Android Schulung am KIT

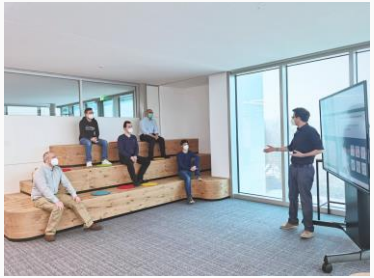
arconsis IT-Solutions GmbH

felix.schmid@arconsis.com
patrick.jung@arconsis.com

Introduction

About arconsis

- Software house and technology consulting company
  - „Our aim is to develop the system the customer wished he had asked for“
  - „Technological excellence and passion“
- Mobile Enterprise, Adaptive Enterprise, Enterprise AI
- 4 locations, 38 employees, 9 nationalities
- Opportunites for students:
  Theses, internship semester, student traineeships



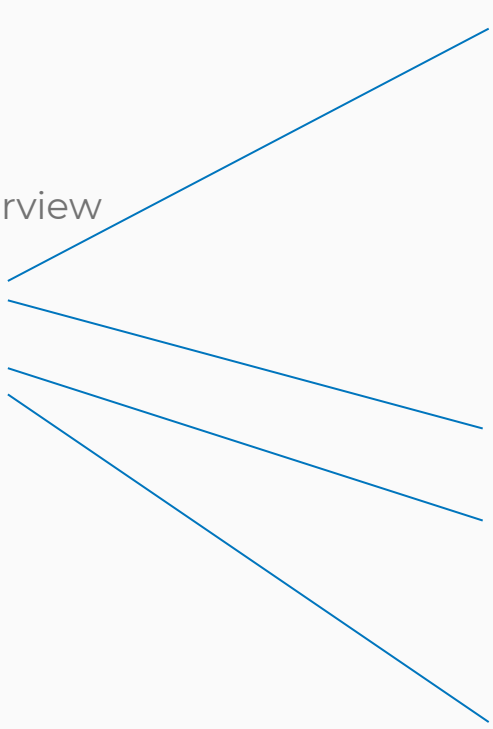
www.arconsis.com

Helpful links

- Android courses with Kotlin
- Google Developer Guides
- Modern Android Development (MAD) Skills
- Android course at Udacity by Google
- Now in Android: Demonstrates best practices

Overview

- Introduction
- Setup
- Kotlin
- Architecture Overview
- UI
- Architecture
- Testing



- Activity
- UI (compose)
- UI control elements
- UI Layout
- State in Compose
- Resources
- Lists
- Navigation
- Dialogs

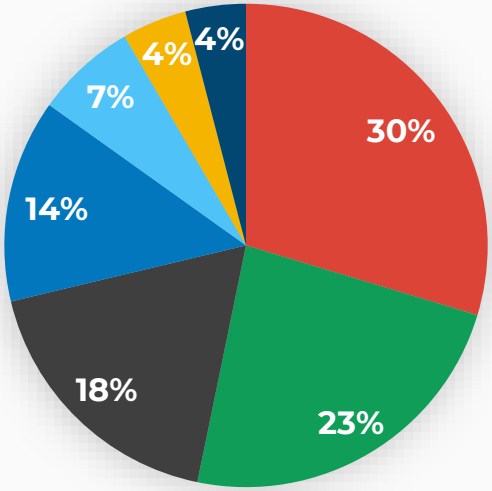
- ViewModel
- Dependency Injection
- Data persistence
- Coroutines
- Network

What is Android?

- OS for mobile and embedded devices
- Initially created by Android Inc. (bought by Google in 2005)
- OS itself is open source (Android Open Source Project)
  - Not including low level drivers and proprietary software, e.g. some Google Services
- Maintained by Open Handset Alliance
  - Association of tech companies for the development of open standards for mobile systems (Google, Sony, HTC, Samsung, T Mobile, Qualcomm, ...)

