Course in (Jetpack Compose) Kotlin on Android. Android OS. Android Studio HelloWorldApp

Dmitry Makarenkov

https://dmpsy.club

JetpackCompose <= (Kotlin && Android)

Target Audience

- Kotlin enthusiasts who target their applications to Android OS following modern Jetpack Compose UI principles

Mind the story:

Andy Rubin (the Founding Father of Android) was in love with robots to the point that his friends called him Android.

(see https://en.wikipedia.org/wiki/Andy_Rubin)

Mobile App R&D Big Picture

Manufacturers	Apple	Samsung, Huawei, Google, Vivo, Xiaomi *etc.*	Huawei
Operating system	iOS	Android OS	HarmonyOS
Installer files	*.dmg (installer) *.ipa (archive)	*.apk (installer), *.aab (bundles)	*.hap, *.app
Preferred IDE	Xcode	Android Studio	Dev Eco Studio
Native Programming Languages	Swift Objective C	Kotlin (JetPack Compose) Java (not recommended by Google Play since 2019)	Java, eTS ArcTS JS
Alternative languages	React Native (JS) KMM (Kolin Multiplatform Mobile)	React Native (JS)	Kotlin (?) React Native (?)

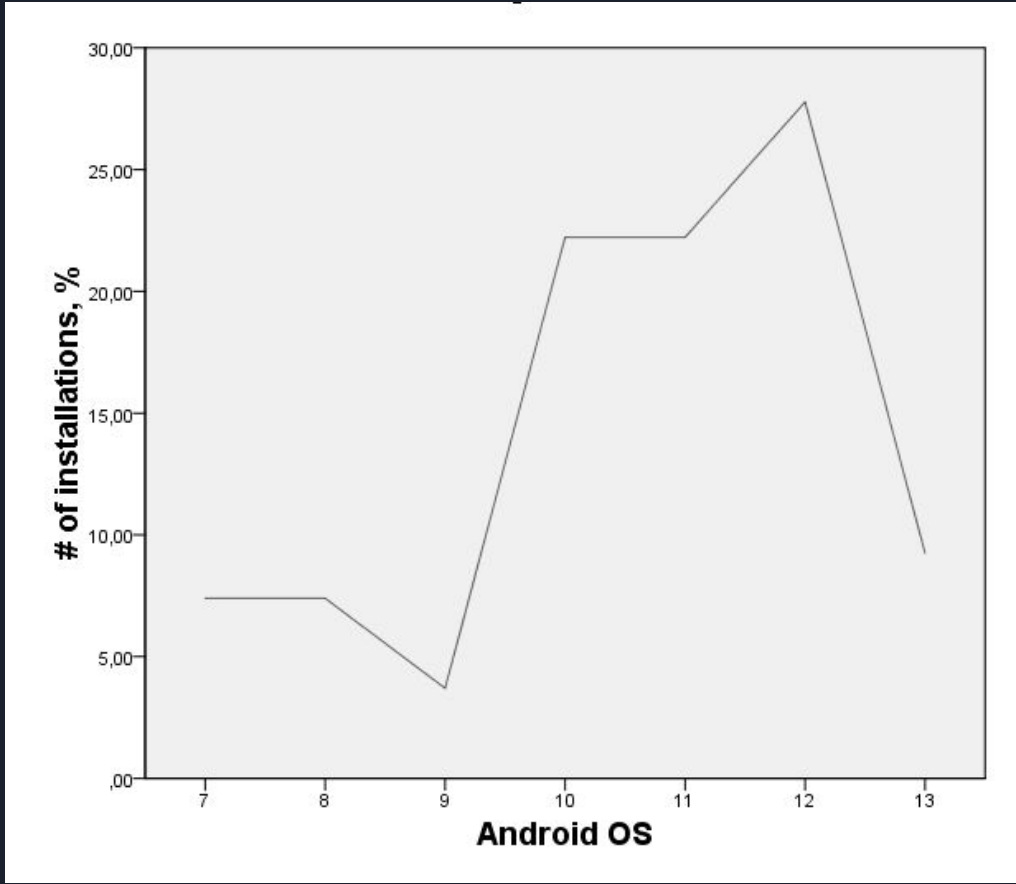
Android OS Intro

- Android OS 1st release took place in 2008
- Android is Linux-based open source OS owned by Google
- Apps are running on top of the (Java Virtual Machine) JVM by Google
- Java is eventually replaced by Kotlin, the “preferred” language since 2019
- Google provides (Software Development Kits) SDKs for Android app development for each unique (Application Programming Interface) API level
- For convenient development, use the latest Android Studio IDE, the most popular IDE based on IntelliJ IDEA of JetBrains
- Android Studio output file formats: (*.apk, installers) or (*.aab, bundles)
- Key App Marketplaces: Google Play, AppGallery (Huawei), RuStore (Russia)

