Android Interview Questions

Activity lifecycle:

- check this link

Fragment

- Lifecycle: check this link
- When should I use it? To **reuse a fragment in multiple activities** and **build flexible UI for different screen sizes**

Activities vs Fragments => OS , app

Fragment called by Application system when it's time for the fragment to draw its layout

Activity Called by Operating System

I'm not sure from this answer

Lunch Mode (Link)

1. Standard (Video) : https://www.youtube.com/watch?v=p2VcRmZLbpA

2. Single top (Video)

   If an instance of the **activity already exists at the top** of the current task, the system routes the intent to that instance through a call to its onNewIntent() method, rather than creating a new instance of the activity

   For example, suppose a task's back stack consists of root activity A with activities B, C, and D on top (the stack is A-B-C-D; D is on top). An intent arrives for an activity of type D. If D has the default "standard" launch mode, a new instance of the class is launched and the stack becomes A-B-C-D-D. However, if D's launch mode is "singleTop", the existing instance of D receives the intent through **onNewIntent()**, because it's at the top of the stack—the stack remains A-B-C-D. However, if an intent arrives for an activity of type B, then a new instance of B is added to the stack, even if its launch mode is "singleTop".

3. Single Task (Video)

   The system creates a new task and instantiates the activity at the **root** of the new task. However, if an instance of the activity already exists in a separate task, the system routes the intent to the existing instance through a call to its onNewIntent() method, rather than creating a new instance. **Only one instance of the activity can exist at a time**.

For example, suppose a task's back stack consists of root activity A with activities B, C, and D on top (the stack is A-B-C-D; D is on top). An intent arrives for an activity of type B.if B's launch mode is "singleTask", the existing instance of B receives the intent through onNewIntent() the existing instance and the stack will be (A-B) also notice that C and D activities get destroyed

4. Single Instance (Video)

   Always will be at the root of new task, no other activity can be part of this task

   A - B* - C - D -> Task 1: A -C - D and in the foreground of the tasks stack , Task 2 : B in back

OnSaveInstanceState Vs ViewModel

- Check this link for comparison (Link)
- **OnSaveInstanceState** is called before placing the activity in such a background state, allowing you to save away any dynamic instance state in your activity into the given Bundle, to be later received in onCreate(Bundle)
- **ViewModel** Is designed to store and manage UI-related data in a lifecycle conscious way. The ViewModel class allows data to survive configuration changes such as screen rotations.

Service Vs Intent Service

Check this comparison link

Service & Service types

A Service is an application component that can perform long-running operations in the **background (it doesn't mean that it runs out of the main thread)** , and it doesn't provide a user interface.A service runs in the **main thread** of its hosting process; the **service does not create its own thread** and does not run in a separate process unless you specify otherwise. If your service is going to perform any CPU-intensive work or blocking operations, such as MP3 playback or networking, **you should create a new thread within the service** to complete that work. By using a separate thread, you can reduce the risk of Application Not Responding (ANR) errors, and the application's main thread can remain dedicated to user interaction with your activities

Foreground Service: A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services must display a Notification. Foreground services continue running even when the user isn't interacting with the app.

Background: A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.

Bound : A service is *bound* when an application component binds to it by calling `bindService()`. A bound service offers a client-server interface that **allows components to interact with the service, send requests, receive results**, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

Service Limitation

While an app is in the foreground, it can create and run both foreground and background services freely. When an app goes into the background, it has a window of several minutes in which it is still allowed to create and use the services. At the end of that window, the app is considered to be idle. At this time, the system stops the app's background services

if you attempt to call startService() when your application is not in the foreground then an IllegalStateException

Broadcast Receiver

allows you to **register for system or application events**. All registered receivers for an event are notified by the Android runtime once this event happens.

Content provider

Share data between applications. you can access an existing content provider in another application using **content resolver** , or you may want to create a new content provider in your application to share data with other applications

Transformation Functions : Map & Flat Map & Switch Map & MediatorLiveData

Map: Transform one stream to another by Apply **function on each element and return a single value for each element**

Flat Map: Transform one stream to another by Apply function on each element and return a stream of new values **,FlatMap is used to map over asynchronous operations**

Switch Map: it will unsubscribe from previous observable every time it gets a new item that needs to be mapped, completely switching from one item to another

- **To understand the difference between Flat and Switch : check this example**

MediatorLiveData lets you add one or multiple sources of data to a single LiveData observable.

Dialog vs AlertDialog

The Dialog class is the base class for dialogs, but you should avoid instantiating Dialog directly. Instead, use one of the following subclasses:

AlertDialog: A dialog that can show a title, up to three buttons, a list of selectable items, or a custom layout.

