https://github.com/MindorksOpenSource/android-interview-questions
https://blog.mindorks.com/kotlin-android-interview-questions
https://aalishan565.medium.com/

1. How does kotlin work on android?

The Kotlin code is also compiled into the java bytecode and is executed at runtime by the java virtual machine (JVM) when a koltin file named Main.kt is compiled then it will eventually turn into class and then the bytecode of the class will be generated. the name of the bytecode fill will be MainKt.class and this file will be executed by the JVM.

2. Why should we use kotlin?

1. **Kotlin is Concise - Concise means reduce the extra code**
2. **koltin is null -safe**

   String name? = null // compile successfully

3.kotlin is interoperable -kotlin runs on JVM so it is totally interoperable with java

4.functional and Object-Oriented Capabilities- kotlin has a rich set of many useful methods and higher-order functions, lambda expressions, operator overloading, lazy elevation, operator overloading.

5. Smart Cast- it explicitly typecasts the immutable values and inserts the value in its safe cast automatically.

```
  fun main(args:Array){
  var str: String? = "BYE"
  print(str.length)// error compile time
  if(str!=null){   // compile successfully
  print(str.length)
} }
```

3.what is the difference between the variable declaration with Var and Val?

https://blog.mindorks.com/what-is-the-difference-between-const-and-val

1.immutable using Val Keyword- Immutable is also called read-only variables.we can not change the variable values if declared variable using Val Keyword.

eg. Val myName = "Sainath"

myName = "Pankaj" // compile-time error it gives error kotlin Val cannot be assigned

2.mutable using var Keyword - In mutable variable, we can change the value of the variable

```
eg.  var myAge =55
     myAge = 99
  print("my new age is ${myAge}") // compiles successfully
```

4. What is the difference between the variable declaration with val and cost?

https://blog.mindorks.com/what-is-the-difference-between-const-and-val

We use the keyword const and Val to declare an immutable property.

Const:- const keyword is used to declare those properties which are immutable in nature. Those properties are read-only properties. Const variable is known as compile-time constant
1.must be at the top level or member of object or member of a companion object
2.must be initialized with a String type or primitive type.
3.no custom getter
Val:- the Val keyword is also used for read-only properties. but the main difference between the const and Val is that the Val properties can be initialised at the runtime and const at compile-time

```
const  val companyname= “examen”// will not work
val companyname =”examen” // this will work

const val companyname = getCompanyName() //will not work because const val
val  companyName = getCompanyName() // this will work
```

5.how to ensure null safety in kotlin?
https://blog.mindorks.com/safecalls-vs-nullchecks-in-kotlin
One of the major advantages of using kotlin in null safety. In java, if you access some null variable then you will get a NullPointerException example
Code in kotlin will produce a compile-time error

```
var name:String = “Sainath”
Name = null // error
```

So to assign null values to a variable you need to declare the name variable as a nullable string and then during the access of this variable you need to use a safe Call operator

```
var name: String? = “Sainath”
print(name?.length) // ok
name = null // ok
```

6.what is the difference between safe calls(?) and null check(!!)?
https://blog.mindorks.com/safecalls-vs-nullchecks-in-kotlin
Safe call operator? is used to checking if the value of the variable is null or not if it is null then Null Will be returned otherwise it will return the desired value.

```
var name: String? = “Sainath”
println(name?.length) // 7
name  = null
println(name?.length) // null
```

If you want to throw NullPointerException when the value of the variable is null, then you can use the null check or !! operator

var name: String? =”sainath”
println(name?.length)//8
name = null
println(name!!.length) //KotlinNullPointerExecption

7. Do we have a ternary operator in kotlin just like java?

No, we don’t have a ternary operator in kotlin but you can use the functionality of ternary by Operator by using if-else and Elvis Operator.

8. What is the Elvis operator in kotlin?

In Kotlin you can assign null values to a variable by using the null safety property. To check if a value is having null value then you can use if-else or can use the Elvis operator ?:
var name: String? = “Sainath”
val namelength = name?.length ?: -1
println(namelength)

The Elvis operator (?:) used above will return the length of name if the value is not null otherwise if the value is null then it will return -1

9.how to convert a kotlin source file to java file

1. Open your kotlin project in intellij IDEA/Android studio
2. Then navigate to Tools>Kotlin> show kotlin Bytecode.
3. Now click on the decompile button to get your java code from the bytecode

10.what is RxJava

RxJava is a Java VM implementation of Reactive Extensions.
What are Reactive Extensions
The official doc describes Reactive extensions(ReactiveX) as a library for composing asynchronous and event-based programs by using observable sequence.

11.what is observable and operator and Observer in Rxjava?

Observable - it does some work and emits some values.
Operator - it translates/modifies data from one form to another form.
Observer - get those values

12.what is Hilt and use benefits?

The hilt is a dependency injection library which is used injecting binding without worrying whole DI setup and writing.it is built on top of the dagger, Hilt benefits from it
Is compile-time correctness improved runtime performance and scalability?

13.What is the use of @JvmStatic, @JvmOverloads, and @JvmFiled in Kotlin?

@JvmStatic: This annotation is used to tell the compiler that the method is a static method and can be used in Java code.
@JvmOverLoads: to use the default values passed as an argument in kotlin

Code form the Java code, we need to use the @JvmOverloads annotation.
@JvmField: to access the field of a kotlin class from java code without using any getter and setter, we need to use the @JvmField in the kotlin code.

14. What is a data class in kotlin?

Data classes are those classes which are made just to store to some data, In Kotlin it marked as data, following eq

```
data class Developer(Val name: String, Val mobile: String)
```

When we mark a class as a data class you don't have to implement or create the following function as we do in java: hashCode(), Equal(), toString(), copy(). The compiler automatically creates those internally, so it also leads to clean code.
https://blog.mindorks.com/learn-kotlin-data-class

15.can we use primitive types such as int, float, in Kotlin?

In kotlin, we can't use primitive types directly. We can use classes like Int, Double, etc as an object wrapper for primitive, but the compiled bytecode has these primitive types.

16. What is String interpolation in kotlin?

If you want to use some variable or perform some operation inside a string then String Interpolation can be used. You can use the $ sign to use some variable in the string or can perform some operation in between {} sign.

```
var name =" Sainath"
println("Hello! I am learning from $name")
```

17 What do you mean by destructing in kotlin?

Destructing is a convenient way of extracting multiple values from data stored in a (possibly nested) object and arrays.it can be used in locations that receive data (such as the left-hand side of an assignment). Sometimes it is convenient to destructure an object into a number of variables for example.

```
val (name, age) = developer
```

Now we can use name and age independently like below.

```
println(name)
println(age)
```

https://blog.mindorks.com/learn-kotlin-destructuring-declarations

18.when to use the lateinit keyword in kotlin?

Lateinit is late initialization
Normally properties declared as having a non-null type must be initialized in the constructor however fairly often this is not convenient.
For example, properties can be initialized through dependency injection or in the setup method of the unit test. In this case, you cannot supply a non-null initializer in the constructor but you still want to avoid null check when

referencing the property inside the body of a class.to handle this case you can mark the property with the lateinit modifier.

https://blog.mindorks.com/learn-kotlin-lateinit-vs-lazy

19. How to check if a lateinit variable has been initialized or not?

You can check if the lateinit variable has been initialized or not using it with the help of isInitialized() method. This method will return true if the lateinit property has been initialized otherwise it will return false for example

```
class Person{
Lateinit var name: String
fun initilaizeName(){
println(this::name.isInitialized)// false
name = “sainath”
println(this::name.isInitialized)// true
}
}
fun main(args:Arrays<String>){
Person(). initilaizeName()
}
```

https://blog.mindorks.com/how-to-check-if-a-lateinit-variable-has-been-initialized

17. What is the difference between lateinit and lazy in kotlin?

Lazy can only be used for val properties whereas lateinit can only be applied to var because it can’t be compiled to a final field thus no immutability can be guaranteed.
If you want your property to be initialized from outside in a way probably unknown beforehand use lateinit.

https://blog.mindorks.com/learn-kotlin-lateinit-vs-lazy

18. Is there any difference between == operator and === operator?

Yes the == operator is used to compare the values stored in the variable and the === operator is used to check if the reference of the variable is equal or not. But in the case of primitive types, the === operator also checks for the value and not reference

Primitive example

```
val int1 =10
val int2 =10
println(int1==int2)// true
println(int1==int2)// true
```

Wrapper example

```
val num1 = Integer(10)
val num2 = Integer(10)
println(num1==num1) // true
println(num1===num2)// false
```

19. What is the forEach in kotlin?

In Kotlin, to use the functionality of a for-each loop just like java, we use a forEach function. Following is an example of the same

```
var listofItem = listOf("sai","om","ganesh","raj","bhagwan")
listofItem.foreach{
Log.d(Tag,it)
}
```

20.what are companion objects in kotlin?

In Kotlin, if you want to write a function or any member of the class that can be called without having the instance of the class then you can write the same as a member of a companion object inside the class.

