Interview Questions for Senior Android Developer

Medium and Advanced Topics of the Kotlin Programming Language (20 Questions)

Medium Complexity (10 Questions)

1. What are Kotlin's collection types, and how do they differ from Java's collection types?
   - Follow-up: Can you explain the difference between `List`, `Set`, and `Map` in Kotlin?
   - Follow-up: How would you convert a Java collection to a Kotlin collection?
   - Follow-up: What is the purpose of the `mutable` and `immutable` collections in Kotlin?
   - Follow-up: Can you give an example of using the `filter` function on a list?
   - Follow-up: How do you handle nullability in Kotlin collections?

2. How do you use extension functions in Kotlin? Can you provide an example?
   - Follow-up: What are the benefits of using extension functions?
   - Follow-up: Can you override an extension function?
   - Follow-up: How do extension functions affect performance?
   - Follow-up: Can you create an extension function for a Java class?
   - Follow-up: How do you handle extension functions in a multi-module project?

3. Explain the concept of higher-order functions in Kotlin.
   - Follow-up: Can you provide an example of a higher-order function?
   - Follow-up: How do you use lambda expressions with higher-order functions?
   - Follow-up: What are inline functions, and how do they relate to higher-order functions?
   - Follow-up: Can you explain the difference between `invoke` and calling a function directly?
   - Follow-up: How do higher-order functions improve code readability?

4. What is the purpose of the `data class` in Kotlin?
   - Follow-up: How do data classes differ from regular classes?
   - Follow-up: Can you explain the auto-generated functions in a data class?
   - Follow-up: How do you handle inheritance with data classes?
   - Follow-up: Can you use data classes with Java libraries?
   - Follow-up: What are some best practices when using data classes?

5. How do you handle null safety in Kotlin?
   - Follow-up: What are the different types of nullability in Kotlin?
   - Follow-up: Can you explain the use of the `!!` operator?
   - Follow-up: How do you use the safe call operator `?.`?
   - Follow-up: What is the Elvis operator `?:` and how is it used?
   - Follow-up: How do you handle nullability when integrating with Java code?

Hard Complexity (10 Questions)

6. Explain the concept of coroutines in Kotlin and how they differ from threads.

  - Follow-up: What are the advantages of using coroutines over traditional threading?
  - Follow-up: Can you explain the role of `CoroutineScope`?
  - Follow-up: How do you handle cancellation in coroutines?
  - Follow-up: What is the difference between `launch` and `async`?
  - Follow-up: How do you handle exceptions in coroutines?

7. How do you integrate Kotlin with existing Java libraries?
  - Follow-up: What are some common interoperability issues you might encounter?
  - Follow-up: Can you explain how to call a Java method from Kotlin?
  - Follow-up: How do you handle Java's checked exceptions in Kotlin?
  - Follow-up: What are some best practices for using Java libraries in Kotlin?
  - Follow-up: Can you provide an example of a Kotlin extension function for a Java class?

8. Describe the use of sealed classes in Kotlin and their advantages.
  - Follow-up: How do sealed classes differ from enums?
  - Follow-up: Can you provide an example of using sealed classes for state management?
  - Follow-up: How do you handle when expressions with sealed classes?
  - Follow-up: What are the limitations of sealed classes?
  - Follow-up: How do you use sealed classes in conjunction with coroutines?

9. What is the purpose of the `companion object` in Kotlin?
  - Follow-up: How does it differ from static members in Java?
  - Follow-up: Can you have multiple companion objects in a class?
  - Follow-up: How do you use companion objects for factory methods?
  - Follow-up: Can you access a companion object from a Java class?
  - Follow-up: What are some use cases for companion objects?

10. Explain the concept of delegation in Kotlin and how it works.
  - Follow-up: What are the different types of delegation in Kotlin?
  - Follow-up: Can you provide an example of using the `by` keyword?
  - Follow-up: How does delegation improve code reusability?
  - Follow-up: Can you use delegation with interfaces?
  - Follow-up: What are some common pitfalls when using delegation?

Medium and Advanced Topics of Android and Jetpack (20 Questions)

Medium Complexity (10 Questions)

1. What is the MVVM architecture pattern, and how does it differ from MVC?
  - Follow-up: Can you explain the role of ViewModel in MVVM?
  - Follow-up: How do you handle data binding in MVVM?
  - Follow-up: What are the advantages of using LiveData in MVVM?

  - Follow-up: How do you test ViewModels in MVVM?
  - Follow-up: Can you provide an example of a simple MVVM implementation?