In-Use Android OS Versions & SDKs

Codename	Version	API Level
Android 13, Tiramisu	13	33
Android 12L, Snow Cone	12.1	32
Android 12, Snow Cone	12	31
Android 11, R, Red Velvet Cake	11	30
Android 10, Q, Quince Tart	10	29
Pie	9	28
Oreo	8.1.0	27
	8.0.0	26
Nougat	7.1	25
	7.0	24
Marshmallow	6.0	23
Lollipop	5.1	22
	5.0	21

Statistics of Eysenck App Downloads from RuStore



Key Android App Parameters

- Language: Kotiln (as a preferred language since 2019)
- Output file formats : normally, *.apk as not all the stores accept bundles and you cannot run *.aab on real devices or emulators
- Mind that the release version of the output file must be signed
- minSdk: 21 as defaulted by Android Studio (the OS allows installing your app downto this Android OS version, otherwise the system denies to install it on that particular device)
- targetSdk: 32 (default) or 33 (the latest API level)
- As of August, 2021, apps must target Android 11 (API level 30) or higher
- versionCode of your app, must be unique integer starting with 1
- versionName "1.0" is a string name shown to the end-User in the store

Build App Parameters Inside Android Studio Project

```
android {
    signingConfigs {
        debug {
            storeFile file('K:\\KeyStore\\TestKey.jks')
            storePassword 'TestPassword'
            keyAlias 'keyTest'
            keyPassword 'TestPassword'
        }
    }
    namespace 'club.dmpsy.hellocompose'
    compileSdk 33

    defaultConfig {
        applicationId "club.dmpsy.hellocompose"
        minSdk 21
        targetSdk 33
        versionCode 1
        versionName "1.0"
```

build.gradle(app) & build.gradle(Project) contain all build parameters

Possible build variants:

1. ***Unsigned debug***
2. ***Unsigned release***
3. ***Signed debug***
4. ***Signed release***

Build App Parameters Inside Android Studio Project

```
buildscript {
    ext {
        compose_version = '1.3.2'
    }
}// Top-level build file where you can add configuration options com
plugins {
    id 'com.android.application' version '7.3.1' apply false
    id 'com.android.library' version '7.3.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.6.10' apply false
}
```

```
dependencies {

    implementation 'androidx.core:core-ktx:1.9.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.5.1'
    implementation 'androidx.activity:activity-compose:1.6.1'
    implementation "androidx.compose.ui:ui:$compose_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_version"
    implementation 'androidx.compose.material3:material3:1.1.0-alpha03'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.4'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_version"
    debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_version"
}
```

