`

```
①import android.app.Service;  
  
public class DownloaderService extends Service {
```

2. Override necessary functions

```
@Override  
public IBinder onBind(Intent intent) {  
    // TODO Auto-generated method stub  
    return null;  
}  
  
@Override  
public void onCreate() {  
    // TODO Auto-generated method stub  
    super.onCreate();
```

3. Registration in the Manifest

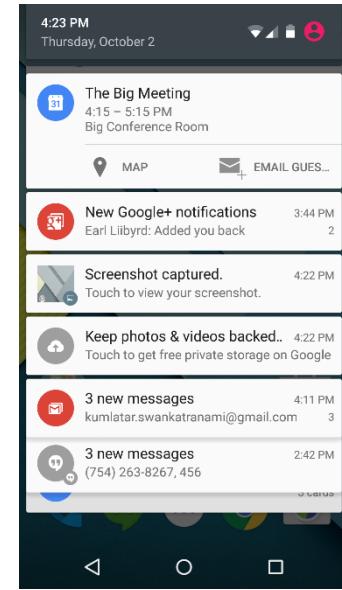


Service restart

- Service: runs in the background
- OS can kill a service in case of low memory
- 3 different restart modes
- Return value of *onStartCommand()*:
 - > Service.START_STICKY
 - > Service.START_NOT_STICKY
 - > Service.START_REDELIVER_INTENT

Notifications

- A message you can display to the user outside of your application's normal UI.
- First appears as an icon in the **notification area**.
- User opens the **notification drawer**.
- Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.
- Supports custom layouts



Simple Notification

```
var mBuilder =
    NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");
// Creates an explicit intent for an Activity in your app
var resultIntent = Intent(this, ResultActivity::class.java);

// The stack builder object will contain an artificial back stack for the
// started Activity.
// This ensures that navigating backward from the Activity leads out of
// your application to the Home screen.
var stackBuilder = TaskStackBuilder.create(this);
// Adds the back stack for the Intent (but not the Intent itself)
stackBuilder.addParentStack(ResultActivity.class);
// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
    stackBuilder.getPendingIntent(
        0,
        PendingIntent.FLAG_UPDATE_CURRENT
    );
mBuilder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
// mId allows you to update the notification later on.
mNotificationManager.notify(mId, mBuilder.build());
```

How to add action buttons to Notification?

```
var notification = NotificationCompat.Builder(this)
    .setContentTitle("This the MyTimeService")
    .setContentText(text)
    .setSmallIcon(R.mipmap.ic_launcher)
    .setVibrate(long[]{1000, 2000, 1000})
    .addAction(R.mipmap.ic_launcher,"Play",playIntent)
    .setContentIntent(contentIntent).build();
```

Let's practice!

- Extend our previous Started Service demo with a Notification that shows the current time in every 5 seconds on a Notification!

FOREGROUND SERVICE

Android resource management

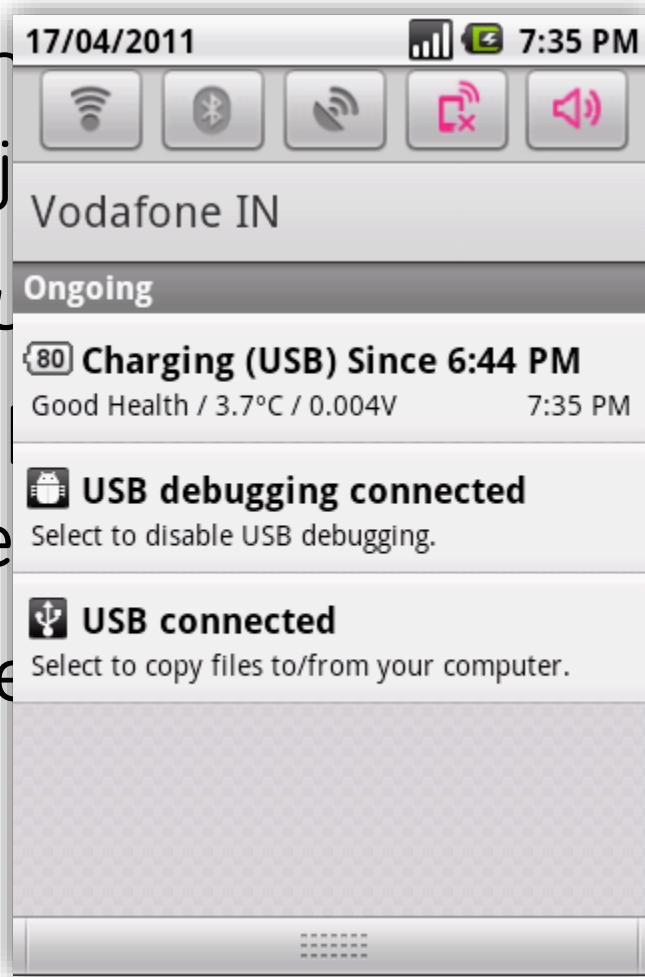
- Android releases resources when foreground application needs more
 - > Kill background apps to get more CPU, and memory
- Services may be closed as well
- If a Service does something that the user is actively aware of, it should not be killed by the system
 - > E.g. music player, USB connection

Android resource management

- An already running service can declare itself important, and jump to a higher priority level
- Called *Foreground Service*
- Must provide a Notification which will be displayed in the „Ongoing” section
- If the critical execution is finished, Service can go back to normal priority without exiting

Android resource management

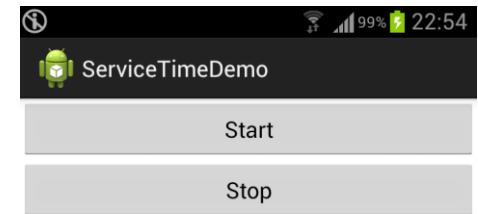
- An already running application is important, and justifies the priority level
- Called *Foreground*
- Must provide a visual representation displayed in the notification bar
- If the critical exception occurs back to normal



lare itself
ority level
will be
Service can go
ing

Let's practice!

- Create a Started Service that displays the time in a Toast in every 5 seconds.
- Make it possible to start and stop the service.
- Create a Notification bar entry that shows that the service is running.



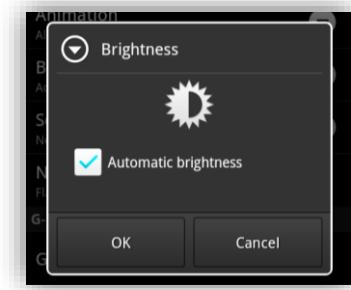
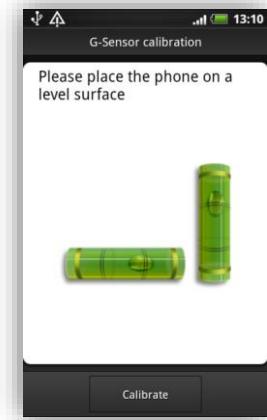
SENSORS

Introduction

- The devices contain many built-in sensors
 - > Microphone
 - > Accelerometer
 - > Compass
 - > Light sensor
 - > Temperature sensor
 - > Etc.