2. Describe the Room persistence library and its benefits.
  - Follow-up: How do you define an entity in Room?
  - Follow-up: What is the purpose of DAO in Room?
  - Follow-up: How do you handle migrations in Room?
  - Follow-up: Can you explain the difference between `@Insert`, `@Update`, and `@Delete`?
  - Follow-up: How do you perform queries using Room?

3. Explain the Navigation component in Jetpack and its advantages.
  - Follow-up: How do you set up a navigation graph?
  - Follow-up: Can you explain the difference between `NavHostFragment` and `NavController`?
  - Follow-up: How do you pass data between destinations in the Navigation component?
  - Follow-up: What are deep links, and how do you implement them?
  - Follow-up: How do you handle back navigation with the Navigation component?

4. What is Dagger, and how does it facilitate dependency injection in Android?
  - Follow-up: Can you explain the difference between Dagger 2 and Hilt?
  - Follow-up: How do you define a module in Dagger?
  - Follow-up: What are the scopes in Dagger, and why are they important?
  - Follow-up: How do you inject dependencies into a ViewModel using Dagger?
  - Follow-up: Can you provide an example of a simple Dagger setup?

5. Describe the purpose of the Android Lifecycle library.
  - Follow-up: How do you use LifecycleObserver in your application?
  - Follow-up: What are the different lifecycle states of an Android component?
  - Follow-up: How do you handle configuration changes using the Lifecycle library?
  - Follow-up: Can you explain the relationship between LiveData and Lifecycle?
  - Follow-up: How do you test components that use the Lifecycle library?

Hard Complexity (10 Questions)

6. Explain the concept of WorkManager and its use cases.
  - Follow-up: How do you schedule a one-time work request?
  - Follow-up: What are the differences between WorkManager and JobScheduler?
  - Follow-up: How do you handle constraints in WorkManager?
  - Follow-up: Can you explain the retry mechanism in WorkManager?
  - Follow-up: How do you observe the status of a work request?

7. What are the best practices for consuming APIs in Android applications?
  - Follow-up: How do you handle network errors gracefully?
  - Follow-up: Can you explain the use of Retrofit for API calls?
  - Follow-up: How do you implement caching for API responses?

   - Follow-up: What are the advantages of using Kotlin Coroutines with Retrofit?
   - Follow-up: How do you handle pagination in API responses?

8. Describe the concept of LiveData and its advantages in Android development.
   - Follow-up: How do you observe LiveData from a ViewModel?
   - Follow-up: Can you explain the difference between MutableLiveData and LiveData?
   - Follow-up: How do you handle configuration changes with LiveData?
   - Follow-up: What are some common use cases for LiveData?
   - Follow-up: How do you test LiveData in your application?

9. Explain the role of the Repository pattern in Android architecture.
   - Follow-up: How do you implement a Repository in your application?
   - Follow-up: What are the benefits of using a Repository pattern?
   - Follow-up: How do you handle data from multiple sources in a Repository?
   - Follow-up: Can you provide an example of a Repository that uses Room and Retrofit?
   - Follow-up: How do you test a Repository?

10. What is the purpose of the Android Jetpack libraries, and how do they improve app development?
    - Follow-up: Can you name some key Jetpack libraries and their functionalities?
    - Follow-up: How do Jetpack libraries help with backward compatibility?
    - Follow-up: What are the advantages of using Jetpack Compose?
    - Follow-up: How do you integrate Jetpack libraries into an existing project?
    - Follow-up: Can you explain the role of the Paging library in Jetpack?

Software Architecture (10 Questions)

Medium Complexity (5 Questions)

1. What is microservices architecture, and what are its advantages?
   - Follow-up: How do you handle communication between microservices?
   - Follow-up: What are some challenges you might face with microservices?
   - Follow-up: Can you explain the role of API gateways in microservices?
   - Follow-up: How do you manage data consistency in a microservices architecture?
   - Follow-up: What are some best practices for deploying microservices?

2. Describe the RESTful API design principles.
   - Follow-up: What are the key HTTP methods used in REST?
   - Follow-up: How do you handle versioning in REST APIs?
   - Follow-up: Can you explain the concept of statelessness in REST?
   - Follow-up: What are some common status codes used in REST APIs?
   - Follow-up: How do you document a RESTful API?

