Google's Associate Android Developer Certification

An introduction to the Android paradigm

Through this article, Andrew Keymolen, Consultant at ADNEOM in Luxembourg tells you about the most notorious certification in Android development. Thinking about adding another string to your bow? Then this article is for you as it will tell you about an official Google certification, what it's about and how to get it.

What's the certification about and why should I pass it?

Developing Android apps isn't particularly hard. The hard part is developing *good* Android apps. To be able to do that, what better organization to learn from than those responsible for the development of Android themselves?
Being certified by Google will demonstrate your expertise in Android development to companies worldwide. It's valid for **36 months** during which you'll not only stand out in a sea of CVs, but you'll also be **listed in an index of certified developers** often consulted by potential employers.
The certification is relatively **easy** and **accessible**. First of all, it will only cost you **149$,** and Google being the experts they are at documenting their work, you will easily find study material all over the web and on Google's official site for Android app developers especially.
Starting from nothing *will* require some effort but the Android development community being what it is, they created a now officially recognized programming language made especially for Android development, Kotlin. That language is easy to pick up and will allow you to not have to learn the specificities of Java, the language in which Android apps used to be and still are developed in. Kotlin is worth diving into but it won't be necessary for the certification if you already have a background in Java development.
As a preparation for the exam, I can only recommend that you develop an app and actually publish it on the play store. It will only cost you a one-time payment of 25$ and will not only allow you to have a full first experience in mobile development but by combining the publication of your app with a Google Adwords promo, you will quickly get a thousand downloads and therefore have an app worth showcasing in your portfolio. Plus, if your app is successful, there wouldn't be any reason not to get a few bucks out of it!

What are the requirements?

As mentioned earlier, the exam being relatively easy, a **basic level of proficiency in Android development** is expected. Having developed one full app using Google's many guidelines on Android development would probably mean you're ready to take the exam.
The instructions and exit interview are in English meaning a **conversational level of English** is required. Also, Android development is easiest on Android Studio, a free and full IDE allowing

you to code in Java and in Kotlin, and it is expected that you will be using it to work on the certification so I can only recommend that you familiarize yourself with that indispensable tool. Finally, the fee of **149$** will get you **one attempt** at the certification. Beware that failing the first attempt will imply a 14 days waiting period before the second attempt, then a 2 months one, then a one year one.

How does the certification work?

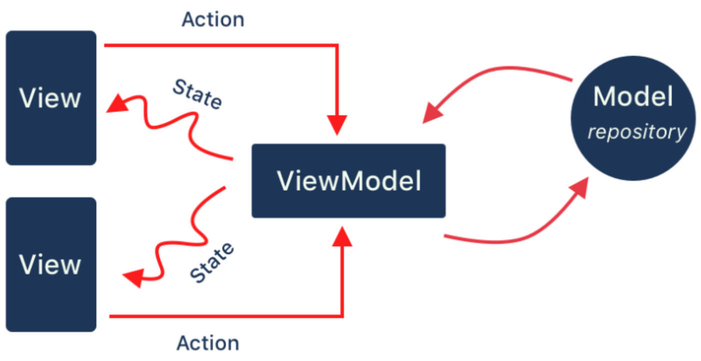
You'll be asked to download a package with a partly made app and finish it using Android studio before submitting it within **8 hours**. The exam will be about finishing the development of some features, debugging broken functionalities of the app, add features from scratch and test the app. All of which you'll have to do while following Google's clean code guidelines and design patterns, all while identifying the added and removed lines of code with simple comments. The app usually has most of the following functionalities:

- a database that you have to generate from Json data and that needs to be readable and writable;
- a RecyclerView component, which is the typical timeline-style list you have in most apps;
- notifications ;
- a settings/preferences screen ;
- some navigation features like menus, optimized for accessibility;
- custom Views - additional screens or dialogs you'll have to create from scratch;
- it's reactive thanks to the applied design pattern (MVVM).