Supported sensors

- Android supports many real and “abstract” sensors:
 - Sensor.TYPE_ACCELEROMETER: three axis accelerometer, the current value is returned in m/s^2
 - Sensor.TYPE_GYROSCOPE: rotation degree to all three axis
 - Sensor.TYPE_LIGHT: ambient light in *lux*. The operating system uses it for setting the screen brightness automatically.



Supported Sensors

- Sensor.TYPE_MAGNETIC_FIELD:
Magnetic power on all three direction.
It is necessary for compass
applications.
- Sensor.TYPE_ORIENTATION:
orientation sensor, typically we query
the SensorManager.getOrientation()
value
- Sensor.TYPE_PROXIMITY: Distance
between the device and another
object. E.g.: the screen is turned off
when moving it close to our head.



Working with sensors

- A system service is used to reach the sensors:
 - > **SensorManager**
- Get reference to SensorManager:

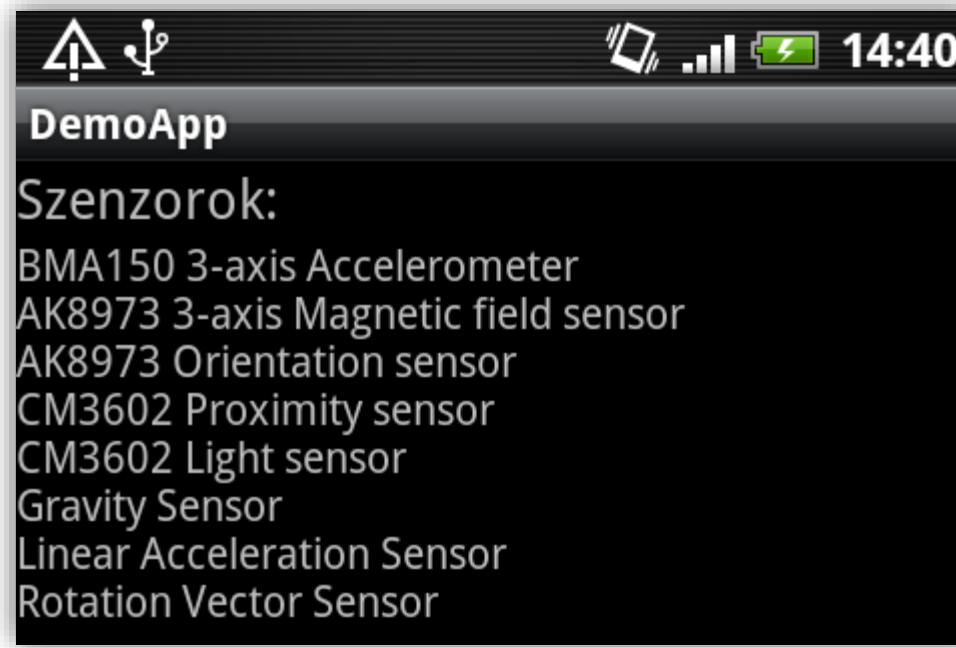
```
val sensorManager =  
    getSystemService(SENSOR_SERVICE) as SensorManager
```

- Sensor instance:

```
val sensor = sensorManager.getDefaultSensor(  
    Sensor.TYPE_MAGNETIC_FIELD)  
sensorManager.registerListener(this, sensor,  
    SensorManager.SENSOR_DELAY_NORMAL)
```

List all sensors

```
sensorManager.getSensorList(Sensor.TYPE_ALL).foreach {  
    tvStatus.append("${it.name}\n")  
}
```



Using sensors

- Event driven programming by implementing the *SensorEventListener* interface
 - > *onSensorChanged(SensorEvent)*: When sensor value changes
 - > Values:
 - Sensor
 - Accuracy of the measurement
 - Measured values in a float[]
 - Nanosecond timestamp of the measurement

Standard implementation

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var sensorManager: SensorManager
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager

        ...
    }

    override fun onStop() {
        super.onStop()
        sensorManager.unregisterListener(this);
    }

    private fun listSensors() {
        sensorManager.getSensorList(Sensor.TYPE_ALL).forEach {
            binding.tvStatus.append("${it.name}\n")
        }
    }

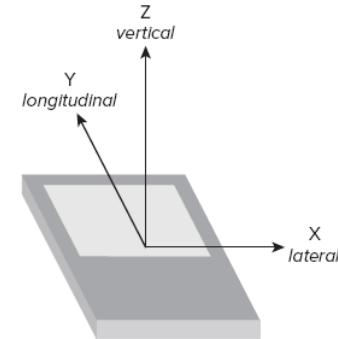
    private fun startSensor() {
        val sensor = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)
        sensorManager.registerListener(this, sensor,
            SensorManager.SENSOR_DELAY_NORMAL)
    }

    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}

    override fun onSensorChanged(event: SensorEvent) {
        binding.tvStatus.setText("Magneto: ${event.values[0]}")
    }
}
```

How to use accelerometer

- Measures acceleration on three axes (it is not the speed!)
- Phone on back:
 - > X axis: right-left
 - > Y axis: forward-backward
 - > Z axis: up-down
- By default Z measures the gravity: (in Hungary ~9.81 m/s²)



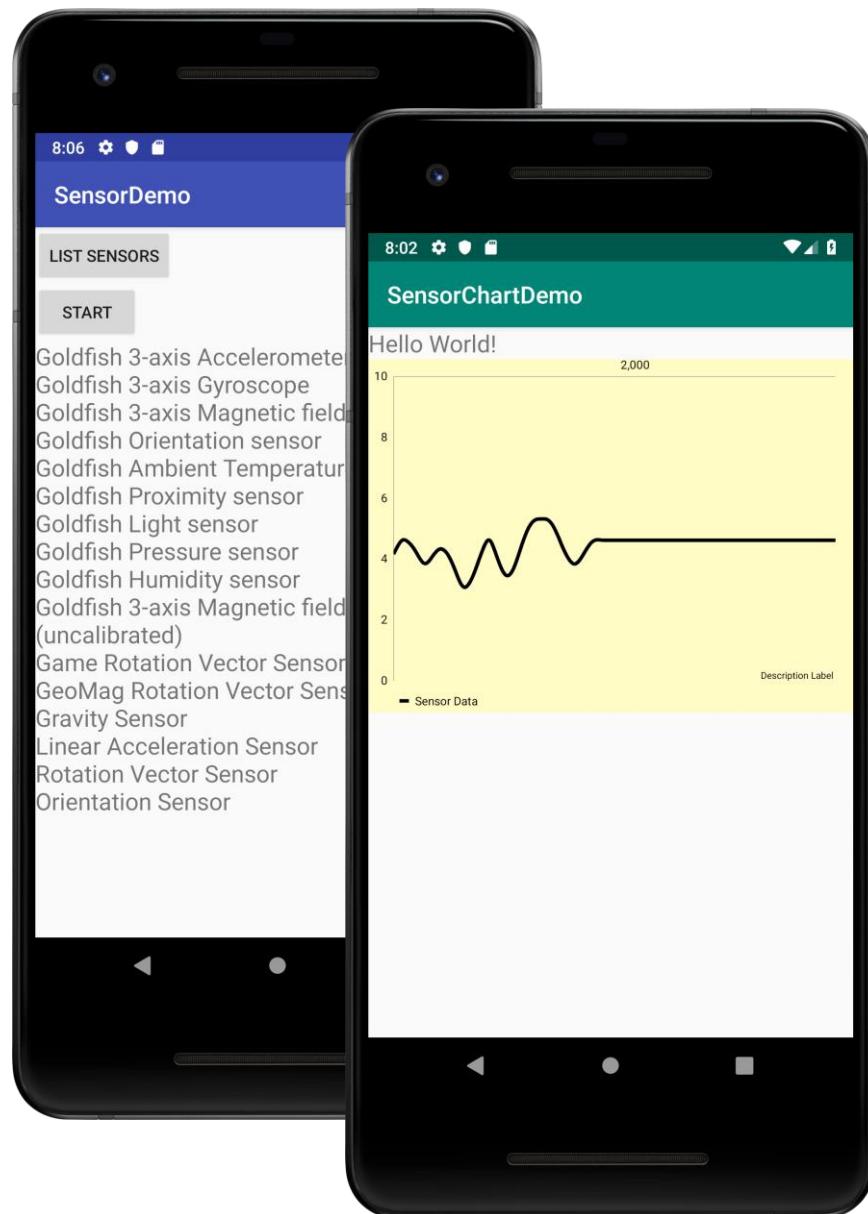
How to use accelerometer

- Understanding measurement data:
 - (G power calculation = add the squares of the three acceleration values then get the root of them then minus the gravity)