To create a companion object, you need to add the companion keyword in front of the object declaration.

```
class ToBeCalled{
companion object Test{
fun callMe() = println("you are calling me")
}
}
fun main(args:Array<String>){
ToBeCalled.callMe()
}
```

https://blog.mindorks.com/companion-object-in-kotlin

21. What is the difference between FlatMap and Map in Kotlin?

FlatMap is used to combine all the items of lists into one list.

Map is used to transform a list based on certain conditions

https://blog.mindorks.com/flatmap-vs-map-in-kotlin

22.What is the difference between List and Array types in Kotlin?

If you have a list of data that is having a fixed size, then you can use an Array. But if the size of the list can vary then we have to use a mutable list.

https://blog.mindorks.com/difference-between-list-and-array-types-in-kotlin

24.Can we use the new keyword to instantiate a class object in kotlin?

No, In Kotlin we don't have to use the new keyword to instantiate a class object. To instantiate a class object, simply we use.

```
var varName = ClassName()
```

25.what are visibility modifiers in kotlin?

A visibility modifier or access specifier or access modifier is a concept that is used to define the scope of something in a programming language. In Kotlin, we have four visibility modifiers.
private: visible inside that particular class or file containing the declaration.
protected: visible inside that particular class or file
26.What are the types of constructors in Kotlin
Primary constructor: These constructors are defined in the class header and you can't perform some operation in it, unlike Java's constructor.
Secondary constructor: These constructors are declared inside the class body by using the constructor keyword. You must call the primary constructor from the secondary constructor explicitly. Also, the property of the class can't be declared inside the secondary constructor. There can be more than one secondary constructor in Kotlin.
https://blog.mindorks.com/primary-and-secondary-constructors-in-kotlin
27.What are Coroutines in Kotlin?
A framework to manage concurrency in a more performant and simple way with its lightweight thread which is written on top of the actual threading framework to get the most out of it by taking the advantage of cooperative nature of functions.
https://blog.mindorks.com/mastering-kotlin-coroutines-in-android-step-by-step-guide
28.What is the suspend function in Kotlin Coroutines?
Suspend function is the building block of the Coroutines in Kotlin. Suspend function is a function that could be started, paused, and resumed. To use a suspend function, we need to use the suspend keyword in our normal function definition.
29.What is the difference between Launch and Async in Kotlin Coroutines?
The difference is that the launch{} does not return anything and the async{} returns an instance of Deferred<T>, which has an await() function that returns the result of the coroutine like we have future in Java in which we do future.get() to the get the result.
launch: fire and forget
async: perform a task and return a result
https://www.youtube.com/watch?v=nC30UiDv8Xc
30.What are scopes in Kotlin Coroutines?
https://www.geeksforgeeks.org/scopes-in-kotlin-coroutines/
31.What is the open keyword in Kotlin used for?

By default, the classes and functions are final in Kotlin. So, you can't inherit the class or override the functions. To do so, you need to use the open keyword before the class and function. For example:
https://blog.mindorks.com/understanding-open-keyword-in-kotlin

32.What are lambdas expressions?

Lambdas expressions are anonymous functions that can be treated as values i.e. we can pass the lambdas expressions as arguments to a function, return them, or do any other thing we could do with a normal object. For example:

```
val add : (Int, Int) -> Int = { a, b -> a + b }
val result = add(9, 10)
```

https://blog.mindorks.com/understanding-higher-order-functions-and-lambdas-in-kotlin

33.What are Higher-Order functions in Kotlin?

A higher-order function is a function that takes functions as parameters or returns a function. For example, A function can take functions as parameters.

```
fun passMeFunction(abc: () -> Unit) {
 // I can take function
 // do something here
 // execute the function
 abc()
}
```

For example, A function can return another function.

```
fun add(a: Int, b: Int): Int {
 return a + b
}
```

And, we have a function returnMeAddFunction which takes zero parameters and returns a function of the type ((Int, Int) -> Int).

```
fun returnMeAddFunction(): ((Int, Int) -> Int) {
 // can do something and return function as well
 // returning function
 return ::add
}
```

And to call the above function, we can do:

```
val add = returnMeAddFunction()
val result = add(2, 2)
```

https://blog.mindorks.com/understanding-higher-order-functions-and-lambdas-in-kotlin

34.What are extension functions in Kotlin?

Extension functions are like extensive properties attached to any class in Kotlin. By using extension functions, you can add some methods or functionalities to an existing class even without inheriting the class. For example: Let's say, we have views where we need to play with the visibility of the views. So, we can create an extension function for views like,

```
fun View.show() {
 this.visibility = View.VISIBLE
}
fun View.hide() {
 this.visibility = View.GONE
}
```

and to use it we use, like,

```
toolbar.hide()
```

https://blog.mindorks.com/extension-functions-vs-static-utility-class

|| Android ||

1.What is Android

Android is an operating system based on the Linux kernel. Android is developed in the Android Open Source Project now maintained by Google. Android operating system divided into the four areas

1.Application: Contains the applications, like the Browser, Camera, Gallery, Music and Phone

2.Application framework:An API which allows high-level interactions with the Android system

3.Libraries and runtime - The libraries for many common framework functions, like graphic rendering, data storage, web browsing. Also contains the Android runtime, as well as the core Java libraries for running Android applications

4.Linux kernel - Communication layer for the underlying hardware.

2.What is AAPT?

AAPT2 (Android Asset Packaging Tool) is a build tool that Android Studio and Android Gradle Plugin use to compile and package your app's resources. AAPT2 parses, indexes, and compiles the resources into a binary format that is optimized for the Android platform.

3.What is DDMS- Dalvik Debug Monitor Server.

(Dalvik Debug Monitor Server) A debugging tool from the Android software development kit (SDK). Able to monitor operations in the emulator as well as real devices, DDMS reports the details of each processing thread and time spent whether in the app or Android OS.

4.How DVM works and what is that?

The Java Compiler(javac) converts the Java Source Code into Java Byte-Code(.class). Then DEX Compiler converts this (.class) file into Dalvik Byte Code i.e. “.dex” file. Dalvik Virtual Machine uses its own byte-code and runs “.dex”(Dalvik Executable File) file

Advantages-DVM supports the Android operating system only.
In DVM executable is APK.
Execution is faster.
From Android 2.2 SDK Dalvik has it’s own JIT (Just In Time) compiler.
DVM has been designed so that a device can run multiple instances of the Virtual Machine effectively.
Applications are given their own instances.
https://www.geeksforgeeks.org/what-is-dvmdalvik-virtual-machine/

5.What is ART?

ART or Android Runtime is an Android runtime that uses Ahead Of Time(AOT). By using AOT, what it does is it converts or compiles the whole High-level language code into Machine level code and at the time of installation of the app and not dynamically as the application runs(like in case of Dalvik). Compiling the whole code during installation results in no lag that we see when we run our app on our device. By doing so, the compilation becomes very fast.
https://blog.mindorks.com/what-are-the-differences-between-dalvik-and-art

6.Advantage of ART over DVM?

https://stackoverflow.com/questions/31957568/what-is-difference-between-dvm-and-art-why-dvm-has-been-officially-replaced-wi

7.What is ADB?

ADB, Android Debug Bridge, is a command-line utility included with Google’s Android SDK. ADB can control your device over USB from a computer, copy files back and forth, install and uninstall apps, run shell commands, and more.

8.What is ANR?

When the UI thread of an Android app is blocked for too long, an "Application Not Responding" (ANR) error is triggered. If the app is in the foreground, the system displays a dialog to the user,
https://developer.android.com/topic/performance/vitals/anr

9.What is the bundle and what is the use of it

https://www.geeksforgeeks.org/bundle-in-android-with-example/

Android Bundles are generally used for passing data from one activity to another. Basically here the concept of key-value pair is used where the data that one wants to pass is the value of the map, which can be later retrieved by using the key. Bundles are used with intent and values are sent and retrieved in the same fashion, as it is done in the case of Intent. It depends on the user what type of values the user wants to pass, but bundles can hold all types of values (int, String, boolean, char) and pass them to the new activity.

10.JVM, DVM and ART?

https://medium.com/programming-lite/android-core-jvm-dvm-art-jit-aot-855039a9a8fa

|| Intent ||

|| Base ||

51.tell all the android components?

App Components are the essential building block of an android app. There are four types of component

1.Activity 2.Broadcast receiver 3.Service 4.content providers

1.Activity: Activity is an android component that represents a single screen with having user interface UI eq. Email app contains an email list in a single screen.

2.Broadcast Receiver: A Broadcast Receiver is an Android component which allows you to register for system or application events. All registered receivers for an event are notified by the Android runtime once the event happens.

Applications can register for various system events like root complete or battery low, and the android system sends broadcasts when specific events occur. any application can also create its own custom broadcasts.

https://android.jlelse.eu/local-broadcast-less-overhead-and-secure-in-android-cfa343bb05be

3.Service: Services is an application component that can perform long-running operations in the background and it doesn't have a user interface.it can run in the background, even when the user is not interacting with your application.

4.Content providers: A Content provider is an android component.it manages a shared set of app data that you store in the file system in a sqlite database on the web or on any other persistent storage location that your app can access through the content providers.

52.what is the context and how to use it in android?

Context is the current state of the application,and it can be used to get information regarding the activity and application and it can be used to get access to resources, database and shared preference and both the activity and application classes extend the context class.

