**Possible Exam Questions**

In Section Overview: Droid Development

The answers to following questions can be found in the [Study Material](/omd/droid-devel/-/blob/master/Overview/study-material--overview.md).

Hint: Explain your answers to a fellow student!

**Apps and Architecture**

* How does the **architecture of Android apps** differ from typical iOS-Apps?
* What is the main entry point into an app. Explain what the *manifest* file is used for. Explain, what are *activities* and what are *intents*?
* Explain and give examples of permissions granted at install time. Compare to runtime permissions.
* Could you explain terms such as activities, fragments, and views.
* Which unique ids and which certificates are necessary for development, signing and distribution?

**Kotlin Programming**

* How and where is Kotlin code executed?
* How does garbage collection work?
* Explain some of the key language features of Kotlin. For example: **type inference**, strong typing.
* Discuss compile-time vs. run-time errors/exceptions.
* What ar nullable types.
* Kotlin idioms (*idiomatic programming*): compare two of selected features (such as data classes, smart casts, operator overloading, generics, coroutines, default values, lazy properties, singletons, not-null shorthands, if-expressions...) to other languages.
* Can you show the usage of object-oriented programming (including initialisers, inheritance, **properties**, **extensions**, …) in Kotlin?
* Are calsses or data classes **passed by value**? Explain the (dis-) advantages of **pass by reference**.
* What is special with **Nullable types** and with the **save call operator** ?.?
* What are **default arguments** for functions?
* Fuctional programming: explain the use of **high-order-functions** and **lambdas**.
* Can I explain and demo advanced features of the Kotlin programming language: generics, **operator-overloading**, …
* Could you explain the usage, the advantages and disadvantages of advanced Object Oriented Programming (OOP) with Interfaces and Extensions?
* Explain the differences between **primary** and **secondary constructors**.
* Delegates: what could be an usage of **observable** properties?
* Why would many consider val more secure than var?
* Argue for the usage of == over ===.
* Compilation (on demand): could you explain the work of the Android Runtime (**ART**), which combines the advantages of *just in time* compiling with *ahead of time* compiling.

**GUIs, Navigation, and Accessibility**

* Explain what is necessary to create interfaces for Android Apps.
* Explain the use of an activity and a fragment.
* How can developers assign code for life-cycle methods (view appears) and user interactions (swipe gestures, tap a button).
* Name ways for **navigation**, and explain how to **pass data** from one activity to another.
* Name ways for **navigation**, and explain how to **pass data** from one fragment to another.
* Code generation: explain how (two-way) **data binding** works.
* Could you explain the most important **life-cycle** events?

**Concurrency and Web Services**

* How to keep the UI responsive with background tasks for longer operations?
* How to retrieve data in the background from web services?
* Explain the use of **suspendable functions** with **coroutines** to prevent blocking UIs and to improve performance?
* How and why are ReSTful web services called **asynchronously** (in concurrent threads)?
* Explain ways of **JSON serialisation** and the elaborate on the problem of type checking?
* How and when would you use **threads** over **coroutines**?

**Saving State and Persistency**

* How do **shared preferences** work? Give an example.
* Name three different ways to persist data.
* How to setup the O/R-Mapping (ORM) and how to update a schema?
* For which kind of data is it necessary to use secure storage?
* Explain the disadvantage of keeping data in a keystore?
* What is the difference between eager and lazy loading!
* Explain the main features of **firebase** platform, expecially authentication and the cloud firestore.

**Location Based Services**

* How to minimize user input? How to suggesting location, movement and context?
* What is necessary to use maps and add custom landmarks and points-of-interest (POIs)?
* How to use existing web services to improve user-experience?
* Could you explain the usage, the advantages and disadvantages of Location Based Services concerning privacy?
* Maps: what differences are between *Overlays*, *Markers*, and *Tiles*?
* Which accuracy can we expect from different ways of determining the location of a user?
* Could you name the idea and application of **geofencing**?
* What are the use and (practical) limitations of the **(reverse) geocoding**?