```
override fun onSensorChanged(event: SensorEvent) {  
    val accX = event.values[0].toDouble()  
    val accY = event.values[1].toDouble()  
    val accZ = event.values[2].toDouble()  
  
    var origin = Math.sqrt(  
        Math.pow(accX, 2.0) +  
        Math.pow(accY, 2.0) +  
        Math.pow(accZ, 2.0)  
    )  
    origin = Math.abs(origin - SensorManager.STANDARD_GRAVITY)  
}
```

Sensor Demos

- SensorDemo
- SensorChartDemo



WORKING WITH THE MAP

Map view

- Display map data
- Full control over the map
 - > Location, zoom level, etc.
 - > Map, satellite, traffic, etc.
- Overlay support
- POI objects on the map
- Custom tile providers can be implemented



Google Maps API V2

- Google Play Services SDK
- MapFragment
- Vector graphics based rendering
 - > fast
 - > less network traffic
- Advanced map cache
- 3D support

How to create map view

- MapFragment based map object
- On-demand map-tile download
- Manifest permissions
- OpenGL ES
- Requires project registration and API key generation
 - > Google APIs console
 - > <https://code.google.com/apis/console/>

MapFragment parameters

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:map="http://schemas.android.com/apk/res-auto"  
    android:id="@+id/map"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    class="com.google.android.gms.maps.SupportMapFragment"  
    map:cameraBearing="112.5"  
    map:cameraTargetLat="-33.796923"  
    map:cameraTargetLng="150.922433"  
    map:cameraTilt="30"  
    map:cameraZoom="13"  
    map:mapType="normal"  
    map:uiCompass="false"  
    map:uiRotateGestures="true"  
    map:uiScrollGestures="false"  
    map:uiTiltGestures="true"  
    map:uiZoomControls="false"  
    map:uiZoomGestures="true"/>
```

Demo – MapFragment



Show MapFragment 1/3

- Create new project/new Activity from Google Maps Activity template
- Details/generated code:
 - > Gradle:
compile 'com.google.android.gms:play-services-maps:16.0.0'
 - > Import only the relevant play services library:
 - <https://developers.google.com/android/guides/setup>
 - > Manifest meta:
`<meta-data android:name="com.google.android.gms.version
" android:value="@integer/google_play_services_version" />`
 - > (remove ProGuard)
 - > Keystore SHA-1:
`keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias
androiddebugkey -storepass android -keypass android`
 - > Registration in Google API Console:
<https://code.google.com/apis/console/?noredirect>
 - > Manifest key:
`<meta-data
 android:name="com.google.android.maps.v2.API_KEY"
 android:value="API_KEY"/>`

Show MapFragment 2/3

- Permissions:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- The following two permissions are not required to use
     Google Maps Android API v2, but are recommended. -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

- OpenGL ES 2

```
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true"/>
```

- MapFragment

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.google.android.gms.maps.MapFragment"/>
```

Show MapFragment 3/3

- SupportMapFragment if backward compatibility is required
- MapFragment holds the GoogleMap object
- GoogleMap object control:

```
val mMap =  
    ((MapFragment) getFragmentManager() .  
     findFragmentById(R.id.map)) .getMap()  
  
mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID)  
mMap.setTrafficEnabled(true)
```

- Google Maps API AVD (Emulator)

Map example

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
    private lateinit var myMap: GoogleMap
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    override fun onMapReady(googleMap: GoogleMap) {
        myMap = googleMap

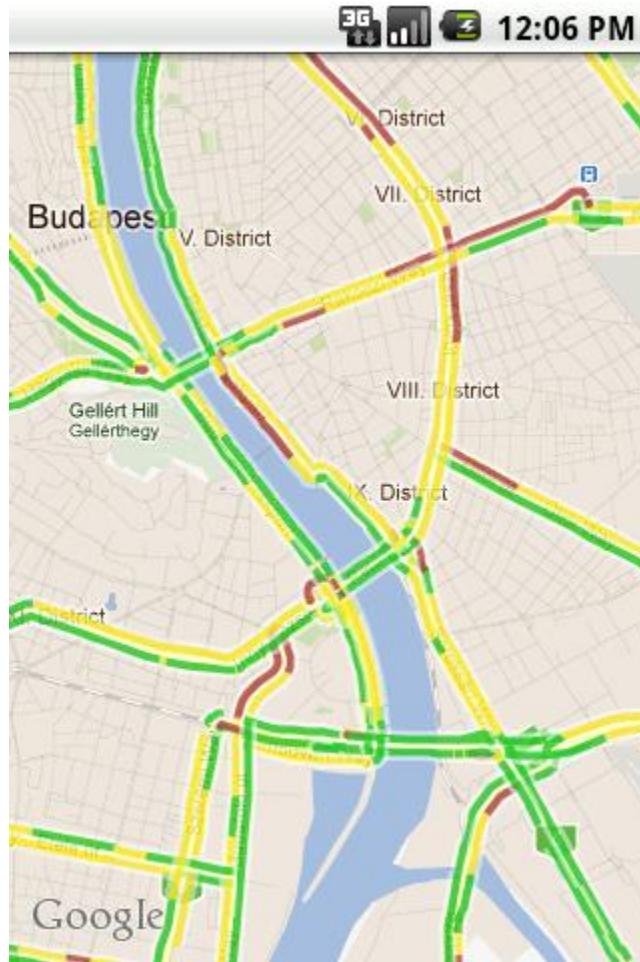
        myMap.isTrafficEnabled = true
        myMap.mapType = GoogleMap.MAP_TYPE_SATELLITE

        val budapest = LatLng(47.0, 19.0)
        myMap.addMarker(MarkerOptions()
            .position(budapest)
            .title("Marker in Hungary"))
        myMap.moveCamera(CameraUpdateFactory.newLatLng(budapest))
    }
}
```

