1. What is the Activity lifecycle?
2. How services is working in the background?
3. what is a fragment?
4. what is the fragment lifecycle?
5. which method is getting called once in lifecycle even of Activity?
6. what is push notification? how to implement it?
7. what is the use of GIT PUSH?
8. when to use GIT -REBASE?
9. what is the use of GIT STASH?
10. How to update the current branch with the master?
11. how kotlin is different from Java?
12. what is the INLINE function in Kotlin?
13. what is Data class in kotlin?
14. what is the use of the SCOPE function in Kotlin?
15. What is Abstraction in java?
16. what is the interface in java?
17. what is the use of Synchronised keyword in java?
18. what is the internal implementation of Retrofit?
19. what is the difference between PICASSO and GLIDE?

<https://medium.com/nerd-for-tech/android-kotlin-java-2020-2021-interview-questions-f87eef039220>

**onSaveInstanceState():**This callback is meant to retain a **small**amount of UI related data in two situations:

* The app’s process is stopped when it’s in the background due to memory constraints.
* Configuration changes.

**Fragment.setRetainInstance(true)**: The [Handling Configuration Changes documentation](https://developer.android.com/guide/topics/resources/runtime-changes.html#RetainingAnObject) describes a process for storing data during a configuration change using a retained fragment. This sounds less useful than onSaveInstanceState() which covers both configuration changes as well as process shutdown. The usefulness of creating a retained fragment is that it’s meant to retain large sets of data such as images or to retain complex objects like network connections.

**ViewModels only survive configuration change-related destruction; they do not survive the process being stopped.** This makes ViewModels a replacement for using a fragment with setRetainInstance(true) (in fact ViewModels use a fragment with [setRetainInstance](https://developer.android.com/reference/android/app/Fragment.html" \l "setRetainInstance(boolean)" \t "_blank) set to true behind the scenes).

**What is the difference between a val and const?**

Both are immutable once the value is assigned to them we can not modify it. However, we have to assign the value to **const**during the initialization point and **val**during run time.

**Enum**: allow a single instance of each value and can’t encode more information on each type. for eg. in an error case of success, or failure we can not have an associate exception.

**Sealed class**: like abstract classes, sealed classes allows us to represent hierarchies. The child class can be any type of class like a data class, an object, a regular class, or even  
another sealed class. But we have to define hierarchies in the same class or nested classes. If we try to extend a sealed class outside the file cause compile time error.

**What is coroutine in Kotlin?**

Coroutine is nothing but a lightweight thread that executes code asynchronously. They simply simplify asynchronous programming. It is used to manage long-running operations in the background to avoid blocking the main i.e UI thread. It is recommended because of its cheap, lightweight, and built-in cancellation properties.

**Difference between launch and async in coroutine?**

Both create a new coroutine. With the *launch,* we don’t receive any value.

With *async*, we get a return value. It returns a deferred object and we can call await with the object. await will suspend the execution of the coroutine until async finished its work and it will return the value of the coroutine

*launch* throws the exception and *async* hold on to an exception until await is called.

1. What are the features of OOPS :

In simple language just name the feature and define it in one-word

*if we used heavy word / google definition then be ready for the next question !!*

There are four major features in object-oriented programming **encapsulation, inheritance, abstraction, and polymorphism**.

1. **Encapsulation**: fields + methods together as a single unit.

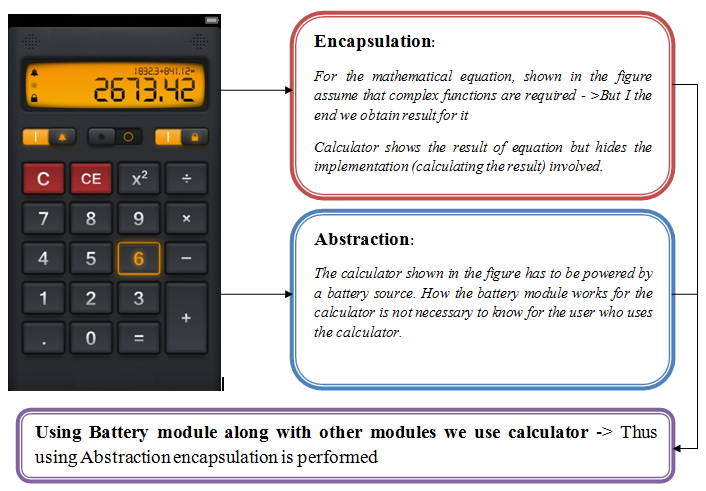
*(Example: Let’s take an example of a mobile device. interaction between a user and a mobile phone as an example of encapsulation. when we accept/decline a call using the red and green button we only know that the red button cut the call and the green button accept, we don’t know how the decline or accept button mechanism work)*

2. **Abstraction:**Abstraction is a process of hiding the implementation details and showing only functionality to the user.

*Example: Baking a cake. If you are following a recipe to bake a cake, you are using abstraction. In this example, you’re following only the necessary steps to prepare and bake the batter.*

***Most of the time interviewer confuses us in encapsulation and abstract class***

Give the below example to the interviewer he will not question you further !!



3. **Polymorphism**: Depending on the situation decide what to do !! it is a concept by which we can perform a *single action in different ways*.

*The best example of polymorphism is****human behavior****. One person can have different behavior. For example, a person acts as an employee in the office, a customer in the shopping mall, a passenger on a bus/train, a student in school, and a son at home.*

4. **Inheritance**: one class acquires the property of another class.

*In the real world, a child inherits the features of its parents such as the beauty of the mother and the intelligence of the father.*