Hard Complexity (5 Questions)

3. What are some best practices for consuming APIs in Android applications?
   - Follow-up: How do you handle authentication and authorization in API calls?
   - Follow-up: Can you explain the use of Retrofit for API calls?
   - Follow-up: How do you implement error handling for API responses?
   - Follow-up: What are the advantages of using Kotlin Coroutines with Retrofit?
   - Follow-up: How do you handle rate limiting when consuming APIs?

4. Explain the concept of API versioning and its importance.
   - Follow-up: What are the different strategies for API versioning?
   - Follow-up: How do you handle breaking changes in an API?
   - Follow-up: Can you provide an example of a versioned API endpoint?
   - Follow-up: How do you communicate version changes to API consumers?
   - Follow-up: What are some common pitfalls to avoid with API versioning?

5. Describe the role of API gateways in microservices architecture.
   - Follow-up: How do API gateways improve security in microservices?
   - Follow-up: Can you explain the difference between an API gateway and a load balancer?
   - Follow-up: How do you implement rate limiting with an API gateway?
   - Follow-up: What are some common features of API gateways?
   - Follow-up: How do you monitor API usage with an API gateway?

Design Patterns (10 Questions)

Medium Complexity (5 Questions)

1. What is the Builder pattern, and when would you use it?
   - Follow-up: Can you provide an example of the Builder pattern in Kotlin?
   - Follow-up: How does the Builder pattern improve code readability?
   - Follow-up: What are the advantages of using the Builder pattern over constructors?
   - Follow-up: Can you explain the difference between the Builder pattern and the Factory pattern?
   - Follow-up: How do you implement the Builder pattern for a complex object?

2. Explain the Model-View-ViewModel (MVVM) pattern.
   - Follow-up: How does MVVM differ from MVP?
   - Follow-up: What are the roles of Model, View, and ViewModel in MVVM?
   - Follow-up: How do you handle data binding in MVVM?
   - Follow-up: Can you provide an example of MVVM in an Android application?
   - Follow-up: How do you test ViewModels in MVVM?

Hard Complexity (5 Questions)

3. Describe the Model-View-Presenter (MVP) pattern and its advantages.
   - Follow-up: How does MVP differ from MVVM?
   - Follow-up: What are the roles of Model, View, and Presenter in MVP?
   - Follow-up: How do you handle user interactions in MVP?
   - Follow-up: Can you provide an example of MVP in an Android application?
   - Follow-up: How do you test Presenters in MVP?

4. Explain the Model-View-Intent (MVI) pattern.
   - Follow-up: How does MVI differ from MVVM and MVP?
   - Follow-up: What are the key components of MVI?
   - Follow-up: How do you handle state management in MVI?
   - Follow-up: Can you provide an example of MVI in an Android application?
   - Follow-up: What are the advantages of using MVI over other patterns?

5. What are some common design patterns used in Android development?
   - Follow-up: Can you explain the Singleton pattern and its use cases?
   - Follow-up: How do you implement the Observer pattern in Android?
   - Follow-up: What is the Factory pattern, and how is it used in Android?
   - Follow-up: Can you provide an example of the Strategy pattern in Android?
   - Follow-up: How do design patterns improve code maintainability?

Performance and Optimization (6 Questions)

1. What are some common performance issues in Android applications?
   - Follow-up: How do you identify memory leaks in an Android app?
   - Follow-up: What tools do you use for performance profiling in Android?
   - Follow-up: How do you optimize the rendering performance of a RecyclerView?
   - Follow-up: Can you explain the impact of background tasks on app performance?
   - Follow-up: How do you handle large bitmap images efficiently?

2. Describe the importance of efficient layout design in Android.
   - Follow-up: How do you use ConstraintLayout to improve layout performance?
   - Follow-up: What are some best practices for using nested layouts?
   - Follow-up: How do you measure layout performance in Android?
   - Follow-up: Can you explain the role of ViewStub in optimizing layouts?
   - Follow-up: How do you handle layout inflation efficiently?

3. How do you optimize network calls in Android applications?
   - Follow-up: What are some strategies for caching network responses?

   - Follow-up: How do you handle large data transfers efficiently?
   - Follow-up: Can you explain the use of Retrofit's OkHttp for network optimization?
   - Follow-up: How do you implement pagination for API responses?
   - Follow-up: What are some best practices for handling API rate limits?

4. Explain the concept of lazy loading and its benefits.
   - Follow-up: How do you implement lazy loading in a RecyclerView?
   - Follow-up: Can you provide an example of lazy loading images?
   - Follow-up: What are the advantages of lazy loading for performance?
   - Follow-up: How do you handle lazy loading with data from a database?
   - Follow-up: What are some common pitfalls to avoid with lazy loading?