Possible issues with Map

- „Google Play services out of date. Requires 9877000 but found 9875470”
 - > Use Play Services 9.6.1
 - compile 'com.google.android.gms:play-services-maps:9.6.1'
- Error: Execution failed for task ':app:transformClassesWithDexForDebug'.>
com.android.build.api.transform.TransformException:
com.android.ide.common.process.ProcessException:
java.util.concurrent.ExecutionException:
com.android.dex.DexIndexOverflowException: method ID not in
[0, 0xffff]: 65536
 - > Import only the relevant Play Services library (e.g. only the maps):
 - compile 'com.google.android.gms:play-services-maps:9.6.1'

Demo – Traffic view



Control map object

- Touch event
 - > `GoogleMap.setOnMapClickListener (OnMapClickListener)`

- UiSettings object:

```
GoogleMap myMap;
```

```
...
```

```
myMap.uiSettings.isRotateGesturesEnabled = true  
myMap.uiSettings.isCompassEnabled = true  
myMap.uiSettings.isZoomControlsEnabled = true
```

Marker 1/2

```
val hungary = LatLng(47.0, 19.0)
myMap.addMarker(MarkerOptions().
    position(hungary).
    title("Marker in Hungary"))
myMap.moveCamera(CameraUpdateFactory.newLatLng(hungary))
```

Marker 2/2

```
val markerHU = myMap.addMarker(  
    MarkerOptions()  
        .position(hungary)  
        .title("Hungary")  
        .snippet("Population: 9.700.000")  
        .icon(BitmapDescriptorFactory.fromResource(  
            R.mipmap.ic_launcher_round)))  
markerHU.isDraggable = true
```

Marker control

- InfoWindow:
 - > Custom InfoWindow layout
 - > InfoWindowAdapter
 - > Event handler: OnInfoWindowClickListener
- Marker event handlers (for the whole map!):
 - > OnMarkerClickListener
 - > OnMarkerDragListener
 - > Stb.

Map events

```
mMap.setOnMapClickListener {
    val markerHU = mMap.addMarker(
        MarkerOptions()
            .position(it)
            .title("Hungary")
            .snippet("Population: 9.700.000"))
    markerHU.isDraggable = true

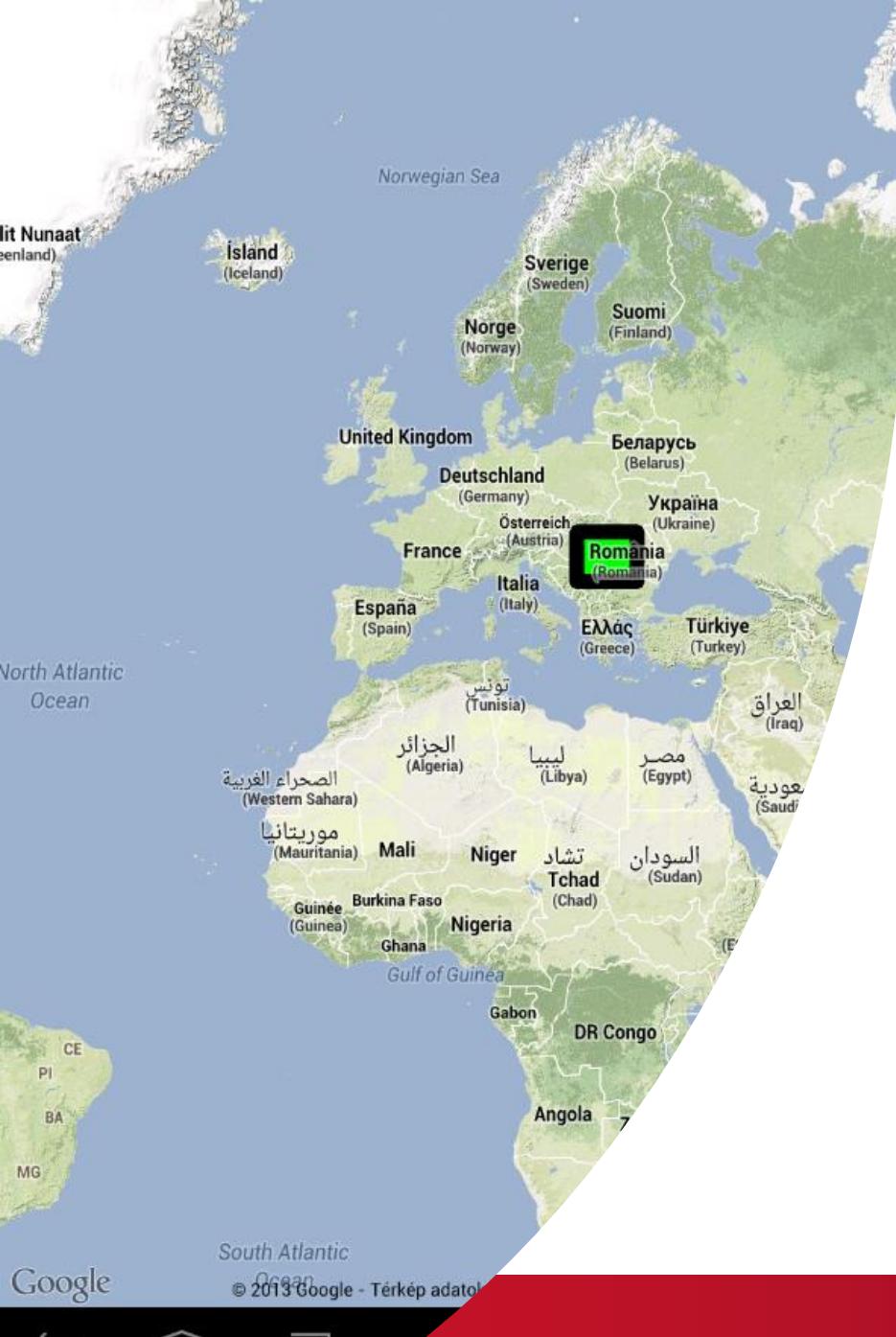
    mMap.animateCamera(CameraUpdateFactory.newLatLng(it))
}
```

Draw on the map

- Supported items:
 - > Polygon
 - > Polyline
 - > Circle
- Customized drawing:
 - > Line color
 - > Fill color
 - > Z-index
 - > Visibility
 - > Etc.