It is the context of the current state of application. It can be used to get information regarding the activity and application,getresource, databases, and shared preferences

53.what is AndroidMainfest.xml?

Android manifest.xml file contains information about your packages including Components of your application like services, activities, content providers, broadcast receivers, permission, and give a complete description of your app.

54.what is application class?

The application class in android is the base class within an android app that contains all other components such as activities and services. the application class or any subclass of the application class is instantiated before any other class when the process for your application/package is created.

|| Activity and fragment ||

1.What is Activity and its lifecycle?

Activity is an android component it is single screen with having UI that known Activity there are seven method onCreate(), onStart(), onResume(), onPause(), onStop(),
onDestory(), onRestart()
onCreate()-creates the activity and prepare to display
onStart() -when user can see the screen
onResume()-when user can interact the screen
onPause()-when part of app is visible but in background
onStop()-when app is not visible to user
onRestart()-called after your activity has been stopped,it being started again
onDestory()-when activity is destroyed
Test Case-1 when open the app - onCreate()-> onStart() ->onResume()
Test Case-2 when back button pressed and exit the app
onPause()-> onStop() -> onDestory()
Test case-3 when home button pressed or lock button
onPause(()-> onStop()

Test case-4 after pressed home button again open an app from the recent task list or clicking on the icon
onRestart()-> onStart()-> onResume()
Test case-5 when open app another app from notification bar or open settings
onPause()-> onStop()
Test case-6 back button pressed from another app or settings then the user can see our app
onRestart()-> onStart() ->onResume()
Test case-7 when any dialog open on the screen
onPause()
Test case-8 after dismissing the dialog or back button from the dialog
onResume()
2.when only onDestory() is called for an activity without onPause and onStop()?
When we call finish() method in onCreate() method system will call onDestory() method Directly.
3. Why do we need to call setContentView() in onCreate() activity class?
onCreate() method in activity lifecycle is called only once.in onCreate() method we want do most of initialization that why we set content by calling setContentView()
4.who is onSavedInstanceState() and onRestoreInstanceState() in activity?
1.onSavedInstanceState()-this method is used to store data before pausing the activity
2.onRestoreInstanceState()- this method is used to recover the saved state of an activity when the activity is recreated after destruction,so the onRestoreInstanceState() receives the bundle that contains the instance state information.
5.what is fragment and its lifecycle.
A fragment represents a reusable portion of your app's UI.A fragment defines and manages its own layout has its own lifecycle, and can handle its own input events.fragment cannot live on their own they must be hosted by activity or another fragment
Fragment provides two major thing
1.Modularity - fragment is better suited to define and manage the UI of a single screen and portion of screen eg.email application it will work on tablet and android phone
2.adaptablity - you can combine multiple fragments in a single activity to build a multi pane UI like what's up swiping tab chat status calls.

https://medium.com/@shubham_nikam/understand-the-activity-lifecycle-e4526ec1c436

6.why is it recommended to use only the default constructor to create a fragment?

Android framework decides to recreate our fragment. orientation changes it, recreates the fragment using the no-argument constructor and attaches a bundle to the fragment as it has stored the bundle earlier.

7.What is Launch mode?

Launch mode is an instruction for android os which specifies how the activity should be launched.it instruct how any new activity should be associated with the current task

Task- A task is a collection of activities that users interact with when performing a certain job. In general an application contains a number of activities.Normally when a user launches an application a new task will be created and the first activity instance is called the root of the task.

Back Stack- activities are arranged with the order in which each activity is opened

This maintained stack called back stack.when you start a new activity using startActivity(), it pushes a new activity into your task and put the previous activity in the back stack

There are four launch modes for activity

1.Standard-This is the default launch mode of an activity (If not specified). It creates a new instance of an activity in the task from which it was started. Multiple instances of the activity can be created and multiple instances can be added to the same or different tasks. In other words you can create the same activity multiple times in the same task as well as in different tasks.

2.SingleTop-In this launch mode if an instance of activity already exists at the top of the current task, a new instance will not be created and the Android system will route the intent information through onNewIntent(). If an instance is not present on top of task then new instance will be created.Using this launch mode you can create multiple instance of the same activity in the same task or in different tasks only if the same instance does not already exist at the top of stack

3.SingleTask-In this launch mode a new task will always be created and a new instance will be pushed to the task as the root one. If an instance of activity exists on a separate task, a new instance will not be created and the Android system routes the intent information through onNewIntent() method. At a time only one instance of activity will exist.

4.SingleInstance-This is a very special launch mode and only used in the applications that have only one activity. It is similar to singleTask except that no other activities will be created in the same task. Any other activity started from here will create a new task.

8.when should you use a fragment rather than an activity?

When you have some UI component to use across various activities.
When viewed, it can be displayed side by side just like a viewpager.

9.what is the difference between activity and fragment?

Activity is an application component that gives a user interface where the user can interact	The fragment is only part of an activity is basically contributes it ui
Activity is not dependent on fragment	Fragment is dependent on activity.it can't exist independently
We need to mention all activity it in the manifest.xml file	Fragment is not required to mention in the manifest file
We can't create multi-screen ui without using fragment in activity	Fragment cannot be used without an activity
Lifecycle methods are hosted by os.the activity has its own life cycle	Lifecycle methods in fragments are hosted by hosting the activity
Activity is not lite weight	The fragment is lite weight

10.what is a fragment lifecycle?

When fragment come up the screen
1.onAttach() - this method is called first to know that our fragment has been attached to an activity.we are passing the activity and will host our fragment.
2.onCreate()-this method called when a fragment instance initializes just after the onAttach where fragment attaches to the host activity.
3.onCreateView()-The method called when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View component from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.
4.onActivityCreated()-This method called when Activity completes its onCreate() method.
5.onStart()-This method called when a fragment is visible.
6.onResume()-This method called when a fragment is visible and allowing the user to interact with it. Fragment resumes only after activity resumes.

When fragment goes out off the screen:-
1.onPause()-This method called when a fragment is not allowing the user to interact; the fragment will get change with other fragment or it gets removed from activity or fragment activity called a pause
2.onStop()-This method called when the fragment is no longer visible; the fragment will get change with other fragment or it gets removed from activity or fragment activity called stop.
3.onDestroyView()-This method called when the view and related resources created in onCreateView() are removed from the activity's view hierarchy and destroyed
4.onDestroy()-This method called when the fragment does its final clean up.
11.What is the difference between FragmentPagerAdapter vs FragmentStatePagerAdapter?
1.FragmentPagerAdapter:each fragment visited by the user will be stored in the memory but the view will be destroyed.when the page is revisited then the view will be created not the instance of the fragment.
2.fragementStatePagerAdapter: here the fragment instance will be destroyed when it is not visible to the user except the saved fragment.
12.What is the difference between adding/replacing fragments in backstack?
replace removes the existing fragment and adds a new fragment.
add retains the existing fragments and adds a new fragment that means existing fragment will be active and they won't be in 'paused' state hence when a back button is pressed onCreateView() is not called for the existing fragment(the fragment which was there before new fragment was added).
13.what is Bundle and where to use and how to pass data from fragment to fragment?
The bundle is a container which carries data. We can use bundle in fragment to pass data one fragment to another fragment using Bundle object with key/value pair and set an object to fragment instance
Eg. HomeFragment homefragment = new HomeFragment();
homefragment.putString("NAME, "sai")//Key, value
homefragment.setArgument(bundle);
-Receiving data in SecondFragment using getArgument() method with KeyName
Eg: String name= getArgument("NAME") // KeyName
14.what is handler and how to use and implement
https://blog.mindorks.com/android-core-looper-handler-and-handlerthread-bd54d69fe91a

A handler allows communicating back with the UI thread from another background thread.This is useful in android as android doesn't allow other threads to communicate
Directly with UI thread.
Note: Android handles all the UI operations and input events from one single thread which is known as the Main or UI Thread.android collects all events in this thread in a queue and processes this queue with an instance of the looper class

15.what is Serialization and Deserialization ?

The process of serialization converting "Java Object to Bit and Byte"
The process of Deserialization converting "Bit and Byte to Java-Object"

16.serializable vs Parcelable

1.Serializable is a marker interface in JDK techniques which takes more memory and is slower.
2. Serializable is going to convert an object to byte stream so the user can pass the data between one activity to another activity.
3. Main advantage of serializable is that the creation and passing of data is very easy but it is a slow process as compared to parcelable.
1.Parcelable is an android predefined interface to pass an object faster and less memory wastage as compared to Serializable.
2.parcelable is going to convert objects to byte streams and pass the data between two activities.

17.what is difference between jdk and dvm

https://www.geeksforgeeks.org/difference-between-jvm-and-dvm/

|| Services ||

https://google-developer-training.github.io/android-developer-fundamentals-course-concepts/en/Unit%203/74_c_services.html

1.What is service in android?

Android service is a component that is used to perform operations on the background such as playing music, handling network transactions, interacting with content providers etc. It doesn't have any UI (user interface). The service runs in the background indefinitely even if the application is destroyed.

Types of services-

1.Started service or Background service(starts by calling startService())
2.Foreground service(starts by calling startForeground())
3.Bound service(starts by calling bindService())

2.What to use when?