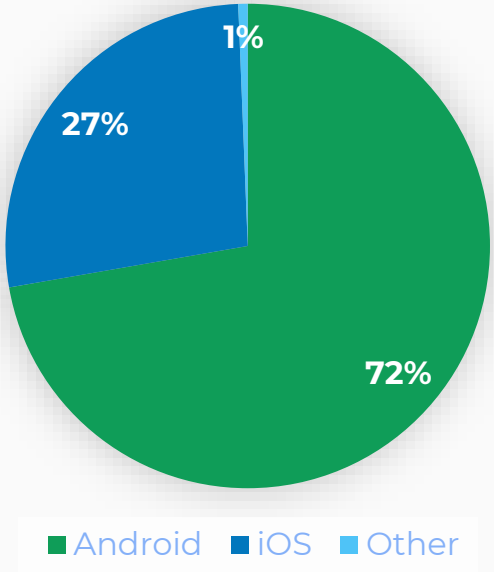
Distribution of mobile Operating Systems

OS-Distribution 12.2010



Source

OS-Distribution 02.2023



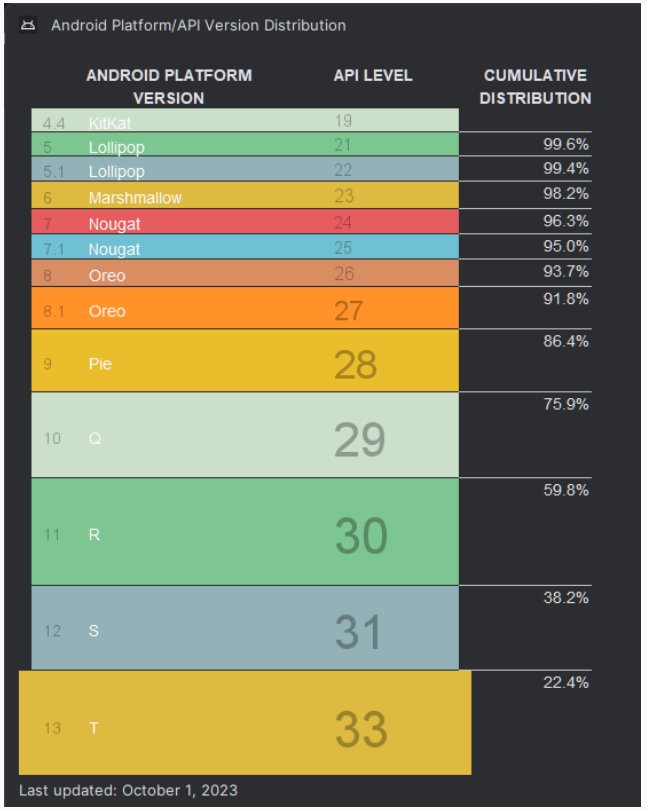
Source

Structure

- Is based on Linux Kernel
- Is built as a layered architecture
  - Upper layer can access lower layer
- Kotlin as programming language
  - Was Java until 2019
- The lecture will focus on the layer Java API Framework

Fragmentation

- Problems
  - Update process is complex
  - Device manufacturer often add changes to AOSP code
- Consequences
  - Devices don’t get much updates
  - Developers need to develop for a lot of different OS versions and configurations
- Solutions
  - Publish security patches separately
  - Treble
  - Android One
  - Outsource code to external libraries



