1. **Question: What is the difference between Serializable and Parcelable interfaces in Android?**

**Answer**: Both Serializable and Parcelable interfaces are used to transfer data between components in Android. However, Parcelable is more efficient than Serializable when it comes to performance. Serializable uses reflection, which can slow down the serialization and deserialization process. On the other hand, Parcelable requires explicit implementation, but it performs better by using direct memory access.

2. **Question: Explain the concept of Content Providers in Android.**

**Answer**: Content Providers facilitate the sharing of data between applications. They offer a standardized interface to access and manipulate data stored in a central repository. Content Providers allow applications to query, insert, update, and delete data, making it accessible to other applications with appropriate permissions. They play a crucial role in enabling inter-app communication and data sharing.

**3. Question: What are the differences between a Fragment and an Activity in Android?**

**Answer**: An Activity represents a single screen in an Android application. It manages the user interface and responds to user interactions. On the other hand, a Fragment represents a portion of an Activity. Fragments allow you to build more modular and reusable components within an application. They have their lifecycle and can be added or removed dynamically within an Activity.

**4. Question: How does Dependency Injection (DI) work in Android?**

**Answer**: Dependency Injection is a design pattern that promotes loose coupling between classes. In Android, DI frameworks like Dagger or Koin are commonly used to manage dependencies. DI involves creating interfaces or abstract classes for dependencies, which are then injected into the dependent classes during runtime. This approach improves code maintainability, testability, and allows for easy swapping of dependencies.

**5. Question: What is the purpose of ProGuard in Android development?**

**Answer**: ProGuard is a tool used for code shrinking, optimization, and obfuscation in Android development. It analyzes the compiled code and removes unused classes, fields, and methods, reducing the application’s size. Additionally, ProGuard can obfuscate the code by renaming classes, fields, and methods, making it harder for reverse engineers to understand the code and modify it.

**6. Question: How can you handle orientation changes in an Android application?**

**Answer**: Orientation changes, such as rotating the device, can cause an Activity to restart. To handle orientation changes properly, you can override the onSaveInstanceState() method to save important data. The saved data can be retrieved in the onCreate() or onRestoreInstanceState() method to restore the previous state of the Activity. Additionally, using Fragments and ViewModel can provide a more robust solution for managing orientation changes.

**7. Question: What is the purpose of the Android Manifest file?**

**Answer**: The Android Manifest file (AndroidManifest.xml) is an essential file in every Android application. It provides essential information about the application to the Android operating system, such as the application package name, permissions required, activities, services, broadcast receivers, and content providers. It acts as a blueprint for the application, defining its components and characteristics.

**8. Question: What is the difference between a Service and an IntentService in Android?**

**Answer**: Both Service and IntentService are used to perform background operations in Android. The main difference lies in how they handle requests. A Service runs on the*main thread* and requires manual handling of background tasks and thread management. On the other hand, IntentService automatically creates a *worker thread* for each start request and handles requests sequentially, making it more suitable for simple, independent background tasks.

**9. Question: Explain the concept of Broadcast Receivers in Android.**

**Answer**: Broadcast Receivers are components in Android that enable communication between the system and the application or between different applications. They allow an application to receive and respond to system-wide events or custom-defined events. For example, a BroadcastReceiver can listen for events such as device boot, network connectivity changes, incoming SMS, or custom broadcast messages. It helps in implementing event-driven behavior in Android applications.

1. **Question: What is the difference between Serializable and Parcelable interfaces in Android?**

**Answer**: Both Serializable and Parcelable interfaces are used to transfer data between components in Android. However, Parcelable is more efficient than Serializable when it comes to performance. Serializable uses reflection, which can slow down the serialization and deserialization process. On the other hand, Parcelable requires explicit implementation, but it performs better by using direct memory access.

2. **Question: Explain the concept of Content Providers in Android.**

**Answer**: Content Providers facilitate the sharing of data between applications. They offer a standardized interface to access and manipulate data stored in a central repository. Content Providers allow applications to query, insert, update, and delete data, making it accessible to other applications with appropriate permissions. They play a crucial role in enabling inter-app communication and data sharing.

**3. Question: What are the differences between a Fragment and an Activity in Android?**

**Answer**: An Activity represents a single screen in an Android application. It manages the user interface and responds to user interactions. On the other hand, a Fragment represents a portion of an Activity. Fragments allow you to build more modular and reusable components within an application. They have their lifecycle and can be added or removed dynamically within an Activity.

**4. Question: How does Dependency Injection (DI) work in Android?**

**Answer**: Dependency Injection is a design pattern that promotes loose coupling between classes. In Android, DI frameworks like Dagger or Koin are commonly used to manage dependencies. DI involves creating interfaces or abstract classes for dependencies, which are then injected into the dependent classes during runtime. This approach improves code maintainability, testability, and allows for easy swapping of dependencies.

**5. Question: What is the purpose of ProGuard in Android development?**

**Answer**: ProGuard is a tool used for code shrinking, optimization, and obfuscation in Android development. It analyzes the compiled code and removes unused classes, fields, and methods, reducing the application’s size. Additionally, ProGuard can obfuscate the code by renaming classes, fields, and methods, making it harder for reverse engineers to understand the code and modify it.

**6. Question: How can you handle orientation changes in an Android application?**

**Answer**: Orientation changes, such as rotating the device, can cause an Activity to restart. To handle orientation changes properly, you can override the onSaveInstanceState() method to save important data. The saved data can be retrieved in the onCreate() or onRestoreInstanceState() method to restore the previous state of the Activity. Additionally, using Fragments and ViewModel can provide a more robust solution for managing orientation changes.

**7. Question: What is the purpose of the Android Manifest file?**

**Answer**: The Android Manifest file (AndroidManifest.xml) is an essential file in every Android application. It provides essential information about the application to the Android operating system, such as the application package name, permissions required, activities, services, broadcast receivers, and content providers. It acts as a blueprint for the application, defining its components and characteristics.

**8. Question: What is the difference between a Service and an IntentService in Android?**

**Answer**: Both Service and IntentService are used to perform background operations in Android. The main difference lies in how they handle requests. A Service runs on the*main thread* and requires manual handling of background tasks and thread management. On the other hand, IntentService automatically creates a *worker thread* for each start request and handles requests sequentially, making it more suitable for simple, independent background tasks.

**9. Question: Explain the concept of Broadcast Receivers in Android.**

**Answer**: Broadcast Receivers are components in Android that enable communication between the system and the application or between different applications. They allow an application to receive and respond to system-wide events or custom-defined events. For example, a BroadcastReceiver can listen for events such as device boot, network connectivity changes, incoming SMS, or custom broadcast messages. It helps in implementing event-driven behavior in Android applications.

**LiveData**

LiveData is a class that holds a data object and allows observers to receive updates whenever the data changes.

**SingleLiveEvent**

SingleLiveEvent is a subclass of LiveData designed for handling events in Android applications. It addresses the problem of re-triggering events on configuration changes (e.g., screen rotations) when using LiveData.

**MediatorLiveData**

MediatorLiveData is another subclass of LiveData that allows you to combine multiple LiveData sources into a single source. It is useful when you need to observe more than one data source and react to updates from any of them.



This prevents activity to recreate when orientation changes.

android:configChanges="orientation|screenSize|keyboardHidden"

Android will just rotate the screen and invoke one of the lifecycle callback method which is onConfigurationChanged(Configuration).