1.Background service is used when your operation is not noticed by the user. For example if you are syncing your data with the server, a background service is recommended.
2.Foreground service is used when your operation should be noticed by the user. For example playing music or downloading a file.
3.Bound service is used when your operation is related to interprocess communication. A bound service allows components (such as activities) to bind to the service, send requests, receive responses, and even perform interprocess communication (IPC). In simple words, when you want to get results continuously from your service you should use a bound service.

3.What is the intent service?

The Service is the base class for the IntentService. Basically, it uses a “work queue process” pattern where the IntentService handles the on-demand requests (expressed as Intents) of clients. So, whenever a client sends a request then the Service will be started and after handling each and every Intent, the Service will be stopped. Clients can send the request to start a Service by using Context.startService(Intent). Here, a worker thread is created and all requests are handled using the worker thread but at a time, only one request will be processed.
To use IntentService you have to extend the IntentService and implement the onHandleIntent(android.content.Intent).

4.what is the difference between Service and IntentService?

1.usage: IntentService-if you want some background task to be performed for a very long period of time then you should use the IntentService. But at the same time you should take care that there is very less communication with the main thread.if the communication is required then you can use the main thread handler or broadcast receiver.
Service:You can use service for the tasks that don’t require any UI and also is it not a very long running task
2.How to start: IntentService-to start IntentService you have to use Intent and start IntentService by calling Context.startService(Intent).
Service-To start a service you need to call the onStartService() method.
3.Running Thread:IntentService- IntentService runs on a separate Worker-Thread that is triggered from the Main thread.
Service- service always runs on the main-thread
4.Triggering Thread: IntentService can be triggered only from the main thread.firstly the Intent is received on the Main-Thread and after that the worker thread will be executed.
Service can be triggered from any thread.

5.main thread blocking:IntentService there is no involvement of the main thread the tasks are performed in the from of queue first in first out
Service-if you are using service then there are chances that the main thread will be blocked because service runs on the main thread.
6.Stop service:Intentservice there is no need of stopping the service because the service will be automatically stopped once the work is done.
Service-if you are using service then you have to stop the service after using it otherwise the service will be there for an infinite period of time until your phone is in normal state. So to stop service you have to use StopService() and StopSelf()

5.What is the difference between started service and bound service Intent service?

Started Service	Bound Service	Intent Service
Started Service is used to perform long repetitive task	Bounded Service is used to perform background task in bound with another component	Intent Service is used to perform one time task i.e when the task completes the service destroys itself
Started Service gets started by calling startService().	bounded Service gets started by calling bindService().	Intent Service gets started by calling startService().
Started Service is stopped or destroyed explicitly by calling stopService(). Started Service is independent of the component in which it is started.	bounded Service is unbind or destroyed by calling unbindService(). bound Service dependents on the component in which it is started.	IntentService Implicitly calls stopself() to destroy Intent Service is independent of the component in which it is started.

6.What is Service class method and work?

method	Description
onCreate()	This method gets executed by calling startService() method from another component like activity . This method is executed only once during the service creation.
onStartCommand()	Th7is method gets executed by calling startService() method

	from another component like activity . This method is executed only once during the service creation.
onDestory()	If the service is in a running state a call to this method will stop the service from running.

7.what is Service behaviour

The onStartCommand method of service class returns an int type which defines the behaviour of service with the android system , that is how your service should behave to the android system when it gets killed , So Service class provides three static integer constants.

Return Constant	Description
Service.START_STICKY	If the service is got killed by the platform then restart the service without intent data
Service.START_NOT_STICKY	If the service is killed by the platform then don't restart, if you want to start then you must explicitly make a call to startCommand().
Service.*START_REDELIVER_INTENT*	If the service is killed by the platform then restart the service with intent data.

7.What is the difference between thread and services?

Service	Thread
Service is a component of android which performs long running operation in background, mostly without having UI	Thread is a O.S level feature that allows you to do some operation in the background.
If it is destroyed while performing its job, in the middle by Android due to low memory scenario. Then android will make sure that it will restart your service, if you have returned START_STICKY or START_REDELIVER_INTENT from onStartCommand().	Thread - if it is destroyed by android in the middle due to low memory, then android will not guarantee to restart it again. That means the user lost half of his work.

Service - is a component of android, so it has priority levels to be considered while destroying an application due to low memory.	Thread is not a component of android, so android will not take thread priority into consideration while killing an application due to low memory.

|| Broadcast Receiver ||

1.what is broadcast receiver in android?

https://android.jlelse.eu/local-broadcast-less-overhead-and-secure-in-android-cfa343bb05be

A Broadcast Receiver is an Android component which allows you to register for system or application events. All registered receivers for an event are notified by the android runtime once the event happens.

Applications can register for various system events like root complete or battery low, and the android system sends broadcasts when specific events occur.any application can also create its own custom broadcasts.

Receive Broadcast- to be able to receive a broadcast application have to extend the BroadcastReceiver abstract class and override its onReceive() method. If the event for which the broadcast receiver has registered happens the onReceive() method of the receiver is called by the android system.

Broadcast receiver only listens for those broadcast intent which are actually registered

2.How to register broadcast receiver

There are two way to register broadcast receiver

1.Manifest-declared (Statically):Registered application event into manifest.xml file android oreo background execution limitation applied.

2.Context-Registered (Dynamically): Registered application event in a java code. No limitation applied.

Register the receiver using registerReceiver() method in onResume() activity method.

UnRegister the receiver using unregisterReceiver() method in onPause() method.

Note: if you forgot to unregister the receiver then it will throw a leaked intent receiver error.

3. How to implement the broadcast receiver?

Create the class and extend that class with BroadcastReceiver.it will override the onReceive()method.Register the broadcast event using static in manifest xml file or dynamically using java code and set action using IntentFilter

-Never perform long running tasks inside the onReceive() method
-Receiver works in the main thread
-it will block the main ui thread
-system will generate ANR - create Service for long tasks.

4.what is the local broadcast receiver?

https://medium.com/@kreynaldi04/android-fundamentals-07-3-broadcast-receivers-3e9f2f9cfb40

It is use to facilitates the interaction between two application component within the same app.to send a local broadcast create broadcast intent and pass it to LocalBroadcastManager.sendBroadcast.it is introduced in android support library

5.why do we use a local broadcast receiver or LocalBroadcastManager?

-communication between two component
1.Between two activities. 2.Between activity and service.
3.between activity and BroadcastReceiver 4. Between service and BroadcastReceiver.
More Security
1.work locally not globally 2.broadcasted intent does not leave app 3.No data leak
4.no outside app can transmit broadcast to these receivers.

Normal Broadcast	Ordered broadcast	Sticky Broadcast	Local broadcast
sendBroadcast()	sendOrderedBroadcast()	Deprecated in Api 21	LocalBroadCastManager()
Implicit/Explicit	Implicit/Explicit		Implicit/Explicit
Dynamic/Static	Dynamic/Static		Dynamic

6.what is Normal Broadcast Receiver?

Normal broadcasts are asynchronous. Receivers of normal broadcasts run in an undefined order often at the same time to send a normal broadcast, create a broadcast Intent and pass it to sendbroadcast(Intent).

7.what is an Ordered Broadcast receiver?

It is used to send in an order Intent to broadcast receivers. it defines Priority of BroadcastReceiver using priority attribute in manifest file and uses abortBroadCast()to stop the further broadcast. To send an ordered broadcast, create a broadcast intent and Pass it to sendOrderedBroadcast(Intent,String)

|| View ||

1.difference between RelativeLayout and LinearLayout?

1.LinearLayout: arranges elements either Vertically and Horizontally in a row-column.

2.RelativeLayout: arranges elements relative to the parent or other element.

2.what is an adapter?

An Adapter is responsible for converting each data object entry into view that can be added to the AdapterView (ListView/RecyclerView).

3.how to support different screen sizes?

1.create a flexible Layout: the best way to create a responsive layout for different screen sizes is to use Constraintlayout as the base layout in your UI. constraintlayout resize the UI as per the screen resolution and relative to each other.
2.create stretchable nine-patch bitmaps.
3. Avoid hard-coded layout size: use wrap_content and match_parent.

4.what is singleton class in android?

A singleton class is a class which can create only one object that can be shared with all other classes.

5.what is View Group? How are they different from View?

View: View objects are the basic building block of User Interface(UI) elements in android. like EditText, Button, checkbox and it is the base class of all UI classes.
ViewGroup: ViewGroup is the invisible container.it holds view and ViewGroup. Ex: Linearlayout is the ViewGroup that contains Button(View)and layout Also.ViewGroup is the base class for Layout.

6.What is recyclerview and what is use of recyclerview method

RecyclerView is a flexible and efficient version of ListView. It is a container for rendering large data sets of views that can be recycled and scrolled very efficiently. RecyclerView is like a traditional ListView widget, but with more flexibility to customize and optimize to work with larger datasets.
1>The constructor of the Adapter gets called which initializes our desired variables.
2>onCreateViewHolder() function is called to create a new view and it’s ViewHolder and initializes some private fields to be used by RecyclerView. The important thing to be noted here is that this function is called only when we really need to create a new view.