Draw rectangle

```
val polyRect: PolygonOptions = PolygonOptions().add(  
    LatLng(44.0, 19.0),  
    LatLng(44.0, 26.0),  
    LatLng(48.0, 26.0),  
    LatLng(48.0, 19.0))  
val polygon: Polygon = myMap.addPolygon(polyRect)  
polygon.fillColor = Color.GREEN
```

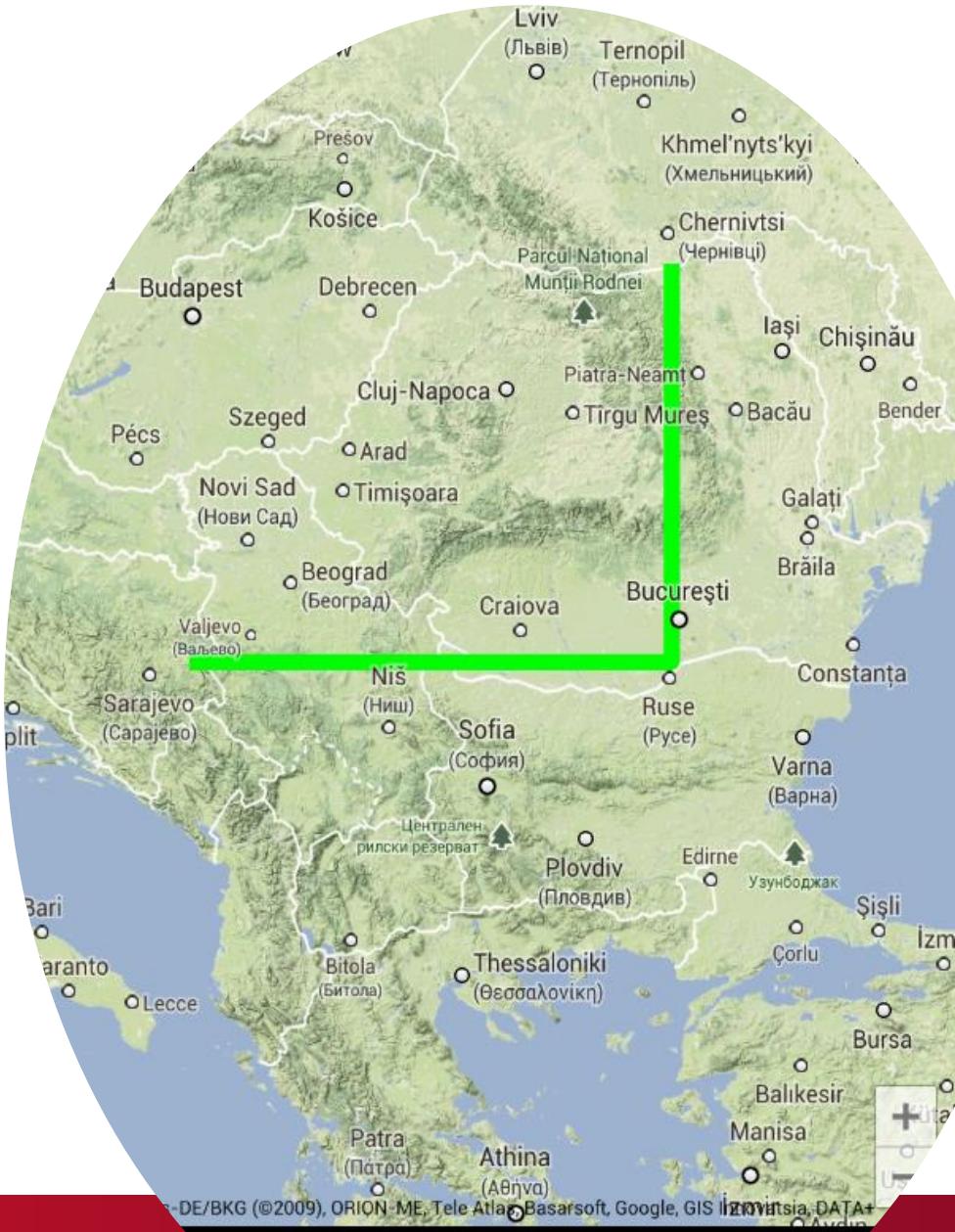


Demo – Draw rectangle

Draw line

```
val polyLineOpts = PolylineOptions().add(  
    LatLng(44.0, 19.0),  
    LatLng(44.0, 26.0),  
    LatLng(48.0, 26.0))  
val polyline = myMap.addPolyline(polyLineOpts)  
  
polyline.color = Color.GREEN
```

Demo – Draw line

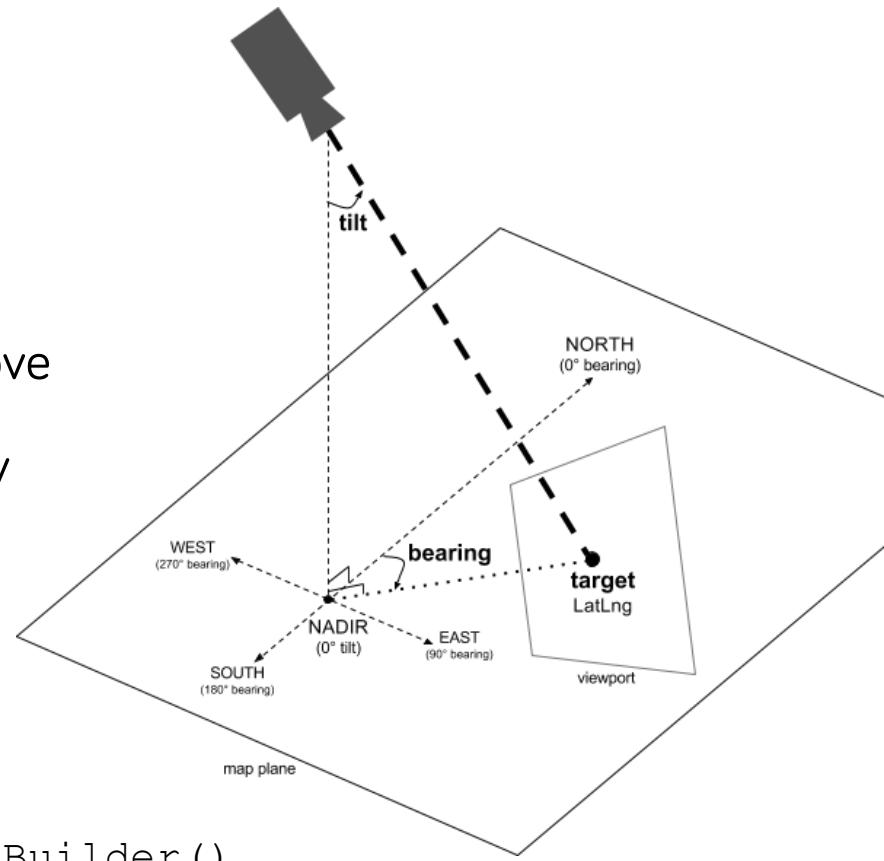


Example Map application



Camera position

- moveCamera (): move camera
- animateCamera (): animated move
- CameraPosition: set camera with a factory
 - > Target
 - > Zoom
 - > Bearing
 - > Tilt



```
val cameraPosition = CameraPosition.Builder()
    .target(LatLng(47.0, 19.0))
    .zoom(17f)
    .bearing(90f)
    .tilt(30f)
    .build()

myMap.animateCamera(CameraUpdateFactory.newCameraPosition(
    cameraPosition))
```

