UA/DETI – Computação Móvel

Ilídio C Oliveira (ico@ua.pt)

Last revision: v2024-11-05 **work in progress**

CM Weekly Activities – Android module

Introduction

Classes plan

See the overall course schedule, in Moodle.

Learning resources

Selected Android learning resources:

- Official Android developers' documentation: tutorials, API documentation, tools, best practices,…
- Main resources for CM classes:
    - "Android Basics with Compose: course"
    - "Android Development with Kotlin: Classroom course" (with lecture slides)
- Other Android training resources:
    - Kotlin course by JetBrains

- o   Android [Courses by Google](#) with codelabs.
- o   YouTube: [Android Developers](#) channel; [Philipp Lackner](#) channel.
- Books
  - o   Access these books at O'Reilly → [CM_playlist](#)
    - ▪   "[Programming Android with Kotlin](#)" (2021) by P. Lawrence et al. → [PAwK]
    - ▪   "[Android Programming](#): The Big Nerd Ranch Guide, 5th Edition" (2022)
    - ▪   "[Professional Android](#)" (2018) by Reto Meier

Week #1- Introduction to Android development workflow and tools

A) Kotlin basics

Concepts & readings:

Kotlin concepts:
- Slides from lesson 1 [Kotlin Basics](#) → Lesson 2 [Functions](#) → Lesson 3 [Classes and Objects](#)
- Kotlin essentials (for developers): [PAwK→chap #1](#)

See also:
- Kotlin [style guide](#)
- Kotlin [online playground](#) (run code in the browser)

Hands-on exercises (with code labs):

- ☐   Take the [codelabs for Unit #1 - Introduction to Kotlin](#) as needed. [suggested: activity #6]

B) Introduction to Android native apps development

Android fundamentals :
- The [Android stack](#)
- Build [UI with (Jetpack) Compose](#) (short tutorial)

See also:
- [API levels](#)

Hands-on exercises & code labs

- ☐   Unit 1 > "[Setup Android Studio](#)" [if needed]. Includes: "how to connect you [physical] Android device".

- ☐   Unit 1 > "[Build a basic layout](#)" [activities #3 and #4. Suggested for homework → #5]

- ☐   Unit 2 > "[Kotlin fundamentals](#)" [activities #3, #4 and #5]

- ☐   Unit 2 > "[Add a button to an app](#)" [activities #2 and #3. Suggested for homework → #4]

Week #2: Composable and UI state

C) UI, events and state managemnt

Resources and readings:

- [Thinking in Compose](#): theory and principles
- Guide to [Android App architecture](#)
- [Unidirectional data flow](#) (UDF) pattern

Hands-on exercises & code labs

☞ [Optional] Hands-on: Unit 2 > [Interacting with UI and State: Tip calculator](#)[Activities → #3, #4]. Focus on "state hoisting"

☞ Hands-on: Unit 4 > "[Architeture Components](#)" [Activities → #2, #5] Focus: UI state and ViewModel.

☞ [Optional] Scrollable list: Unit 3 > "[Build a scrollable list](#)" [Activities → #2].

Activity lifecycle
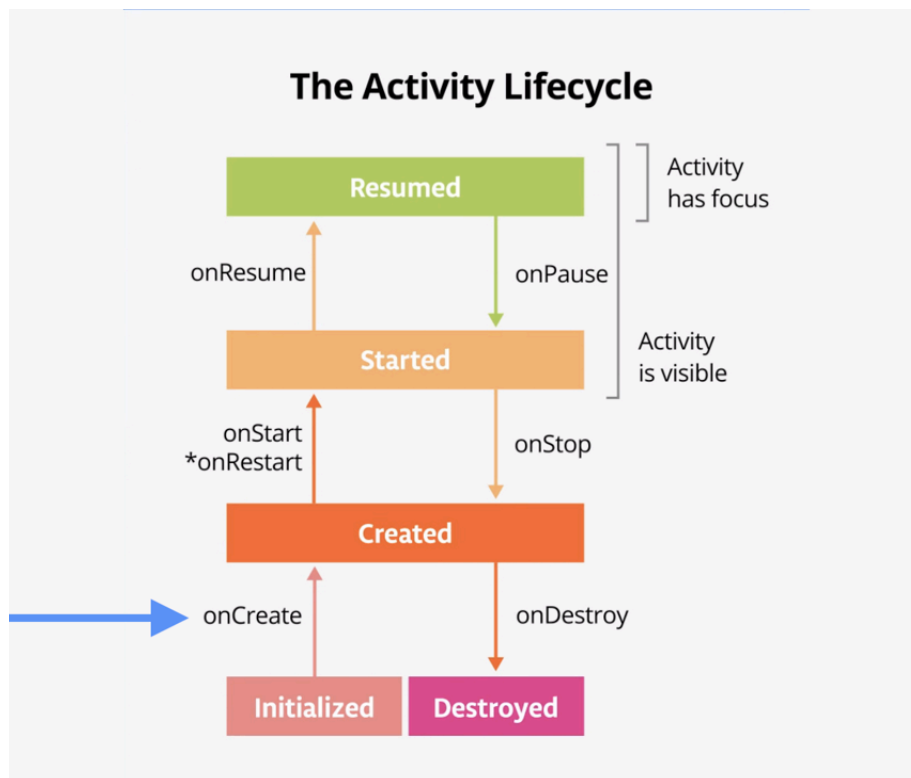
UI State and software design patterns
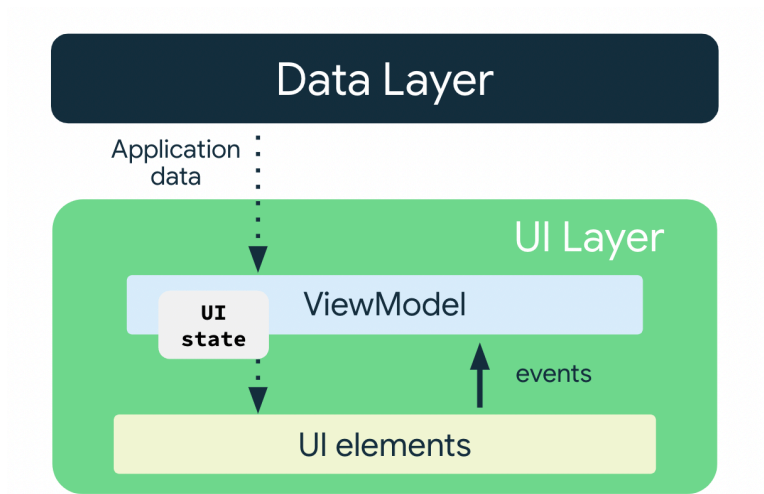
The UI layer is made up of the following components:

- UI elements: components that render the data on the screen. You build these elements using [Jetpack Compose](#).
- State holders: components that hold the data, expose it to the UI, and handle the app logic. An example state holder is [ViewModel](#). The ViewModel component holds and exposes the state the UI consumes. The UI state is application data transformed by ViewModel. ViewModel lets your app follow the architecture principle of driving the UI from the model.

Unidirectional data flow pattern

A unidirectional data flow (UDF) is a design pattern in which state flows down and events flow up. By following unidirectional data flow, you can decouple composables that display state in the UI from the parts of your app that store and change state.

The UI update loop for an app using unidirectional data flow looks like the following:

- Event: Part of the UI generates an event and passes it upward—such as a button click passed to the ViewModel to handle—or an event that is passed from other layers of your app, such as an indication that the user session has expired.
- Update state: An event handler might change the state.
- Display state: The state holder passes down the state, and the UI displays it.
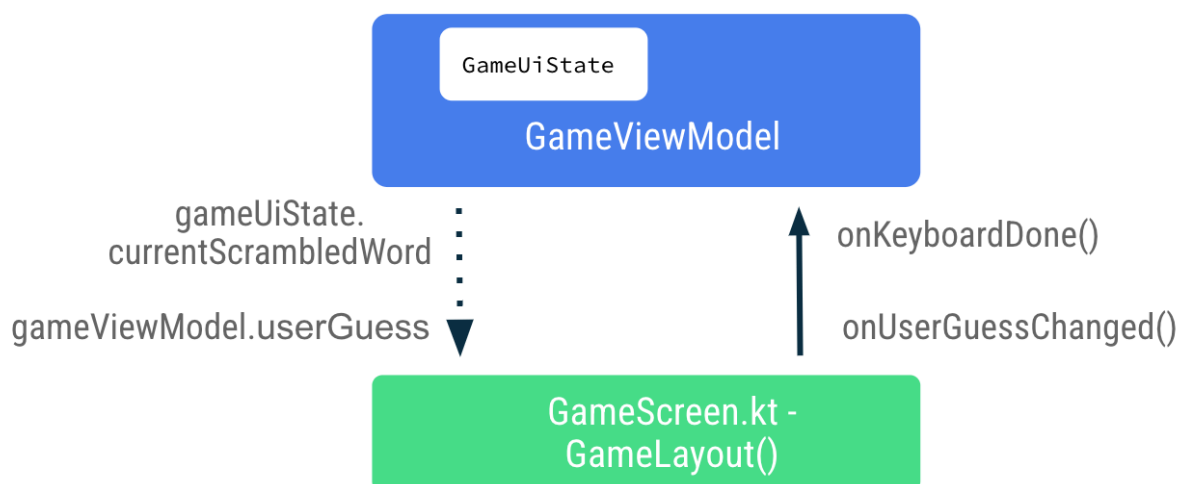
Homework 1: my Watch list

- **study this [example](#)** (it is a hands-on with code-along video support).
- create a simple application to manage your streaming "Watch list" (for movies and series).
- you should be able to add entries to the Watch List (stuff you want to watch) and, later, you should be able to mark entries as already watched (e.g.: tick a box). For convenience, start the app with a few entries pre-filled. Allow deletions too.
- you should use Composables, stateless UI, a list and the ViewModel pattern. Try to rotate the device while using the app (the list should not be lost with configuration changes…).
- for now, it is not needed to save the information in a persistent way (i.e., save to a database), but, if you want to, conside using [Preferences Datastore](#).
- update your personal git with the Homework 1 implementation.

Project Topics - Android module

Assessment criteria

Assessment criteria	Main Topics
A) Goals achievement: mobility use cases	Addresses use cases suitable for mobile/mobile first. Camera-intensive, geo (maps and/or location-intensive). Proactive notifications (local and/or Push notifications) Sensors (built-in or connected sensors) and adaptation to (sensed) user context UX coherent with [mobility use cases and language](#) (e.g.: content first,…) Others depending on the nature of the project
B) Complexity (of the implementation)	Adopts architeture guidelines/patterns and best practices Implements a backend strategy Data layer (strategy for); still useful if offline User-readiness (is UX ready for users?) UX continues across devices Others depending on the nature of the project
C) Effort	Fair effort for the team (taking into consideration the starting baseline for each student profile)

Seed projects (if needed): [proposal](#).