…and configurable dependencies

of your project

Launch App Parameters Inside Android Studio Project

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="HelloCompose"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.HelloCompose"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:label="HelloCompose"
        android:theme="@style/Theme.HelloCompose">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
```

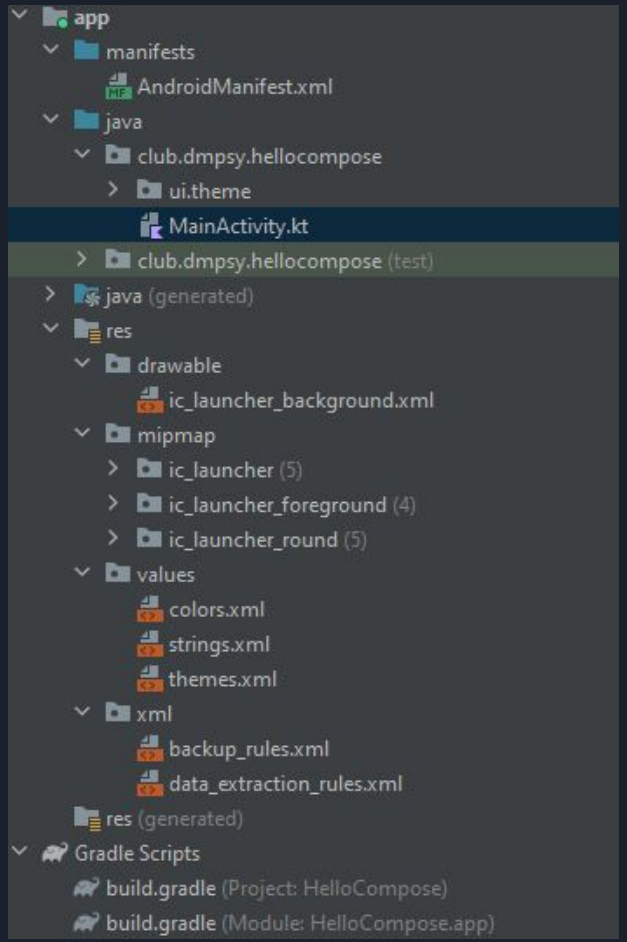
Manifest.xml contains main launch information:

the app icon,

the app name,

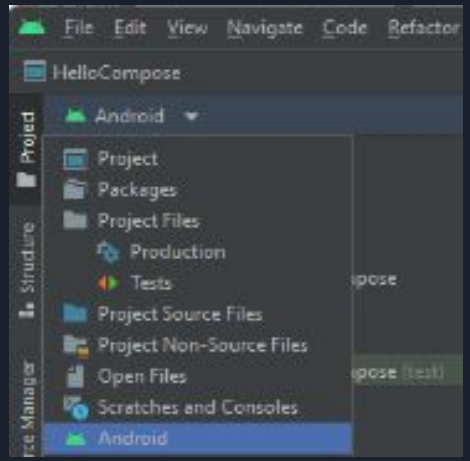
the entry point, i.e., the first module to launch on starting the app (MainActivity)

App Tree of The Android Studio Project



Finally, the "app" project tree contains:

1. *Manifest.xml, the main launch file*
2. *MainActivity.kt , the main entry Kotlin source file*
3. *The res folder with drawables, strings and other resources like Material Design colors, theming etc.*
4. *Gradle (Project & Module) build scripts*



P.S. You may want to use other tree representations as well (see left)

MainActivity class, the entry point of the app

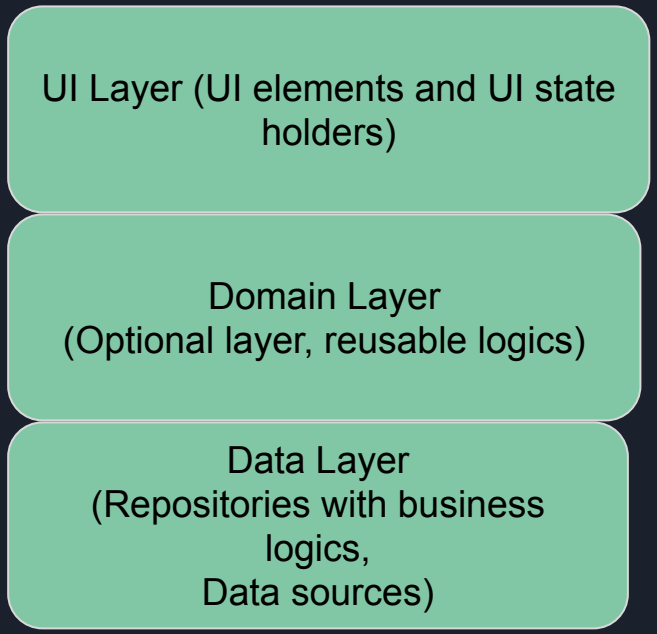
```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            HelloComposeTheme {
                // A surface container using the 'background' color from the th
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    Greeting( name: "Android")
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String) {
```

MainAcitivy.kt contains the main app class MainActivity that:

1. *Sets the (UI) app content, Material Design Theme*
2. *Defines the basic Surface*
3. *Calls the main UI function annotated with @Compose*

Let's first discuss the app logical structure and then get back to the "composables" again

Three-Layered Architecture of the Application



Focus on the UI Layer:

- ***Jetpack Compose*** is the new toolkit for building native Android layouts.
- It was launched in 2021 with the aim to further simplify and accelerate the UI development for Android apps.

Jetpack Compose vs. Classic MDC approach

Jetpack Compose replaces:

1. Xml-based layouts (visual representation of UI components) separated from
2. the Kotlin code programming the UI components behavior

with:

1. The single representation of the UI elements with “Composable functions” or “composables”

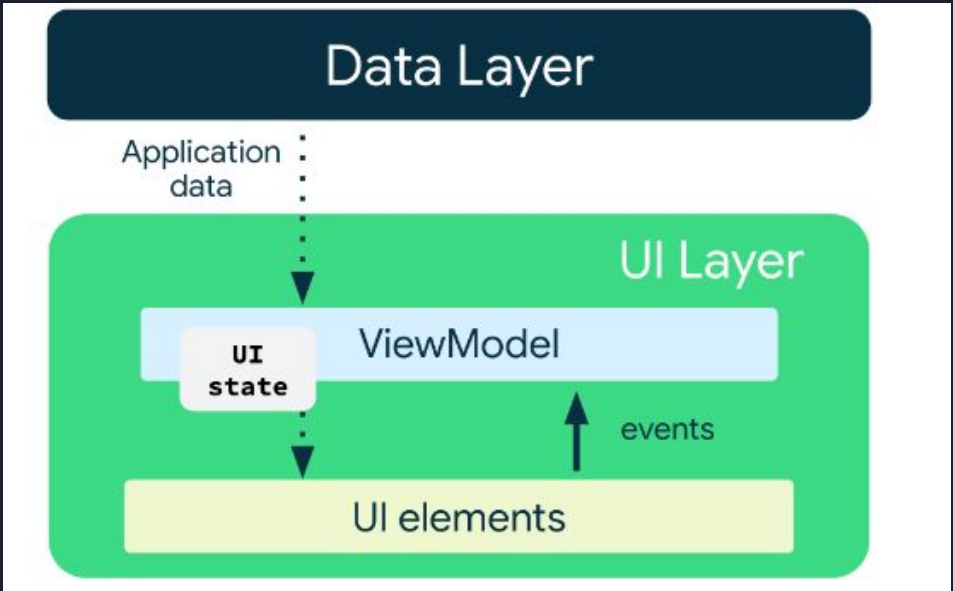
and implements:

2. The UDF (Unidirectional Data Flow) approach (“Events up”, “States down”):

UI elements send their Events to the upper layer

UI elements change their appearance based on the UI States received from the upper layer

Unidirectional Data Flow (UDF)

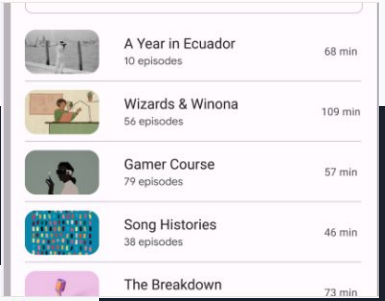


- Events go up to the upper ViewModel layer
- ViewModel changes the UI states on Events and/or Application data received from the Data Layer
- ViewModel send the updated UI states down to UI element to change their appearance

(see https://developer.android.com/topic/architecture/ui-layer)

Layout & Code Replaced by @Composable