LOCATION INFORMATION

Location based services

- Mobile phones are portable
- Application can use location to provide extra information
 - > Navigation
 - > Meeting organization based on routines
 - > Enterprise applications: fleet tracking

Evolution of navigation services



Hyper-personalized GPS "directions"

Current location

- Supported techniques:
 - > GPS, Galileo, GLONASS
 - > Network: Android's Network Location Provider
- **GPS** accurate, but high energy consumption and works only outside
- **Network:**
 - > Cell based
 - > WiFi based
 - > Fast but not so accurate
- Both technique can be used at the same time

Current Position

- Location Manager
 - > Obtain current location
 - > Track movement
 - > Set proximity alerts for detecting movement into and out of a specified area
- Location Provider
 - > Based on WiFi and mobile cells
 - > Fine location: GPS
- Permissions
 - > *android.permission.ACCESS_FINE_LOCATION*
 - > *android.permission.ACCESS_COARSE_LOCATION*

Providers

- Example

```
var locationManager: LocationManager =  
getSystemService(Context.LOCATION_SERVICE) as LocationManager  
locationManager.getProvider(LocationManager.GPS_PROVIDER)
```

- Different capabilities
 - > power consumption, monetary cost, accuracy
 - > ability to determine altitude, speed or heading
- Criteria object for get the best provider
 - > accuracy criteria, power requirement criteria, altitude required, etc.

Android Location API 1/2

- New location arrives via an interface
- LocationManager system service:

```
var locationManager: LocationManager =  
getSystemService(Context.LOCATION_SERVICE) as LocationManager
```

- Continuous monitoring by implementing the LocationListener interface

```
// provider, mintime, mindistance, listener  
locationManager.requestLocationUpdates(  
    LocationManager.NETWORK_PROVIDER, 0, 0,  
    locationListener)
```

Android Location API 2/2

- `requestLocationUpdates()` parameter:
 - > Provider type (GPS or network)
 - > Minimum time between two refresh (0: fastest)
 - > Minimum distance between two refresh (0: shortest)
 - > LocationListener implementation

- LocationListener callback functions:

- > New location:

```
public void onLocationChanged(Location location)
```

- > Provider state change:

```
public void onStatusChanged(String provider, int status, Bundle extras)
```

- > Provider enabled:

```
public void onProviderEnabled(String provider)
```

- > Provider disabled:

```
public void onProviderDisabled(String provider)
```

LocationListener example

```
var locationListener: LocationListener = object : LocationListener {  
    override fun onLocationChanged(location: Location?) {  
    }  
  
    override fun onStatusChanged(provider: String?,  
        status: Int, extras: Bundle?) {  
    }  
  
    override fun onProviderEnabled(provider: String?) {  
    }  
  
    override fun onProviderDisabled(provider: String?) {  
    }  
}
```

Unsubscribe

- `removeUpdates (...)` to all created `LocationListener` implementation
- **DO NOT FORGET!!!**

```
locationManager.removeUpdates (  
    locationListener) ;
```

Location permission

- Required permissions:

```
<manifest ... >  
    <uses-permission android:name=  
        "android.permission.ACCESS_FINE_LOCATION" />  
    ...  
</manifest>
```

- ACCESS_COARSE_LOCATION: only network
- ACCESS_FINE_LOCATION: GPS and network

FusedLocationProvider

- Applications and services can share location info with each other
- Google Play Services dependency:

```
implementation 'com.google.android.gms:play-services-location:16.0.0'
```

- FusedLocationProviderClient

```
private val fusedLocationClient: FusedLocationProviderClient =  
    LocationServices.getFusedLocationProviderClient(context)
```

FusedLocation start and stop

```
fun startLocationMonitoring() {
    val locationRequest = LocationRequest()
    locationRequest.interval = 1000
    locationRequest.fastestInterval = 500
    locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY

    fusedLocationClient.requestLocationUpdates(locationRequest,
        locationCallback, Looper.myLooper())
}

fun stopLocationMonitoring() {
    fusedLocationClient.removeLocationUpdates(locationCallback)
}

private var locationCallback: LocationCallback = object : LocationCallback() {
    override fun onLocationResult(locationResult: LocationResult) {
        super.onLocationResult(locationResult)
    }
}
```

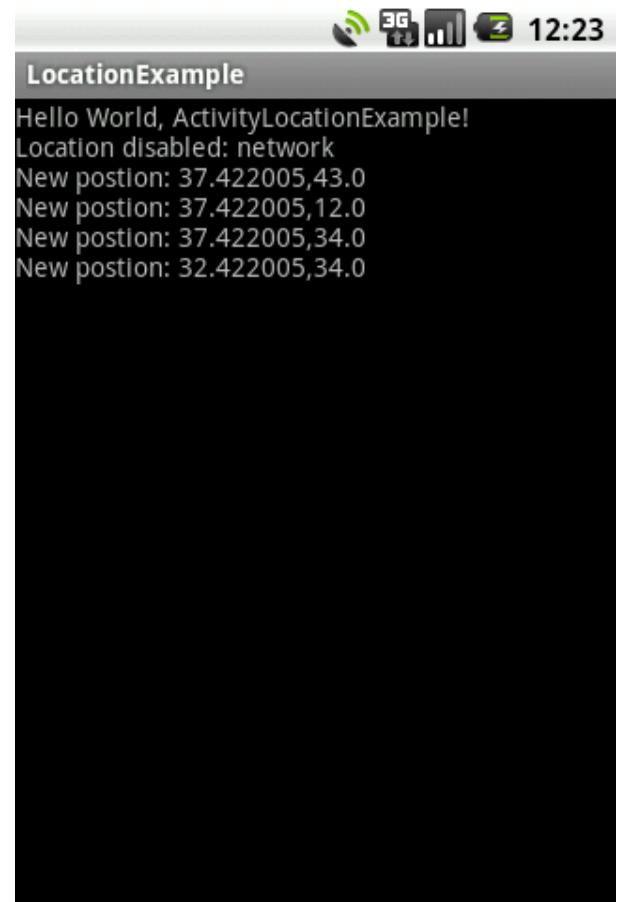
Checklist

- Check whether the Location object is new enough!
- Check the accuracy of the new Location
- Validate the new Location compared to the previous one by using the possible speed
- Check the provider of the new Location object
- In typical application it is not need to check the position in less than 60 seconds
- Typically we do not need to use the network and GPS provider at the same time
- You can use Kalman-filter to calculate more accurate position



Let's practice!

- Create an application that displays our current location!
- Extra:
 - > Show altitude and speed information



Engagement

- Extend the application with a distance counter that displays the sum distance in meters you have moved since the application has started!
- How to calculate distance between two location?