**Sensors and Actuators**

* Explain the advantages and disadvantages (according to a context) of polling versus callbacks to retrieve sensor data.
* How and when to employ different sensors (and different frameworks)? (Proximity, Ambient Light, ...) to provide feedback to users?
* Could you explain the usage, the advantages and disadvantages as well as the proper technical terminology for at least two actuators and five sensors?
* What is special (concerning the performance, power, connection time, kind of connection, security, ...) about bluetooth low energy (**BTLE**)?

**Operating System Security and Insecurity, Forensics, App Analysis and Secure Coding.**

* Explain the **Android Virtual Machine** and how it is embedded into the **Android Stack**.
* Consequences of **sandboxing** an app. How are restrictions/permissions enforced.
* What does it mean to code in a secure way? Give 3 examples!
* How and where to save client data (user credentials) in a secure way?
* Why and how is code quality related to security?
* Explain approaches for testing.
* What means code integrity and how to enforce code integrity.
* How can usability build trust, and could you as developer care for user privacy?
* Which steps are necessary/possible for an \*.apk debugging session?
* (OWASP) **risks**: give examples for *insecure data storage*, *insecure communication*, *insufficient cryptography*, and *insecure authorisation*.
* Explain the deployment certificates needed to put an app to the App Store?
* Could you draw the architecture and flow during usage of **biometric sensors**?
* Which limitations and restrictions are given for **inter-app communication**?
* Name ways for inter-process communication (**IPC**) and known is uses?
* Differences between **certificate** and **public key** pinning.
* Explain the use of **URL schemes** in Android?
* Could you discuss the different approaches of **separating apps** on Android and on iOS including the consequences?
* Can you explain the terms and specific tasks of **Secure Elements**.
* Draft the steps to **decompile** .apk files.
* Name relevant aspects of the Android *permission model*. Detail on differences of **install-time** (normal/signature), **runtime** (prompting users), and **special** (platform/OEM) permissions.
* Limitation of **code obfuscation**. What does it mean that the R8 compiler shrinks, obfuscates and optimises an app.
* **Certificates**: Explain the terms (and differences): *client certificate*, *root certificate*, *certificate authority*, *self-signed certificate*, and *certificate chain*.
* Name at least three ways to harden your app.
* Compare **static** to **dynamic** (malware) **analysis**.

**System Services and Social Frameworks**

* Name and explain kinds of data exchange (not) possible between apps and the system, such as clipboard, file system, inter-process communication (IPC). Explain consequences to security and privacy. How are restrictions enforced?
* How to integrate App logic with system services such as AddressBook, PhotoAlbum or Calendar? Could you sketch the steps for accessing contacts?
* In which ways it is possible to integrate social media services for sharing or login?
* Which way of communication from Android to Android watches can be implemented?
* Explain the basic concepts of selected IPC methods, such as **binder, signals, sockets/ports, streams, pipes, or shared memory**.

**Cloud Services, Monitoring, and Testing**

* Which Google cloud services are available to developers and how to integrate them into own apps? Name limits and consequences, state ways to store documents transparently in the cloud.
* Can you explain different ways of synching to cloud storage?
* Which privacy and security issues have to be considered?
* How to **test asynchronous code**?
* What are the requirement for stable **UI-Tests**? What are the limits for UI-Testing?
* Which drawbacks concerning performance, security, footprint, cross platform usage are to expect with different cloud service providers?
* How could a cloud based password store be safe?

**Local and Remote (Push) Notifications**

* What are the important elements of the Google **push notification architecture**?
* Could you name the main differences between the Apple, Google and Microsoft Push Services?
* State ways to consume the notification inside the client app?
* Can local and/or remote push notifications mitigate the problem of battery draining background processes?
* Could you draft first the **registration process**, the creation and use of credentials/logins/tokens, and then the flow of notifications. Explain the flow (step-by-step) and show how to setup push notification on the server?
* How and where can push notification services be attacked? Consider the security implications for developers, especially the app server with custom setup and custom logic.
* How **background services** work with Android compared to push notifications.