3>onBindViewHolder() function is called by RecyclerView to load the data at the specified position. This is where we will pass our data to our empty views.
4>getItemCount() returns the total number of items in the collection that contains the items we want to display.

7.What is the difference between view and Viewgroups?

View	ViewGroup
View objects are the basic building blocks of User Interface(UI) elements in Android.	ViewGroup is the invisible container. It holds View and ViewGroup
View is a simple rectangle box which responds to the user's actions.Examples are EditText, Button, CheckBox etc..	For example, LinearLayout is the ViewGroup that contains Button(View), and other Layouts also.
View refers to the android.view.View class, which is the base class of all UI classes.	ViewGroup is the base class for Layouts. -ViewGroup is a subclass of View

8.What is the difference between Reyclerview and Listview?

Points	Recyclerview	Listview
ViewHolder pattern	The pattern used to reduce calls to findViewById() method	You can easily build a list without using ViewHolder
Adapter	Recyclerview provides Recyclerview.Adapter class	ListView using default adapters ArrayAdapter, CursorAdapter
Item Arrangement	Linearlayoutmanager GridLayoutManager StaggeredGridLayoutManager	In Listview we can create Vertical Listview in simple and less code
Divider animation click event	DividerItemDecoration OnItemTouchListener	setDivider attribute OnItemClickListner

|| MVVM ||

1.what is the difference between MVP and MVVM?

https://www.geeksforgeeks.org/difference-between-mvp-and-mvvm-architecture-pattern-in-android/?ref=rp

1.It resolves the problem of having a dependent view by using the presenter as a communication channel between model and view.
2.The one-to-one relationship exists between the Presenter and the View.
3.The Presenter has knowledge about the View.
4.Model layer returns the response of the user's input to the Presenter which forwards it to View.
5.Presenter handles the application flow and the View is the actual application.
1.This architecture pattern is more event-driven as it uses data binding and thus makes easy separation of core business logic from the View
2.Multiple Views can be mapped with a single ViewModel.
3.ViewModel has no reference to the View
4.After performing operations according to the user's input, the Model layer returns the response to the View.
5.ViewModel is the actual application and View is the interface for the user in order to interact with the app.

2.what is LifeCycle Owner and LifeCycle Observable?

The Lifecycleowner is a class that holds the information about the lifecycle state of a component (like an activity or a fragment) and allows other objects to observe this state.
Event: The lifecycle events that are dispatched from the framework and the `Lifecycle` class. These events map to the callback events in activities and fragments. oncreate(),
onStarted() onResume() onPause() onStop() onDestroy()
State: The current state of the component tracked by the `Lifecycle` object. Created, Started, Paused, Resumed, Destroyed, Stopped

3. What is a Viewmodel?

The viewmodel class is designed to store and manage UI related data in a lifecycle the viewmodel class allows data to survive configuration changes such as screen rotation-landscape and portrait mode. Once the activity is completely destroyed then the onCleared() method is called and clears the memory of all resources and data from the viewmodel.
Note: Not same as onSaveInstanceState()

4.what is livedata?

Livedata is an observable data holder class. This awareness ensures LiveData only updates app component observers that are in an active lifecycle state. livedata follows the observer pattern. Attach the `Observer` object to the `LiveData` object using the `observe()` method. The `observe()` method takes a `LifecycleOwner` object. This subscribes the `Observer` object to the `LiveData` object so that it is notified of changes. You usually attach the `Observer` object in a UI controller, such as an activity or fragment.

Note livedata work on lifecycle owner state event with help of observer() method to know state changes in activity/fragment and update the UI data into onChanged() method using observer()
Test cases: activity stop observing livedata
Case1:when activity is in pause/stop state

5.what is Room database

Database layer on top of SQLite and provide an abstraction layer over SQLite to allow fluent database access. and it's is object-relational mapping library and easy to caching of relevant pieces of data

6.how many components Rom Database does have?

a)Entity: Represents a table within the database
b)Dao: Contains the method used for accessing the database.
c)Database: Contains the database holder and serves as the main access point for the underlying connection to your app's persisted, relational data.
The class that's annotated with `@Database` should satisfy the following conditions:

1. Be an abstract class that extends `RoomDatabase`.
2. Include the list of entities associated with the database within the annotation.
3. Contain an abstract method that has 0 arguments and returns the class that is annotated with `@Dao`.

7. What is the difference between SQLite Database and Room Database?

SQLite Database	Room Database
Deal with row queries	No row queries
There is no compile-time verification of row Sqlite queries	In-Room there is SQL validation at the compile-time
A lot of boilerplate code to convert between SQL queries to the java data object	Maps database object to java object without boilerplate code

SQLite API is low level so more time more effort to build apps	The room when used with ViewModel and Live data makes it easy

8.what is a firebase test lab?

Firebase Test Lab is a cloud-based app-testing infrastructure with one operation where you can test your android and IOS application across a wide variety of devices and device configurations. Firebase test lab provides you with a physical android virtual device for testing.

9.what is testing in android how many types?

1.local unit test- JUnit Mockito
2.Instrumentation test- JUnit Mockito
3.UI test- Expresso

Local Unit test:- Local Unit test is used in local computers using java virtual machine(JVM).it's very fast. Using Unit testing we can check business logic code and functionality .It is logic-based java code. We used JUnit, and Mockito Library.

Instrumentation test:-it's similar to local unit testing.instrumentation can test android specific functionality like as activity fragment service context. Need a real device or emulator for testing.JUnit5 and Mockito for instrumentation testing

UI test:

10.what are fragment and fragment life cycles?

The fragment is known as sub-activity and you can reuse a fragment in multiple activities and combine multiple fragments in a single activity to build a multi-pane UI.

11.what is Viewgroup in android?

Viewgroups is a special view that can contain other views. The ViewGroups is the base class for layout in android like LinearLayout and Relative layout. ViewGroups is generally used to define the layout in which view(widgets) will be arranged listed on the android screen.

12. What is the difference between Reyclerview and Listview?

Points	Recyclerview	Listview
ViewHolder pattern	The pattern used to reduce calls to findViewById() method	You can easily build a list without using ViewHolder
Adapter	Recyclerview provides	ListView using default

	Recyclerview.Adapter class	adapters ArrayAdapter, CursorAdapter
Item Arrangement	Linearlayoutmanager GridLayoutManager StaggeredGridLayoutManager	In Listview we can create Vertical Listview in simple and less code
Divider animation click event	DividerItemDecoration OnItemTouchListener	setDivider attribute OnItemClickListner

|| Dagger 2 ||

1. What is a Dagger?

Dagger is a fully static, compile-time dependency injection framework for both java and android.it uses code generation and based on annotations.the generated code very relatively to read and debug.
Dagger 2 uses the following annotation
1.@Module and @Provides: defines classes and methods which provide dependencies.
2.@Inject: request dependencies. Can be used on a constructor, a field or a method.
3.@Component: enable selected modules and used for performing dependency injection.
https://developer.android.com/training/dependency-injection#kotlin
https://developer.android.com/training/dependency-injection/dagger-android#java
https://developer.android.com/codelabs/android-dagger#0

2.What is the use-case of @Provides Annotation?

@Provides annotation is used on a method in Module class and can return / provide a Dependency object.

3.What is the use-case of @Component Annotation?

@Component is used in Interface or abstract class. Dagger uses this interface to generate an implementation class with fully formed, dependency injected implementation, using the modules declared along with it. This generated class will be preceded by Dagger. For example if I create an interface named ProgramComponent with @Component annotation, Dagger will generate a Class named 'DaggerProgramComponent' implementing the ProgramComponent interface.

4.What is the use-case of @Scope Annotation?

@Scope is an annotation used on Interface to create a new Custom Scope. A Scope declaration helps to keep a single instance of a class as long as its scope exists. For example, in Android, we can use @ApplicationScope for the object to live as long as

the Application is live or @ActivityScope for the object to be available till the activity is killed.

5.What is the use-case of @Inject Annotation in Dagger?