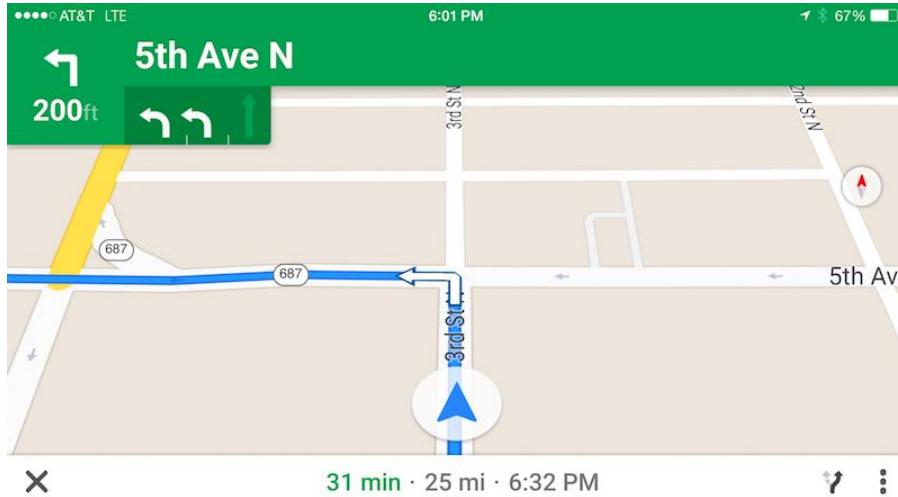
Location loc1 = ...

Location loc2 = ...

float distanceInMeters = loc1.distanceTo(loc2);

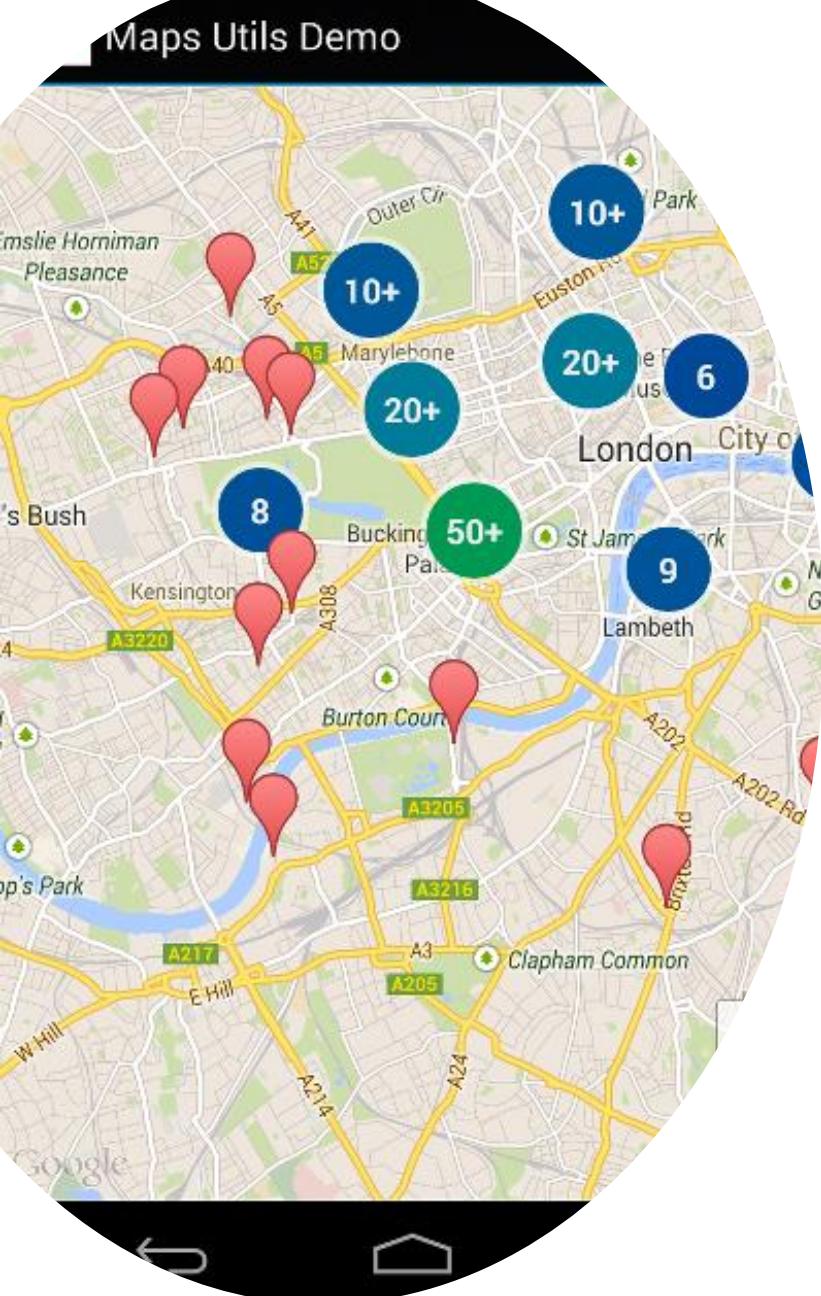
Navigation-like view

- How to implement a navigation like view?



- Monitor the position of the phone
 - The new *Location* object has a *bearing* value, set it as *bearing* for the map's *CameraPosition*

LOCATION BASED SERVICES



<https://developers.google.com/maps/documentation/android-api/utility/?hl=en>

Map Utility Library

Geocoding

- Address to GPS coordinate
- Internet permission is required

```
val geocoder = Geocoder(this, Locale.ENGLISH)
val streetAddress = "Blaha Lujza tér 1, Budapest"
var locations: List<Address>? = null
geocoder.getFromLocationName(streetAddress, 3)
```

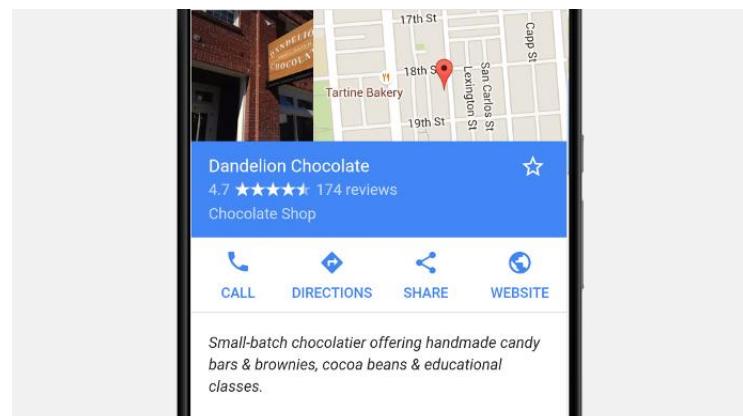
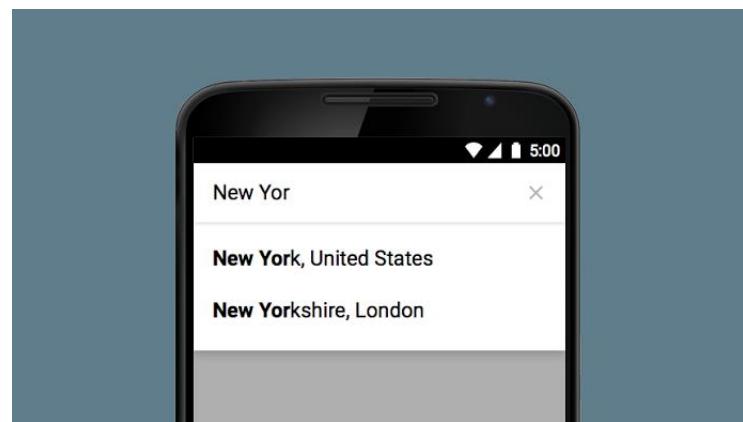
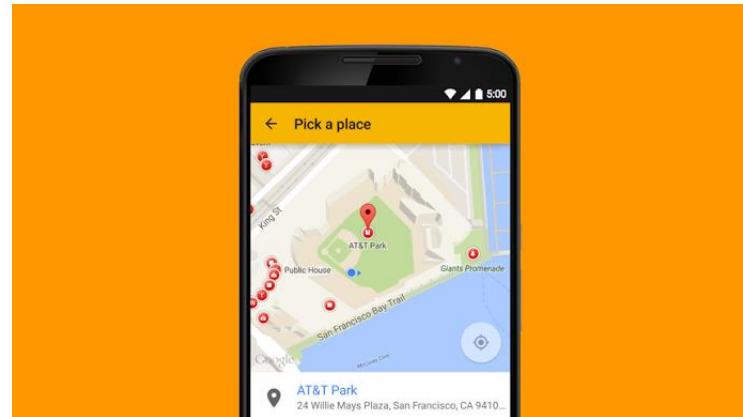
Reverse Geocoding