Once the completed exam submitted, you'll be informed of your success shortly. It will then be time to schedule an exit interview. I can only speak for myself regarding that exit interview but it took the shape of a short call through an app with first some questions to identify me and then mostly technical questions. Those latter felt more like they were made to ensure I was the one having completed the exam than a mean to challenge my knowledge on Android development.

An introduction to the Android paradigm, part 1: An overview of the MVVM Design pattern

The MVVM is the design pattern officially supported by Google for Android development, making its implementation easier with the Android framework, and allowing for decoupled layers in reactive apps. Indeed, Google's supported MVVM is unidirectional, meaning that actions are dispatched by the ViewModel Layer. Thanks to that, the coupling between the View layer and the ViewModel layers is minimal, and the unit testing of the view is made easier, as is the development of additional features for the app. Such a design patterns allows for a reactive behavior as the View will automatically update when the data is updated in the Model layer thanks to exposed datastream, a component reflecting the **state** of the data to which any component can subscribe and therefore react when necessary.



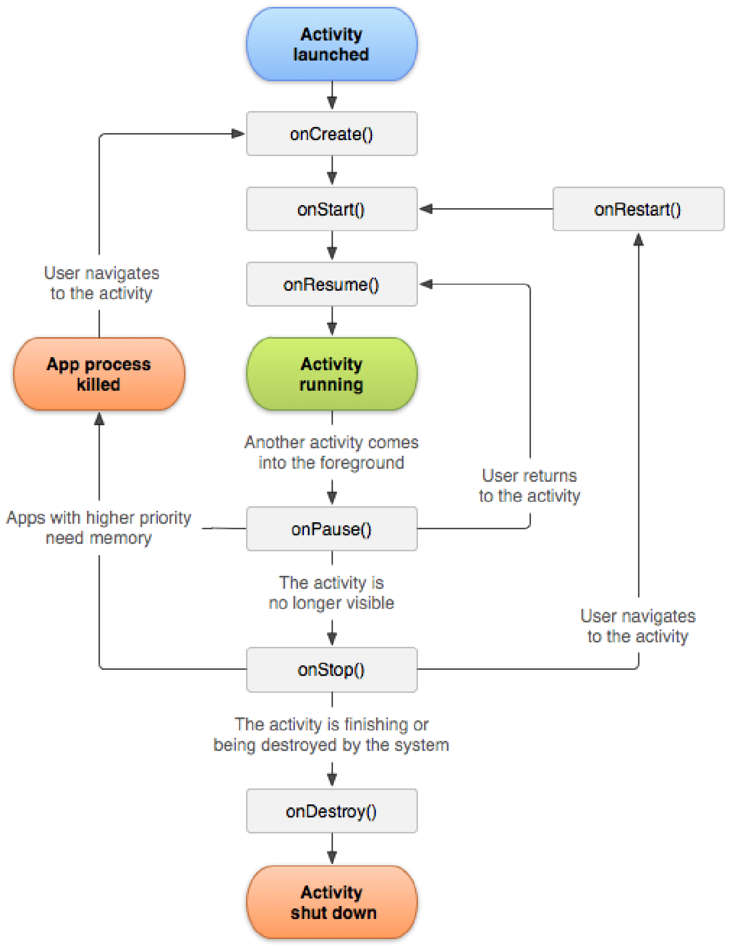
In short:

- The ViewModel interacts with the Model (API, database, etc) : fetch & update.
- The ViewModel exposes data streams of the **state**. On every update, the view component will render itself.
- When users take a certain action (eg. save a note), the view would ask the ViewModel to perform certain actions. The ViewModel would update the Model, create a new **state** and dispatch it to the data streams.
- View component would receive the update and render the updated **state**.