Setup

Environment

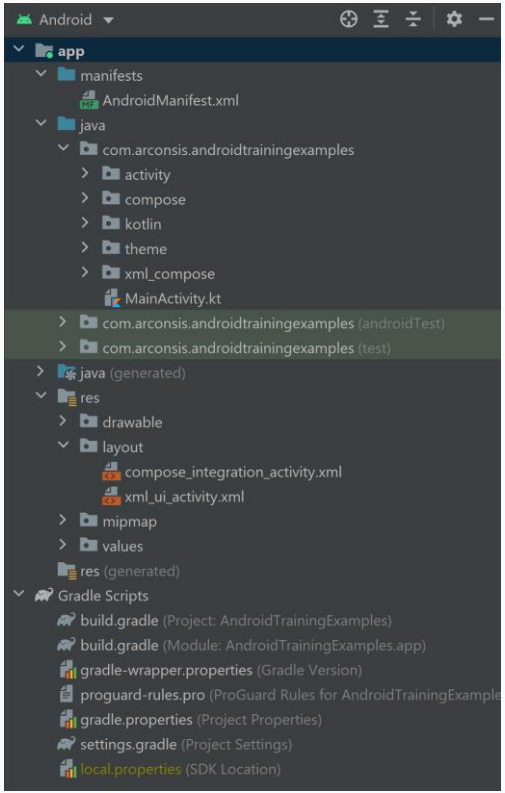
- Android Studio IDE
- Android SDK
- adb command line tool
- Android Emulator

Android Studio

- Based on Intellij Idea Platform
- Uses gradle based Build-System
- Uses emulators to test on a variety of devices
- You can also use scrcpy to mirror a real device on your laptop
- Overview
- Install guide

File structure of a project

- app
  - Manifest
    - Basic app information
  - Source code & tests
  - Ressources
    - Images
    - Strings
- gradle
  - Build tool



Gradle

- Build automation tool (like Maven)
- Uses Groovy DSL or Kotlin DSL for build scripts
- Extended with Android Gradle Plugin
- Project can be build everywhere (IDE, console, CI)

Overview

- Gradle consists of tasks and settings
  - E.g. *./gradlew assembleDebug* which builds an apk from your code
  - E.g. *./gradlew tasks* which lists all available gradle tasks for this project
- We can add plugins to enrich gradle, e.g. with the android gradle plugin (AGP), to be able to set android related settings
- Gradle-wrapper: Each project can use ist own gradle version. To ensure that every developer uses the same one without manually installing it, the wrapper takes care of that.

Common gradle files: libs.version.toml (catalog)

- Define libraries and plugins at one common place.
- They can then be used in all submodules without copy&paste them.
- Needs to be selected when creating a new Android project.
- Can then be used in gradle files like
  - libs.core.ktx
  - libs.plugins.com.android.application
  - libs.version.agp.version

```
# Version variables to use for the libraries.
[versions]
agp-version = "8.1.1"
core-ktx = "1.12.0"
...

# Here we add all the libraries/dependencies that we need for our app
# A library is identified by group-id-name:library_id_name:version, e.g.
# androidx.core:core-ktx:1.12.0
https://androidx.tech/artifacts/core/core-ktx/
[libraries]
core-ktx = { group = "androidx.core", name = "core-ktx", version.ref =
"core-ktx" }
...

[plugins]
com-android-application = { id = "com.android.application",
version.ref = "agp-version" }
...
```

Common gradle files: build.gradle.kts (project level)

- Defines which gradle plugins to use for the whole project. These plugins can then be used in the modules of the project.

```
// Top-level build file where you can add configuration options
   common to all sub-projects/modules.
// These are plugins for gradle. They extend gradle with new tasks.
// We can define them here, then every module can use them.
plugins {
   // Here we add the android gradle plugin (AGP)
   alias(libs.plugins.com.android.application) apply false
   alias(libs.plugins.org.jetbrains.kotlin.android) apply false
   alias(libs.plugins.hilt.android) apply false
   alias(libs.plugins.kotlin.kapt) apply false
}
```

Common gradle files: build.gradle.kts (module level)

- Additionally to the project level one, every module has additionally ist own build.gradle.kts file
- Here we define most of the settings for the specific module
  - Gradle plugins to use
  - Compile-, min-, targetSdk
  - Libraries/Dependencies to use
  - Build types
  - Product flavours
  - Build variants

```
plugins {
    alias(libs.plugins.com.android.application)
    ...
}

    // Android related settings. We can use this, because
    // the module use the Android Gradle plugin
android {
    ...
}

// Here you can add third party libraries.
// We could add them directly like the first one or
// reference them from our version catalog like all the others.
// (which in the end does the same, it's just more maintainable).
dependencies {
    // ViewModel
    // Here we directly enter the id of the library.
    implementation("androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.2")
    // Here we use our version catalog.
    implementation(libs.lifecycle.viewmodel.ktx)
    ...
}
```