- Address from GPS coordinates
- Internet permission is required

```
val location = ...
val latitude = location.getLatitude()
val longitude = location.getLongitude()
val gc = Geocoder(this, Locale.getDefault())
var addrs: List<Address>? =
    gc.getFromLocation(latitude, longitude, 3)
```

Places API

- PlacePicker
 - The Place Picker UI is a dialog that allows a user to pick a Place using an interactive map.
- GeoDataApi
 - The Geo Data API provides access to Google's database of local place and business information.
- PlaceDetectionApi
 - The Place Detection API provides quick access to the device's current place, and offers the opportunity to report the location of the device at a particular place (like a check in).
- AutoComplete
 - UI widget that auto complete place names
- More info:
 - <https://developers.google.com/places/android-api/>



PlacePicker Dialog demo

- build.gradle
 - > implementation 'com.google.android.gms:play-services-location:16.0.0'
 - > Implementation ,com.google.android.gms:play-services-places:16.0.0'
 - <https://developers.google.com/android/guides/setup>
- Manifest:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY" />
<meta-data android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

-
 - > <https://developers.google.com/places/android-sdk/signup>

PlacePicker Dialog demo

- Connect to Google API client

```
googleApiClient = GoogleApiClient.Builder(this)
    .addApi(Places.GEO_DATA_API)
    .addApi(Places.PLACE_DETECTION_API)
    .enableAutoManage(this, this)
    .build()
```

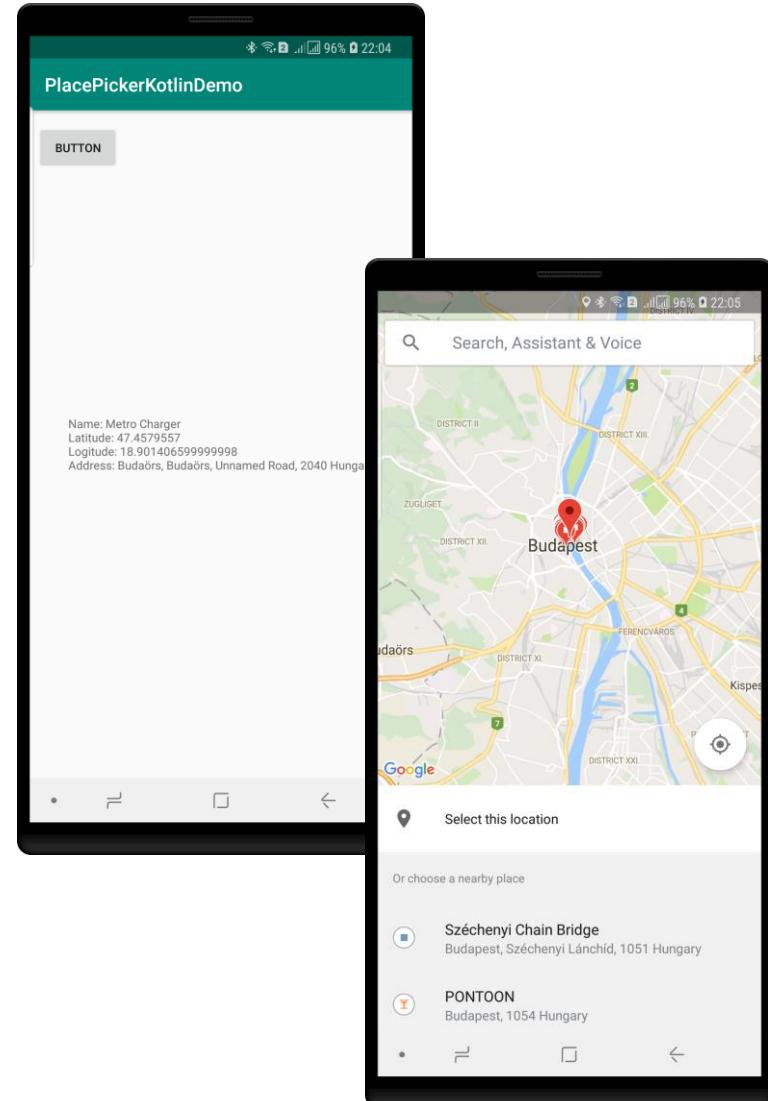
- PlacePicker launch

```
val builder = PlacePicker.IntentBuilder()
startActivityForResult(builder.build(this@MainActivity), PLACE_PICKER_REQUEST)
```

- Handle result in onActivityResult(...)

PlacePicker Dialog demo

- Show selected place info in the Activity



ProximityAlert

- Android can notify when mobile move into or out a specified area
 - > Coordinate and radius
- Intelligent logic for using the correct provider

```
Intent intent = new Intent(ACTION_PROXIMITY_ALERT);
PendingIntent pi = PendingIntent.getBroadcast(this, requestCode, intent, flags);
locationManager.addProximityAlert(lat, long, radius, timeout, pi);
...
public class ProximityIntentReceiver extends BroadcastReceiver {
    @Override
    public void onReceive (Context context, Intent intent) {
        String key = LocationManager.KEY_PROXIMITY_ENTERING;
        Boolean entering = intent.getBooleanExtra(key, false);
        ...
    }
}
```

New Geo APIs

- Implemented in Google Play Services
- New APIs:
 - > Fused location provider:
 - <https://developer.android.com/training/location/retrieve-current.html>
 - > Geofencing APIs
 - <https://developer.android.com/training/location/geofencing>
 - > Activity recognition
 - <https://developers.google.com/location-context/activity-recognition/>

Homework – Readings

- Maps Android API
 - > <https://developers.google.com/maps/documentation/android-api/>
- Android Location Fused Provider
 - > <http://javapapers.com/android/android-location-fused-provider/>
- Displaying a Location Address
 - > <https://developer.android.com/training/location/display-address.html>
- Location based services overview
 - > <https://developer.android.com/training/location>



Questions?