An introduction to the Android paradigm, part 2: An overview of the Android lifecycle

Among the most important aspects of Android development and also the most difficult to get used to is the Activity lifecycle.
An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI. Therefore an Activity is usually a screen dedicated solely to one of the main functionalities of your app and will be furnished with view components dedicated to more specific aspects of the functionality. For example, the main screen on your app will be an Activity and it will contain Fragments, which represent a behavior or a portion of user interface in an Activity. Those Fragments will finally contain the actual visual components such as a RecyclerView or even raw data.
Navigation and memory management being what it is on mobile, it wouldn't be smart to have that Activity loaded at all time, hence the Activity lifecycle. If the implementation of the MVVM design pattern is made properly and according to Google's guideline, this lifecycle should be seamless. However, it's still important to understand and keep that lifecycle in mind as it will probably be the cause of many of the losses of memory and NPE you'll encounter. Some steps

of the lifecycle are more significant than others and we will explore them here, but for a deeper look at this important aspect of Android development, I recommend that you visit Google's dedicated webpage on the topic. It's also worth noting that Fragments have their own lifecycle.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

- **OnCreate()**: On Activity creation, the Activity enters the Created state. In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity. For example, your implementation of onCreate() might bind data to lists, associate the activity with a ViewModel, and instantiate some class-scope variables. This method receives the parameter savedInstanceState, which is a Bundle object containing the Activity's previously saved state.

- **OnResume()**: When the activity enters the Resumed state, it comes to the foreground, and then the system invokes the onResume() callback. This is the state in which the app interacts with the user. The app stays in this state until something happens to take focus away from the app. Such an event might be, for instance, receiving a phone call, the user's navigating to another activity, or the device screen's turning off.
  When an interruptive event occurs, the activity enters the Paused state, and the system invokes the onPause() callback.
  If the activity returns to the Resumed state from the Paused state, the system once again calls onResume() method. For this reason, you should implement onResume() to initialize components that you release during onPause(), and perform any other initializations that must occur each time the activity enters the Resumed state.

- **OnSaveInstanceState()**: As your activity begins to stop, the system calls the onSaveInstanceState() method so your activity can save state information to an instance state bundle. The default implementation of this method saves transient information about the state of the activity's view hierarchy, such as the text in an EditText widget or

the scroll position of a ListView widget.
To save additional instance state information for your activity, you must override onSaveInstanceState() and add key-value pairs to the Bundle object that is saved in the event that your activity is destroyed unexpectedly.

The Android Associate Developer Certification study guide

A study guide can be found online in order to prepare for the certification, so here we'll be having a look at the most important mentioned competencies against which you'll be tested. I can only recommend that you go through the whole study guide as a preparation for the exam.

Study guide part 1: Android Core

- *Know how to build and run an Android app*
  It might seem obvious but as mentioned earlier, the exam will take place on Android Studio so being accustomed to how it works is necessary
- *Be able to display a message outside your app's UI using **Notifications***
  This one is almost certain to be asked at the exam

Study guide part 2: User Interface

- Understand the ***Android activity lifecycle***
- Be able to create an Activity that displays a Layout
- Understand how to create a ***custom View class*** and add it to a Layout
- Understand how to display items in a ***RecyclerView***
  There are RecyclerViews in most modern app even though they come in many shapes. The exam will most certainly ask that you implement one
- Be able to ***bind local data to a RecyclerView list using the Paging library***
- Know how to implement menu-based navigation
- Understand how to implement drawer navigation
  Be aware that accessibility features might be required regarding navigation in the app

Study guide part 3: Data Management

- Understand how data handling works in Android in regard to the MVVM
- Be able to read and parse raw resources or asset files
  For example, creating a database from JSON data
- Be able to create ***persistent Preference data*** from user input
- Understand how to change the behavior of the app based on user preferences
  With those 2 points comes the implementation of a preference screen, facilitated thanks to the Android Framework

Study guide part 4: Other

- Familiarize with Android testing and its dedicated frameworks as well as with debugging using Android Studio

What's next?

You have now a good understanding of the paradigm of Android development and you know what to expect of the exam. What come next are the specifics. And the best way to learn and master them is to start developing an app. Think of an idea, download Android Studio and create your first app. Once you're confident in your skills in Android development, you can access Google's exam and become an Associate Android Developer.

References

I'm available for any enquiries at the address akeymolen@adneom.lu but for study material, the best free resources out there are the study guide and the Android developers website that contain complete guides regarding any aspects of Android development.

Good luck!