```
<com.google.android.material.divider.MaterialDivider
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:dividerInsetStart="16dp"
    app:dividerInsetEnd="16dp"/>
```




```
@Composable
fun Divider(
    modifier: Modifier = Modifier,
    thickness: Dp = DividerDefaults.Thickness,
    color: Color = DividerDefaults.color
): Unit
```

```
divider.setDividerInsetStart(insetStart)
divider.setDividerInsetEnd(insetEnd)
```

(see https://m3.material.io/components/divider/overview)

The Composable Function:

1. Must be annotated with @Composable
2. Must be annotated with @Preview to be viewed in the Android Studio Previewer
3. The composable name is in PascalCase opposite to ordinary Kotlin functions (camelCase)
4. The composable functions return no value
5. One can define and re-use his/her own composable

```
@Composable
fun Divider(
    modifier: Modifier = Modifier,
    thickness: Dp = DividerDefaults.Thickness,
    color: Color = DividerDefaults.color
): Unit
```

Basic Building Blocks of Jetpack Compose:

1. The “container” elements, invisible but mastering the layout carcass:

   Row, Column, Box, Scaffold etc.

2. The Material 3 Design Components, which make the real contents and implement interactions with the app end-user:

   Button, Card, Chip, Dialog, TopAppBar ***etc.***,

   (see https://m3.material.io/components)

The Practice Session Plan:

1. Download and install the latest Android Studio from https://developer.android.com/studio
2. Create a new Kotlin Empty Compose Activity (Material 3) Project
3. Review the project structure
4. Build and run in the emulator the unsigned debug project version
5. Find in app/build/outputs/apk/debug the *.apk file and deploy it to BlueStacks
6. Build the unsigned release *.apk file and try to deploy it to BlueStacks. Should deny installation
7. Upgrade your application to the latest targetSdk = 33, upgrade the dependencies
8. Upgrade the app versionCode = 2 and versionName = “1.1”
9. Change the app icon using Resource manager “+” “Image Asset”
10. Add more functionality to the app (Row/Column/Box/Button/ user-defined Button Block)
11. Rebuild the app, verify it works on the Emulator
12. Sign the app, generate the final signed *apk file (app/release/ *.apk) and upload it to BlueStacks

Next Videos to Follow

- Lecture 1 Practice Session

PS Download the present lecture from:
https://dmpsy.club/references/Kotlin/lesson_001_helloWorld.pdf

PPS I will translate this lecture into Russian if necessary, just let me know, please.

https://dmpsy.club
postmaster@dmspy.club