5. What are some best practices for optimizing battery usage in Android applications?
   - Follow-up: How do you handle background services efficiently?
   - Follow-up: What are the implications of using location services on battery life?
   - Follow-up: How do you implement WorkManager for background tasks?
   - Follow-up: Can you explain the role of JobScheduler in battery optimization?
   - Follow-up: How do you monitor battery usage in your application?

6. How do you profile and analyze the performance of an Android application?
   - Follow-up: What tools do you use for performance analysis?
   - Follow-up: How do you identify and fix performance bottlenecks?
   - Follow-up: Can you explain the use of Android Profiler in Android Studio?
   - Follow-up: How do you measure the impact of code changes on performance?
   - Follow-up: What are some common performance metrics to monitor?

Testing (10 Questions)

Medium Complexity (5 Questions)

1. What is unit testing, and why is it important in Android development?
   - Follow-up: How do you write a simple unit test in Kotlin?
   - Follow-up: What are some common testing frameworks used in Android?
   - Follow-up: How do you mock dependencies in unit tests?
   - Follow-up: Can you explain the role of JUnit in unit testing?
   - Follow-up: How do you test ViewModels in an MVVM architecture?

2. Describe the use of Mockito for mocking in unit tests.
   - Follow-up: How do you create a mock object using Mockito?
   - Follow-up: What are some common annotations used in Mockito?
   - Follow-up: How do you verify interactions with mock objects?
   - Follow-up: Can you explain the difference between `when` and `doReturn` in Mockito?

   - Follow-up: How do you handle exceptions in mocked methods?

Hard Complexity (5 Questions)

3. Explain the concept of integration testing in Android.
   - Follow-up: How do you set up an integration test environment?
   - Follow-up: What are some common tools for integration testing in Android?
   - Follow-up: How do you test API calls in integration tests?
   - Follow-up: Can you provide an example of an integration test for a ViewModel?
   - Follow-up: How do you handle database interactions in integration tests?

4. What is Espresso, and how is it used for UI testing in Android?
   - Follow-up: How do you write a simple UI test using Espresso?
   - Follow-up: What are some common matchers and actions used in Espresso?
   - Follow-up: How do you handle asynchronous operations in Espresso tests?
   - Follow-up: Can you explain the role of Idling Resources in Espresso?
   - Follow-up: How do you run Espresso tests on different devices?

5. Describe the importance of test-driven development (TDD) in Android.
   - Follow-up: How do you implement TDD in your development process?
   - Follow-up: What are the benefits of TDD for code quality?
   - Follow-up: Can you provide an example of TDD in action?
   - Follow-up: How do you handle refactoring with TDD?
   - Follow-up: What are some common challenges faced with TDD?

Continuous Deployment / Continuous Integration (CD/CI) (5 Questions)

1. What is Continuous Integration (CI), and why is it important in software development?
   - Follow-up: How do you set up a CI pipeline for an Android project?
   - Follow-up: What tools do you use for CI in Android development?
   - Follow-up: How do you handle build failures in a CI pipeline?
   - Follow-up: Can you explain the role of automated testing in CI?
- Follow-up: How do you integrate static code analysis into a CI pipeline?

2. **Describe your experience with Github Actions for Android CI/CD.**
   - Follow-up: How do you configure workflows for different build variants (debug/release)?
   - Follow-up: Can you cache dependencies to speed up builds in Github Actions?
   - Follow-up: How do you handle secrets (API keys, signing configs) securely?
   - Follow-up: What strategies do you use to parallelize test execution?
   - Follow-up: How do you trigger deployments to Firebase App Distribution?

3. **Explain the key differences between CircleCI and Github Actions.**

  - Follow-up: How do you configure a multi-module project in CircleCI?
  - Follow-up: What are some advantages of CircleCI's orbs system?
  - Follow-up: How do you handle build artifacts between jobs in CircleCI?
  - Follow-up: Can you describe a scenario where you'd choose one over the other?
  - Follow-up: How do you optimize build times in both systems?

4. **What is Continuous Deployment, and how does it differ from Continuous Delivery?**
  - Follow-up: How do you implement staged rollouts on Google Play?
  - Follow-up: What safeguards prevent broken builds from reaching production?
  - Follow-up: How do you monitor deployments for regressions?
  - Follow-up: Can you explain the role of feature flags in CD?
  - Follow-up: How do you handle rollback scenarios?

