Ilídio C Oliveira (ico@ua.pt)

Last revision: 2020-10-21 *under construction*

Android labs – 2020/21

Learning resources

- Android documentation: tutorials, API documentation, tools, best practices,…
- Android training, including courses by Google
- Android Courses at Udacity (by Google staff members)
- R. Meirs' "Professional Android" Book
- CodePath Android Cliffnotes: very good collection of topics on Android development.

Lab #1- Introduction to the development workflow and tools

Readings & learning resources

- Required: concepts for Lesson 1 and Lesson 2 in Android Developer Fundamentals course
- Optional: slides for lesson 1.* and lesson 2.*
- Optional: Meiers' PA4: Chap. 2, 3 & 5.

Lab

Pre-requirements: install Android Studio.
In this lab, we will complete:

a) "Build your first app" – introductory tutorial, from Android documentation.
b) Code labs for "Lesson 2.x" (from Android Developer Fundamentals course materials)

HW/checkpoint assignment

CA1: build an app that acts as a dialer, with a “keypad” to enter the calling number. Start with the simplest approach possible.
When you press the dial button, a call should be started (just hand-over to the “real” built-in dialer).
You should add a set of 3 “speed dial” buttons (memories); when the users does a long press on one of these “speed dials”/memories, a secondary activity is offered to allow the user to update the speed dial details (define a label and associate a phone number).

Explore

- CodePath Android Cliffnotes: very good collection of topics on Android development.

#2- Flexible user interfaces and fragments

Readings & learning resources

- Concepts for Lesson 4 and Lesson 5 in ADF course
- Concepts for Lesson 1: Fragments in Android Developer Advanced
- Optional: slides for lesson 4.* and 5.* in ADF
- Optional: slides for Lesson 1: Fragments in ADAdv
- The Android Studio visual Layout Editor
- Optional: Meiers’ PA4: Chap. 2, 3 & 5.

Lab

Proposed lab activities: (mostly from the Android Developer Fundamentals course materials):

a) Code lab 4.1 (Clickable images), from ADF course
b) Code lab 4.5 (RecyclerView), from ADF course. Make sure the RecyclerView AndroidX library.
c) Code lab for Lesson 1: Fragments (1.1 + 1.2) from Android Developer Advanced
d) Code lab 4.4 (user navigation), from ADF course
e) Code lab 5.3 (adaptative layouts), from ADF course
f) Code lab 4.3 (menu and pickers), from ADF course

Note 1: some code labs use the old support library (e.g: android.**support**.v7.widget.RecyclerView). You should prefer, instead, the new packages under AndroidX (e.g.: **androidx.**recyclerview.widget.RecyclerView). If you need to use the old packages, when creating the project, select the option to use legacy libraries. More info on migrating to AndroidX.

Note 2: be sure to complete the code labs b) and c). Fragments and the ReciclerView will appear very often.

Explore

- Material design guidelines for User Experience (UX) and look-and-feel.
- Another guide to RecyclerView.
- Example of Master-Detail navigation, with Fragments.

#3- Background tasks & reacting to the user context

Readings & preparation

- Concepts for Lesson 7.3, 7.1 in the Background Tasks chapter of ADF.
- Concepts for Lesson 7.1 (Location) in the ADA.

Lab@Home

Be sure to complete the following labs:

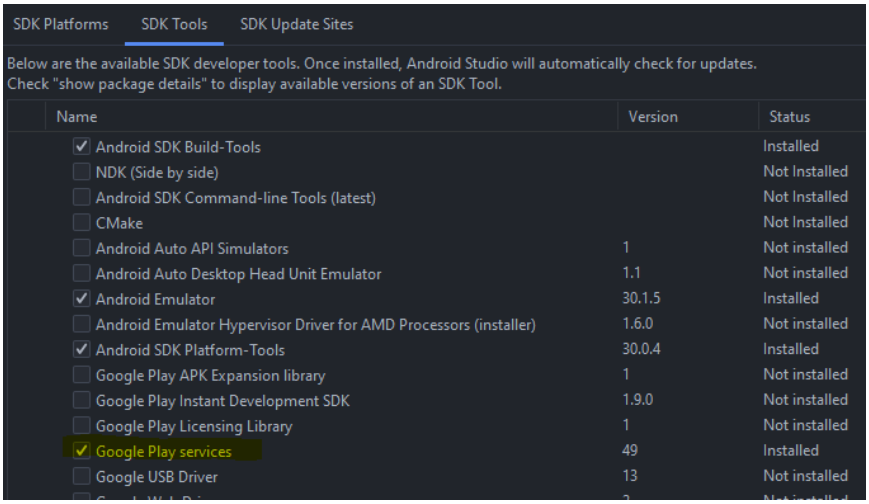
A) Code lab 7.1 (Fundamentals): AsyncTask
B) Code lab 7.1 (Advanced): Get device location and track location updates (this example shows several interesting points, besides the location. Should be completed carefully.)
C) Code lab 7.3 (Fundamentals): Broadcast receivers

Notes on the code labs:

A) In some cases, there is a starter project, or you may want to run the solution code. Note that the projects refer to somewhat old configurations. You may want to:
   - Update ***compileSdkVersion*** and ***targetSdkVersion*** in build.gradle to a recent version.
   - Use the new AndroidX instead og the old Support Libraries. There is a wizard for this refactoring: Menu Refactoring > Migrate to AndroidX

B) Android has two rules concerning the use of threads: do not block the UI thread (→any "heavier" work should be done in a separate thread); do UI work only on the UI thread (→access the UI only from the default thread, also called the UI thread). The **AsyncTask** is a helper class that combine, in the same object, methods that run in a separate thread (can run lengthier tasks), and methods that run in the main thread (can update the UI). Check the key points about AsyncTask usage.
C) To get the device **location** and track the location updates, Google offers an optimized API outside the basic Android SDK. The Fused Location API is included in the Google Play services and it offer a shared, energy-aware location access. Note that to use Play services you need both to add the dependency to you project and have the Google Play API in the device.

   In the SDK Tools, check the availability of "Google Play Services":



Confirm the dependency import, using a recent version.

```
dependencies {
    implementation 'com.google.android.gms:play-services-location:17.1.0'
```

In the Location exercise:

- You will have to handle permissions. The code lab explains how to do it, but there is also a "receipt" in the last section of this document.
- Be sure that, if using an emulator, it supports Google APIs or Google Play.
- Run Google Maps app (inside the emulator) to (force the) update the location cache (otherwise, location changes may not be assumed in the emulator)

D) **Broadcast Receivers** provide a highly decoupled messaging system in Android, in which one component can subscribe for updates in a topic, while other component would post information to data channel (Publisher/Subscriber pattern). This is useful, for example, to get notifications on system events (just boot, lost connection to WiFi,..). Check the key points about Broadcast Receivers usage..

Common programming cases & receipts

Permissions

- Dynamically verify and ask for App permissions (required from API 23+)