Md Rahman

1. What is the difference in lateinit and lazy initialization?

2. What are the advantages of Kotlin over Java?

3. What are checked and unchecked exceptions? Which is not supported by Kotlin?

4. What is Model View Presenter?

5. What is Model View ViewModel?

1. Difference between lateinit and lazy?

Lateinit stands for late initialization, used for vars. We specify the var type and the keyword lateinit, telling the system that our var will hold a value in the future.

Lazy can only be used for vals. Lazy() is a function that takes a lambda to initialize the value, and any subsequent calls simply returns that value.

2. What are the advantages of Kotlin over Java?

Kotlin has tried eliminating NPE’s, has in-line functions and extension functions, has inferred typing, and less boilerplate code

3.What are checked and unchecked exceptions? Which is not supported by Kotlin?

Checked exceptions are found at compile time. If some code inside a function throws an exception, the method must either handle the exception or throw (specify) the exception. Typically we handle these exceptions with a try catch block.

Unchecked exceptions are not checked at compile time; instead at runtime. In Java, these are Error and Runtime Exceptions. In Kotlin, all exceptions are unchecked.

4. What is Model View Presenter?

MPV is a common design architecture. There are four key components:

Presenter - contains the UI logic for the View and controls the model.

Contract - Interfaces the presenter uses to talk to the view.

View - U.I elements

Model - Pojo, data classes, data sources

There are also two variations:

Passive view - View contains almost zero logic and there is a clean separation of the View and Model; the presenter is the middleman connecting the view and the model. One advantage of this is it allows for better testing, but because of no data binding we need more code.

Supervising Controller - View binds to the Model directly, and the presenter handles user events. While this is harder to test, due to databinding the amount of code needed is reduced.

5. What is Model View ViewModel?

Data-Binding - A part of architecture components (a 3rd party library) that helps minimize the code necessary to bind application logic and layouts.

View and Model have 1 to many relationship - View can have multiple view-models, but not-vice versa.

In Android, the main purpose behind MVVM is to abstract our code to XML.

Uses Live Data

In MVVM the logic is stored in the presenter and the view is completely isolated from the model. The presenter is used to represent an abstract view of the user interface. The view is no longer passive, it contains behaviors, events and data binding information. The viewmodel in MVVM is responsible for presentation separation as well as exposing methods and commands to manage the state of a view and manipulate the model. The view and viewmodel communicate using methods and events.