5. **Describe a robust CI/CD pipeline for an enterprise Android app.**
  - Follow-up: How do you integrate SonarQube for code quality checks?
  - Follow-up: What metrics do you track in build/release dashboards?
  - Follow-up: How do you handle environment-specific configurations?
  - Follow-up: Can you implement automated screenshot testing in the pipeline?
  - Follow-up: How do you coordinate CI/CD across multiple feature branches?

Coroutines (5 Questions)

1. **Explain the coroutine dispatchers and their appropriate use cases.**
   - Follow-up: When would you use `Dispatchers.IO` vs `Dispatchers.Default`?
   - Follow-up: How do you create a custom coroutine dispatcher?
   - Follow-up: Why shouldn't you use `Dispatchers.Main` for network calls?
   - Follow-up: How does `Dispatchers.Unconfined` behave differently?
   - Follow-up: How do you test code that uses specific dispatchers?

2. **Describe structured concurrency in Kotlin coroutines.**
   - Follow-up: What happens when a coroutine scope is cancelled?
   - Follow-up: How do `coroutineScope` and `supervisorScope` differ?
   - Follow-up: Why is `GlobalScope` generally discouraged?
   - Follow-up: How do you handle lifecycle-aware coroutines in ViewModels?
   - Follow-up: Can you implement a timeout pattern using structured concurrency?

3. **How do you handle exceptions in coroutines?**
   - Follow-up: What's the difference between `CoroutineExceptionHandler` and try/catch?
   - Follow-up: How do supervisor jobs alter exception propagation?
   - Follow-up: Can you recover from exceptions in a `launch` block?
   - Follow-up: How do you test exception handling in coroutines?
   - Follow-up: What happens when multiple children coroutines fail?

4. **Explain the difference between `launch` and `async` in coroutines.**
   - Follow-up: When would you use `async` without awaiting the result?
   - Follow-up: How do you handle cancellation of parallel `async` operations?
   - Follow-up: Can you convert a callback-based API to use `suspendCancellableCoroutine`?
   - Follow-up: How do you merge multiple `Deferred` results?
   - Follow-up: What are the performance implications of excessive `async` calls?

5. **Optimize this coroutine-heavy screen loading multiple data sources:**
   - Follow-up: How would you implement parallel data fetching?
   - Follow-up: What strategies prevent UI freezing during loading?
   - Follow-up: How do you handle partial loading failures?
   - Follow-up: Can you implement retries with exponential backoff?
   - Follow-up: How would you add a loading timeout?

Publishing (5 Questions)

1. **Describe your process for preparing an app for Google Play release.**
   - Follow-up: How do you configure different build flavors for staging/production?
   - Follow-up: What optimizations do you make to the release APK/AAB?
   - Follow-up: How do you implement app signing with Play App Signing?
   - Follow-up: What metadata is essential for store listings?
   - Follow-up: How do you handle multi-APK splits for different ABIs?

2. **Explain the different release tracks on Google Play.**
   - Follow-up: How do staged rollouts help mitigate risk?
   - Follow-up: When would you use internal testing vs closed/open tracks?
   - Follow-up: How do you promote builds between tracks?
   - Follow-up: What metrics do you monitor before full rollout?
   - Follow-up: How do you handle emergency rollbacks?

3. **What are Android App Bundles, and why are they preferred over APKs?**
   - Follow-up: How do dynamic feature modules work with bundles?
   - Follow-up: What are the size savings mechanisms in bundles?
   - Follow-up: How do you test app bundles locally before upload?
   - Follow-up: Can you explain Play Feature Delivery's install-time vs on-demand?
   - Follow-up: How do bundles affect instant app functionality?

4. **How do you monitor app stability after release?**
   - Follow-up: What Firebase Crashlytics features help diagnose issues?
   - Follow-up: How do you prioritize crash fixes based on impact?
   - Follow-up: Can you implement custom logging for production debugging?

   - Follow-up: How do ANR rates factor into your stability assessment?
   - Follow-up: What's your process for hotfixing critical crashes?

5. **Describe your approach to app update adoption strategies.**
   - Follow-up: How do in-app update APIs (flexible/immediate) work?
   - Follow-up: What analytics help measure update rollout success?
   - Follow-up: How do you communicate changes to users?
   - Follow-up: What's your strategy for deprecating old app versions?
   - Follow-up: How do you handle forced updates for breaking changes?