DatePickerDialog or TimePickerDialog: A dialog with a pre-defined UI that allows the user to select a date or time.

Using DialogFragment to manage the dialog ensures that it correctly handles lifecycle events such as when the user presses the Back button or rotates the screen.

Work manager

WorkManager is intended for tasks that are deferrable—that is, not required to run immediately—and required to run reliably even if the app exits or the device restarts. For example:

- Sending logs or analytics to backend services
- Periodically syncing application data with a server

MVP vs MVVM

MVP	MVVM
Presenter has a view reference	ViewModel has't a view reference
Relation one to one We can't use the same presenter with different views	Relation one to many We can use the same view model with different views

Functional Programming & Reactive programming video

Functional programming :

Enforce functional idea

1. Argument fully determined as a parameter
2. Output depend only on arguments
3. Output is the return value
4. Example filter , filter not , map , reduce

   Map take an array and transform that into an array of the same length but with each individual item transformed.

   Filter transforms an array into a smaller array.

   FilterNot does the same thing as filter but inverted.

   Reduce : it can be used to express any list transformation, if you can't find a prebuilt list transformation https://medium.com/mindorks/functional-programming-in-kotlin-part-1-higher-order-functions-c60e2e4634b3

https://medium.com/mindorks/functional-programming-in-kotlin-part-2-map-reduce-ebdb3ebaa1f6

Reactive Programming

Programming with asynchronous data stream (stream : set of events order in time)

Clean Architecture : Check this link & Link

Dependency Injection Link

SOLID: Check this link

Coroutines (Job - Dispatcher - Scope) : Check this link

A coroutine is a concurrency design pattern that you can use on Android to simplify code that executes asynchronously

coroutines help to solve two primary problems:

- Manage long-running tasks that might otherwise block the main thread and cause your app to freeze.
- Providing main-safety, or safely calling network or disk operations from the main thread.

Dispatchers Types

- **Dispatchers.Main** - Use this dispatcher to run a coroutine on the main Android thread. This should be used only for interacting with the UI and performing quick work. Examples include calling `suspend` functions, running Android UI framework operations, and updating `LiveData` objects.
- **Dispatchers.IO** - This dispatcher is optimized to **perform disk or network I/O** outside of the main thread. Examples include using the Room component, reading from or writing to files, and running any network operations.

- **Dispatchers.Default** - This dispatcher is optimized to perform **CPU-intensive work outside of the main thread**. Example use cases include sorting a list and parsing JSON.

Dagger

Garbage Collection

What cause memory leak exception

ProGard Usage (Security file by encryption , reduce APK size)

Java

Interface Vs Abstract class

Answer in this link

Finally Keyword after Try / Catch

The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not. link

Can we define Final abstract method?

No, Final method can't be overridden

Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.

What is the difference between String literal and String object?

`String strObject = new String("Java");` **And** `String strLiteral = "Java";`

Answer in this link

== only return true for same object, not the same content of different object,

OOP Concepts

Answer in this link

Access Modifier

Private , Public , Protected , Default

Answer in this link

Define the difference between objects and primitives?

Answer in this link

RxJava

Used for multithreading, managing background tasks and removing callback hells.

Types of Observables

- **Observable:** emit a stream elements (endlessly)
- **Flowable**: emit a stream of elements (endlessly, with backpressure to emitting huge numbers of values)
- **Single:** emits exactly one element
- **Maybe:** emits zero or one elements
- **Completable:** emits a "complete" event, without emitting any data type, just a success/failure

How to run function in background in java?

Kotlin

Type of constructor

Static in kotlin

kotlin vs Java (Link)

Coding

1. **Write code to check if Linked List Circular / Linear**

2. **Find repeated number or missing one**

```
 static int getMissingNo(int a[], int n)

    {

        int i, total;

        total = n*(n + 1)/ 2;

        for (i = 0; i < n; i++)

            total -= a[i];

        return total;

    }
```

repeated number : it will be like missing number we will get total summation of element and subtract it from the total

3. **Check If text is palindrome text ?**

```
public boolean isPalindrome(String text) {

   String inputText= text.replaceAll("\\s+", "").toLowerCase();

   int length = inputText.length();

   int forward = 0;

   int backward = length - 1;

   while (backward > forward) {
        char forwardChar = inputText.charAt(forward++);

        char backwardChar = inputText.charAt(backward--);

        if (forwardChar != backwardChar)

            return false;

    }

    return true;

}
```