@Inject annotation is used to request a dagger to provide the respective Object. We use @Inject on Constructor, Fields (mostly where constructor is not accessible like Activities, Fragments, etc.) and Methods.

|| MVVM ||

1.how fragment communicate with the help of ViewModel

https://blog.mindorks.com/how-to-communicate-between-fragments

2.What is data binding

https://medium.com/@frankStrangerZ/android-databinding-basics-one-way-two-way-and-handler-2dcef824aa0c

3.Navigation component

https://blog.mindorks.com/jetpack-navigation-component-in-android

4.difference between mvvm and mvp -

https://www.geeksforgeeks.org/difference-between-mvp-and-mvvm-architecture-pattern-in-android/

5.what is role of livedata in viewmodel

https://blog.mindorks.com/understanding-livedata-in-android

6.What is jetpack and explain its component

https://blog.mindorks.com/what-is-android-jetpack-and-why-should-we-use-it

7.what is viewmodel

https://blog.mindorks.com/android-viewmodels-under-the-hood

8.explain component of Android

https://blog.mindorks.com/what-are-android-architecture-components

9.explain file structure of Android

https://guides.codepath.com/android/Android-Directory-Structure

10.What is the solid principle of android

https://medium.com/mindorks/solid-principles-explained-with-examples-79d1ce114ace

11.Difference between livedata and mutablelivedata in android

12.Which one is the better library among glide and picasso for image loading from server

https://stackoverflow.com/questions/41886999/picasso-and-glide-which-one-the-best-for-get-image-form-server

13.Parallel Multiple Network Calls using Kotlin Coroutines

https://blog.mindorks.com/parallel-multiple-network-calls-using-kotlin-coroutines
14.What is coroutine in kotlin
https://blog.mindorks.com/mastering-kotlin-coroutines-in-android-step-by-step-guide
15.what is workmanager in android
https://blog.mindorks.com/integrating-work-manager-in-android
16.Room Database Migrations
https://blog.mindorks.com/room-database-migrations
17.Component of android
https://www.geeksforgeeks.org/components-android-application/
18.About WorkManager details
https://medium.com/androiddevelopers/introducing-workmanager-2083bcfc4712
19.Jetpack Architecture Components in Android
https://www.geeksforgeeks.org/jetpack-architecture-components-in-android/?ref=rp
20.ViewModel in Android Architecture Components
https://www.geeksforgeeks.org/viewmodel-in-android-architecture-components/?ref=rp
21.LiveData in Android Architecture Components
https://www.geeksforgeeks.org/livedata-in-android-architecture-components/?ref=rp
22.Overview of WorkManager in Android Architecture Components
https://www.geeksforgeeks.org/overview-of-workmanager-in-android-architecture-components/?ref=rp
23.Overview of Data Binding in Android Architecture Components
https://www.geeksforgeeks.org/overview-of-data-binding-in-android-architecture-components/?ref=rp
24.UI Components of Android Jetpack
https://www.geeksforgeeks.org/ui-components-of-android-jetpack/?ref=rp
25.Behaviour Components of Android Jetpack
https://www.geeksforgeeks.org/behaviour-components-of-android-jetpack/?ref=rp
26.Lifecycle in Android Architecture Components
https://www.geeksforgeeks.org/lifecycle-in-android-architecture-components/?ref=rp

|| RxJava ||

1.What is Rxjava ?

Why should we use RxJava on Android? Reactive Extensions (Rx) are a set of interfaces and methods which provide a way for developers to solve problems rapidly

, simply to maintain, and easy to understand. RxJava provides just that, a set of tools to help you write clean and simpler code.
note :RxJava provides Java API for asynchronous programming with observable streams.

2.Why use RxJava ?

Answer :Asynchronous streams
Functional approach
Caching is easy
Operators with Schedulers
Using of Subjects

3.What is Reactive Programming ?

In reactive programming consumers react to the data as it comes in.this is the reason why asynchronous programming is also called reactive programming.
Reactive programming allows the propagation of event changes to registered observers.

4.What is Reactivex ?

Reactivex is a Project which provides implementation for this reactive programming concept for different programming languages.
note: The Observer pattern done right. ReactiveX is a combination of the best ideas from the Observer pattern, the Iterator pattern, and functional programming.

5.what is the building block of Rxjava ?

observables representing sources of data
subscribers (or observers) listening to the observables
a set of methods for modifying and composing the data
An observable emits items; a subscriber consumes those items.
https://www.androidhive.info/RxJava/tutorials/

6.What is an Observable in RXJava2?

An Observable simply emits the data to those which subscribed to it. All the emission is done asynchronously to the subscribers. A simple Observable can be created as follows:

```
// RxAndroid Tutorial - Adding Observable
Observable<String> stringObservable = Observable.just("Hello Reactive Programming!");
```

7.What is an Observer in RXJava2?

Observer consumes the data emitted by the Observable. To do this, the Observer needs to subscribe to the Observable. Example shows how to create an Observable in RxJava2.

```
// RxAndroid Tutorial - Adding observer
Observer<String> stringObserver = new Observer<String>() {
        @Override
        public void onSubscribe(Disposable d) {

        }

        @Override
        public void onNext(String s) {
            Toast.makeText(MainActivity.this, s,Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onError(Throwable e) {
        }

        @Override
        public void onComplete() {
        }
    };
```

8.What are the different types of Observables in RxJava?

A)1) single 2) Maybe 3) Completable 4) Observable 5) Flowable

What is a Single in RxJava?

A) A Single in RxJava is an Observable which emits only one item if completed or returns error.

What is Maybe in RxJava?

A) A Maybe in RxJava is used when the Observable needs to emit a value or a no value or an error.

What is Completable in RxJava?

A) A Completable in RxJava is an Observable which just completes the task and does not emit anything if completed. It returns an error if anything fails. It is similar to the reactive concept of runnable.

What is Back Pressure in RxJava?

A) Back Pressure is the state where your observable (publisher) is creating more events than your subscriber can handle.

What is Flowable in RxJava?

A Flowable in RxJava is used when the Observable emits more data than the Observer can consume. In Other words, Flowable can handle back pressure whereas an Observable cannot.

9.What is a Cold Observable?

A Cold Observable is an Observable that does not emit items until a Subscriber subscribes. If we have more than one Subscriber, then the Cold Observable will emit each sequence of items to all Subscribers one by one.

10.What is a Hot Observable?

A Hot observable is an Observer that will emit items.

|| JAVA ||

https://javabeginnerstutorial.com/core-java-tutorial/collection-in-java/
https://hackr.io/blog/java-interview-questions

Q) What JDK, JRE and JVM?

JDK	JRE	JVM
Java Development Kit is a Kit which provides the environment to develop and execute(run) the Java program.	Java Runtime Environment is an installation package which provides environment to only run the java program/application onto your machine	Java virtual machine contains JRE and JDK.Java program you can run using JRF and JDK goes into JVM.JVM executes program line by line it's known Interpreter.
Development Tools(to provide an environment to develop your java programs)	JRE provides the runtime environment to execute/run Java bytecode	JVM is responsible for executing java programs line by line using an interpreter.
JRE (to execute your java program).	JRE contains a set of lib+other files that JVM uses at runtime.	

2. Difference between final, finally finalize?

1.final: final is a modifier applicable for classes, method and variable. If a class is declared as final then we can’t extend that class and we can’t create a child class for that class.
2.if a method is declared as final then we can’t override that method in the child class.
3.if a variable is declared as final then it will become constant and we can’t perform re-assignment for that variable.
2.finally: finally is block always associated with try and catch block to maintain cleanup code and either exception is handled or not it always executes.
3.finalize: finalize() is a method which is always invoked by the garbage collector just before destroying an object to perform cleanup activities.

3. Difference between String and StringBuffer?

String	StringBuffer
String is immutable	StringBuffer is mutable
String is slower than StringBuffer when we perform concatenations.	StringBuffer is faster than String when we concatenate String
String is not synchronized	StringBuffer is synchronized
String is stored in constant string pool	StringBuffer is stored in Heap memory

4. Difference between == operator and equals() method in java?

We use the == operator for reference comparison and equals() method we use for content comparison.

5. What is the difference between StringBuffer and StringBuilder?

StringBuffer	StringBuilder
Every method present in StringBuffer is synchronized	No Method present in StringBuilder is synchronized.
At a time only one thread allow to operate on StringBuffer object StringBuffer is thread-safe	At a time multiple threads are allow to operate on StringBuffer object so it is not thread-safe
It increase waiting time of thread so that performance is low as compare to StringBuilder	Thread are not required to wait to operate on StringBuilder object so performance is high
Introduced in 1.0 version	Introduced in 1.5 version

6.Difference between Interface and abstract class?

Interface	Abstract class

Inside the interface every method is always public and abstract whether we are declaring or not.interface is 100% pure Abstract class.	**Every method present in abstract class need not be public and Abstract. We can create concrete method and abstract method**
We can not declare private and protected access modifiers in Interface	**In abstract class we can not declare final, static, synchronized, natice access modifiers.**
Inside interface we can't declare constructors	**Inside abstract class we can declare constructors**
Inside interface we can't declare instance and static block otherwise we will get compile time error	**Inside abstract class we can declare instance and static block**
For interface variable compulsory we should perform initialization at the time of declaration otherwise we will get compile time error	**For abstract class variable it is not required to perform initialization at the time of declaration**
Every variable present inside interface is always public Static, final whether we are declare or not	**The variables present inside abstract class need not be public, static, and final**
Interface support multiple inheritance	**Abstract class does not support multiple inheritance**

7. What is a constructor in java?
Constructor refers to a block of code which is used to initialize an object. it must have the same name as that of class.it has no return type and it is automatically called when an object is created.
There are two type of constructor
1.Default constructor
2.Parameterized Constructor
8. What is a checked Exception and unchecked Exception?
The exception which is checked by the compiler for a smooth exception of the program at runtime is called Checked Exception.
Eg: IOException
The Exception which are not checked by the compiler are called Unchecked Exception
Eg: ArithmeticException , RuntimeException

8.what is abstraction ?

Abstraction is an oops concept that hides the data value or data members and shows only essential data that need to be shown to the user.Avoids code duplication and increases reusability.
Ex: A car is viewed as a car rather than its individual components

9.what is Encapsulation and Abstraction ?