Common gradle files: settings.gradle.kts

- Describes from which repositories we can get plugins and libraries from
- Defines which modules should be part of the project.

```
pluginManagement {
    repositories {
        google()
        mavenCentral()
        gradlePluginPortal()
    }
}
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        maven { setUrl("https://jitpack.io") }
        google()
        mavenCentral()
    }
}
rootProject.name = "AndroidTrainingExamples"
include(":app")
include(":weatherapp")
include(":movieapp")
```

Other gradle files

- gradle.properties: Project wide settings, like maximum heap size
- local.properties: Local settings like path to the sdk directory or API-Keys

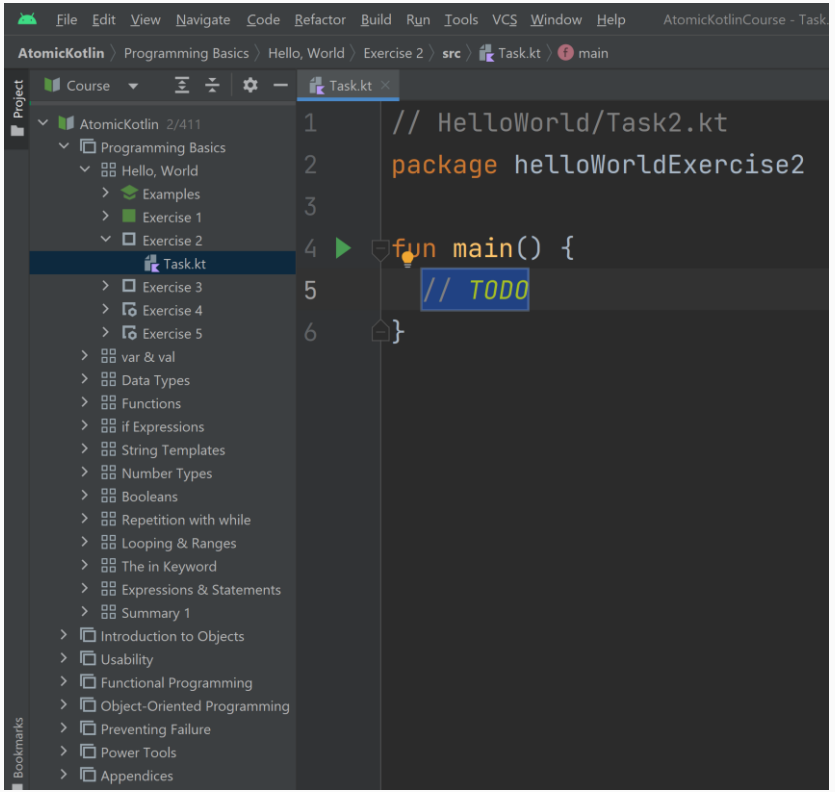
Kotlin

Kotlin

- Modern, pragmatic, robust
- Object oriented with functional programming elements
- Null safety
- Statically typed with type inference
- Precise, less boilerplate code than Java
- Intro course

Learn Kotlin

- To learn Kotlin you can use the course *AtomicKotlin* by *JetBrains Academy*.
- Go to **File** -> **Settings** -> **Plugins** and install *JetBrains Academy* from the Marketplace
- After that go to **File** -> **Learn and Teach** -> **Browse courses** and choose *AtomicKotlin* in the Marketplace



Basic variable syntax

- Statically typed
- Typeinference, explicit types for expressions can be omitted
- Variable declaration [var|val] varName: Type

```
// in Java: int a = 1;
val a: Int = 1

val b = 2

// val not reassignable but not really immutable
//b = 3

var c = 3

c = 5
```