Encapsulation is basically to bind data members & member functions into a single unit called Class
Abstraction is basically to hide complexity of implementation & provide ease of access to the users
Example Of Abstraction and Encapsulation
smartphone is an abstract where the inner implementation details are encapsulated

10.What is Interface?

An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method bodies. It is used to achieve abstraction and multiple inheritance in Java.

1.Java Interface also represents the IS-A relationship.
2.It cannot be instantiated just like the abstract class.
3.Since Java 8, we can have default and static methods in an interface.
4.Since Java 9, we can have private methods in an interface.
5.Variables declared in the interface are public, static and final by default.
6.Does not allow private and protected access modifiers
Example of Interface:->

```
Public interface Animal{
String name = “sai”;
void eat();
void sleep();
//java 8 we can create static and default method using static and default keyword
static void show(){
System.out.println("interface static method block in java 8 features");
}
default void details(){
System.out.println("interface default method block in java 8 features:"+name);
}// interface end brackets
public class Cat implements Animal{
@Override
public void eat() {System.out.println("dog is eating");;}
@Override
public void sleep() {  System.out.println("dog is sleeping");;}} // cat end brackets
public class InterfaceMainClass{
public static void main(String[]args){
Animal.show(); // static method block we call using interface name
Animal a = new Dogs(); // we can create reference variable of interface but we can;t create
object of interface
a.eat();
a.details();
a.sleep();
}
} // InterfaceMainClass end brackets
```

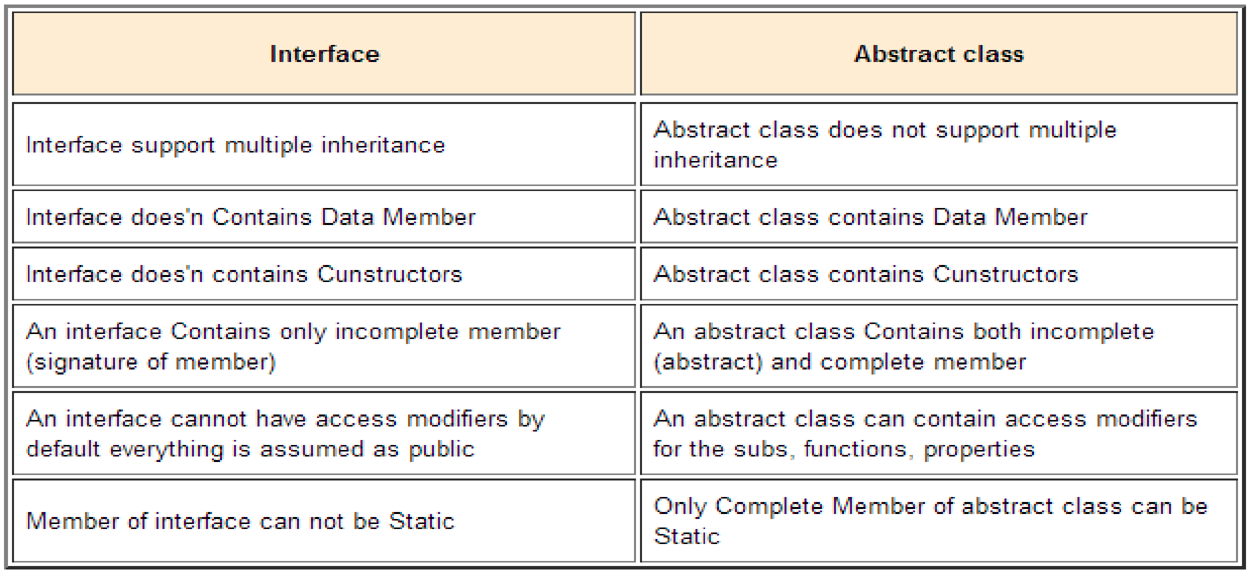
11.Encapsulation vs Data Abstraction

1.Encapsulation is data hiding(information hiding) while Abstraction is detail hiding(implementation hiding).
2.While encapsulation groups together data and methods that act upon the data, data abstraction deals with exposing the interface to the user and hiding the details of implementation.

12.Why are strings Immutable?

Once a value is assigned to a string it cannot be changed. And if changed, it creates a new object of the String. This is not the case with StringBuffer.

13.Difference between Abstract and Interfaces?



Interface	Abstract class
Interface support multiple inheritance	Abstract class does not support multiple inheritance
Interface does'n Contains Data Member	Abstract class contains Data Member
Interface does'n contains Cunstructors	Abstract class contains Cunstructors
An interface Contains only incomplete member (signature of member)	An abstract class Contains both incomplete (abstract) and complete member
An interface cannot have access modifiers by default everything is assumed as public	An abstract class can contain access modifiers for the subs, functions, properties
Member of interface can not be Static	Only Complete Member of abstract class can be Static

14.What is an Exception?

An unwanted, unexpected event that disturbs normal flow of the program is called Exception.Example: FileNotFondException.

15.What is the purpose of Exception Handling?

Ans.The main purpose of Exception Handling is for graceful termination of the program.

16.What is the meaning of Exception Handling?

Exception Handling doesn't mean repairing an Exception, we have to define alternative way to continue the rest of the code normally.

17.What is inner class and when should we go for inner classes?

Some times we can declare a class inside another class such type of classes are called inner classes

18.Explain the concept of boxing, unboxing, autoboxing, and auto unboxing.

Boxing: The concept of putting a primitive value inside an object is called boxing.
Unboxing: Getting the primitive value from the object.
Autoboxing: Assigning a value directly to an integer object.
Auto unboxing: Getting the primitive value directly into the integer object.

19.what is Final modifier?

Final modifiers - once declared cannot be modified. A blank final variable in Java is a final variable that is not initialized during declaration
1.final Classes- A final class cannot have subclasses.
2.final Variables- A final variable cannot be changed once it is initialized.
3.final Methods- A final method cannot be overridden by subclasses.

20.What are Finalize Keywords?

Finalize is a method used to perform clean up processing just before the object is garbage collected.

21.What is Finally keyword?

Finally is a code block which is associated with try catch block and is used to place important code, it will be executed whether an exception is handled or not.

22.What is inner class and when should we go for inner classes?

Some times we can declare a class inside another class such type of classes are called inner classes
Example Of Inner classes:->

```
public class Car{
 void show(){//code here}
// inner class declarations
class Engine{
//code here
} //end inner class
}//end outer class
```

Note:Without an existing Car object there is no chance of an existing Engine object, hence Engine class has been declared inside the Car class.

|| Java Collection ||

1.What is the Collection API ? What is the Collection framework?

It defines a set of classes and interfaces which can be used for representing a group of objects as a single entity.

2.What is the difference between Collections and Collection?

Collection is an interface which can be used for representing a group of individual objects as a single entity and it acts as the root interface of the collection framework.Collections is an utility class to define several utility methods for Collection implemented class objects.

3.Explain about List interface?

List interface is a child interface of Collection interface. This can be used to represent group of individual objects in as a single entity where Duplicates are allowed Insertion order is preserved

1.Duplicates are allowed 2.Insertion order is preserved

4.Explain about Set interface?

Set is a child interface of Collection interface. it can be used to represent a group of individual objects as a single entity where 1.Duplicate objects are not allowed. 2. Insertion order is not preserved

5.What are the differences between arrays and collections? arrays and Vector? arrays and ArrayList?

Arrays	**Collections**
1. Arrays r fixed in size and hence once we created an array we are not allowed to increase or decrease the size based on our requirement.	1. Collections are growable in nature and hence based on our requirement we can increase or decrease the size.
2. Memory point of view arrays are not recommended to use	2. Memory point of view collections are recommended to use.
3. Performance point of view arrays are recommended to use	3. Performance point of view collections are not recommended to use.
4. Arrays can hold only homogeneous elements	4. Collections can hold both homogeneous and heterogeneous elements.
5. Arrays can hold both primitives as well as objects	5. Collections can hold only objects.
6. For any requirement, there is no ready method support compulsory programmer has to code explicitly.	6. For every requirement ready made method support is available. Being a programmer we have to know how to use those methods and we are not responsible to implement those.

6.Explain about ArrayList class?

ArrayList is a Collection which can be used to represent a group of objects as a single entity.
1. it is a implemented class for List interface.Introduced in 1.2 version
2.The underlying data structure is resizable or growable array
3.Insertion order is preserved
4.Duplicates are allowed Heterogeneous objects are allowed
5.null insertion is possible
6.This class implements RandomAccess , Serializable , Cloneable interfaces Best choice for retrieval purpose and worst if our frequent operation is insertion or deletion in the middle.

7.Explain about LinkedList class?

LinkedList is a Collection implemented class which can be used for representing a group of objects as a single entity.LinkedList is the implementation class for List interface
1.Introduced in 1.2 version Underlying data Structure is DoubleLinkedList
2.Allows duplicates
3.Insertion order is preserved. Allows heterogeneous objects
4.null insertion is possible
5.LinkedList class implements Serializable and Cloneable interface but not RandomAccess interface.
6.Best choice if frequent operation is insertion or deletion and objects in middle but worst choice if frequent operation is retrieval.

8.What is the difference between HashMap and Hashtable?

HashMap	Hashtable
No method is synchronized and hance HashMap object is not thread safe	All method are synchronized and it is thread safe
Performance is high	Performance is low
Null insertion is possible for both keys and values	Null insertion is not possible for both key and value violation to nullpointerexception

9.What are the differences between List and Set interface.

List	Set
Insertion order is preserved	Insertion order is not preserved

Duplicate object are allowed	**Duplicate object are not allowed**
The implemented classes are ArrayList, LinkedList, Vector and Stack Classes	**The implemented classes are HashSet, LinkedHashset and Tree**

|| MultiThreading ||

1.What is Multitasking?

Executing several tasks simultaneously is called multitasking.

2.Who uses Thread priority?

Thread Scheduler uses priorities while allocating CPU. The Thread which is having highest priority will get a chance first for execution.

3.What is the difference between process-based and Thread-based Multitasking?

1.Process-based multitasking:- Executing several task simultaneously where each task is A separate independent process such as multitasking is called process based Multitasking.

Example:-While typing a program in the editor we can listen to MP3 audio songs. At the same time we download a file from the net. all these task are executing simultaneously and each task is a separate independent program. Hence it is process based multitasking. It is best suitable at operating system level.

2.Thread-based multitasking:-Executing several tasks simultaneously where each task is a separate independent part of the same program is called Thread-based multitasking. and every independent part is called a thread. This type of multitasking is best suitable at programmatic level.

4.What is Multithreading and explains its application areas?

Executing several thread simultaneously where each thread is a separate independent part of the same program is called multithreading. Java language provides inbuilt support for multithreading by defining a rich library, classes and interfaces like Thread, ThreadGroup,Runnable etc. The main important application area of multithreading are video games implementation, animation development, multimedia graphics etc.

5.Explain about Thread Scheduler?

If multiple threads are waiting for getting the chance for executing then which thread will get a chance is first decided by Thread Scheduler. It is the part of JVM and its behavior is vendor dependent and we can't expect exact output.Whenever the situation comes to multithreading the guarantee behavior is very- very low.

6.Explain the life cycle of a Thread?

Once we create a Thread object then the Thread is said to be in a New/Born state once we call t.start() method then the Thread will be entered into a ready/Runnable state that is Thread is ready to execute. If Thread Scheduler allocates CPU now the Thread will enter into the Running state and start execution of run() method. After completing the run() method the Thread entered into Dead State.

|| Data Structure ||

https://www.interviewbit.com/data-structure-interview-questions/

1.What is an Algorithm?

An Algorithm is a set of well defined instructions in sequence to solve a problem
Qualities of a good algorithm
1.input and output should be defined precisely.
2.eash step in the Algorithm should be clear.

2.What is Data Structure?

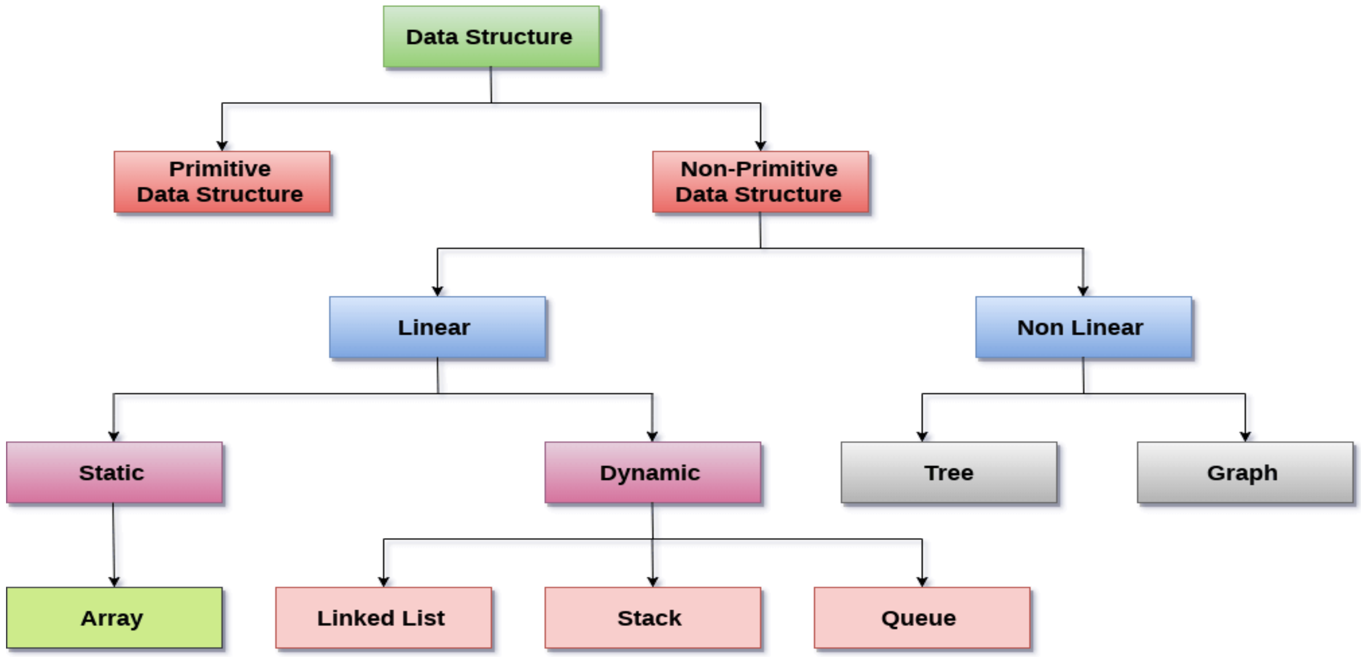
A data structure is a way of storing and organizing data in a computer so that it can be used efficiently. It provides a means to manage large amounts of data efficiently. And efficient data structures are key to designing efficient algorithms.

Advantages of Data Structures:->

Efficiency: Efficiency of a program depends upon the choice of data structures. For example: suppose, we have some data and we need to perform the search for a particular record. In that case, if we organize our data in an array, we will have to search sequentially element by element. Hence, using arrays may not be very efficient here. There are better data structures which can make the search process efficient like an ordered array, binary search tree or hash tables.

Reusability: Data structures are reusable, i.e. once we have implemented a particular data structure, we can use it at any other place. Implementation of data structures can be compiled into libraries which can be used by different clients.

Abstraction: Data structure is specified by the ADT which provides a level of abstraction. The client program uses the data structure through interface only, without getting into the implementation details.

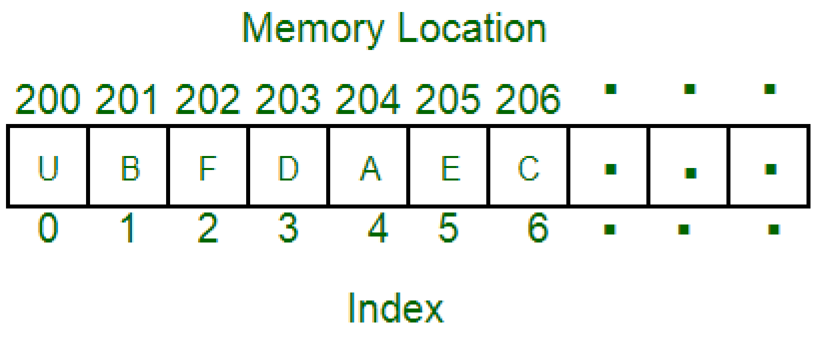


Linear Data Structures:A data structure is called linear if all of its elements are arranged in the linear order. In linear data structures, the elements are stored in a non-hierarchical way where each element has the successors and predecessors except the first and last element.

Type of Linear Data Structures:->
Arrays
Linked List
Stack
Queue

1.Arrays: An array is a collection of similar types of data items and each data item is called an element of the array. The data type of the element may be any valid data type like char, int, float or double.
The elements of the array share the same variable name but each one carries a different index number known as subscript. The array can be one dimensional, two dimensional or multidimensional.
An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the

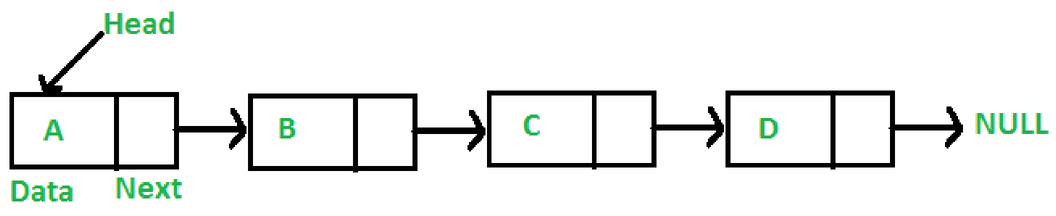
position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).



```
public class ArySum {
public static void main(String[] args) {
        System.out.println("enter the no:");
        Scanner sc = new Scanner(System.in);
        int n =sc.nextInt();
        int ary[] = new int[n];
        int sum=0;
        System.out.println("enter the array of element");
        for (int i=0;i<n;i++){
        ary[i] = sc.nextInt();
        sum = sum + ary[i];
        }
        System.out.println(sum);
        }
        }
```

2.Linked List Data Structure:->

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



|| Spring Boot ||

https://www.interviewbit.com/spring-interview-questions/