**Android Architecture (Jetpack)**

**1**

* Breif Overview of Architecture Component.
* Problem with existing system.
* What is Jetpack?

Androd Architecture Component.

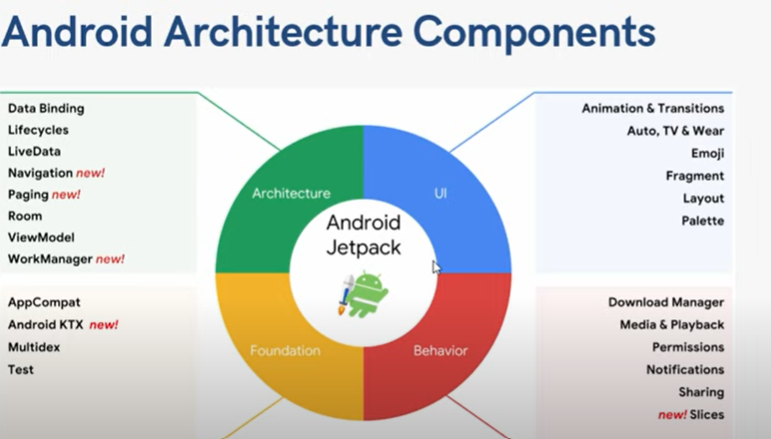
* Android has component-based architecture –where you use different components like Activity,Fragments,Service etc.
* It is Android that manage these Components for you.
* Till now ,we have been writing code in our android activities or fragment for all purpose –wheather it is data fetching or storing data in SQLite.
* This is where the Problems lie.

**Pattern**

* MVC
* MVP
* MVVM

What you want this architecture component.

* Separation of concerns.
* Data Driven Applications.



**Summaries**

* Collection of library that helps build robust,testable and maintainable android Apps.
* Less BoilerPlate code – Room,Navigation Components, Paging etc.
* High Quality,Production Ready Apps.

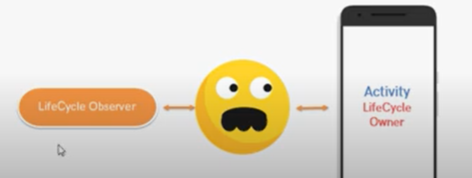
**2**

**Component Likes that.**

* Data Binding.
* ViewModel
* LiveData
* Room Database
* Navigation Component
* Work Manager
* Paging

**Life Cycle Aware Component**

* Most of the code written inside Activity LifeCycle Methods – onCreate,onResume,onPause etc.Due to this, Acitvity has multiple responsibilities.
* But there are scenarious where we want to take actions based on the activity lifecycle.
* For e.g.
* Access the User’s Location
* Playing Video.
* Downloading Images.
* **Components Name**
* LifeCycle Owner
* LifeCycle Observer

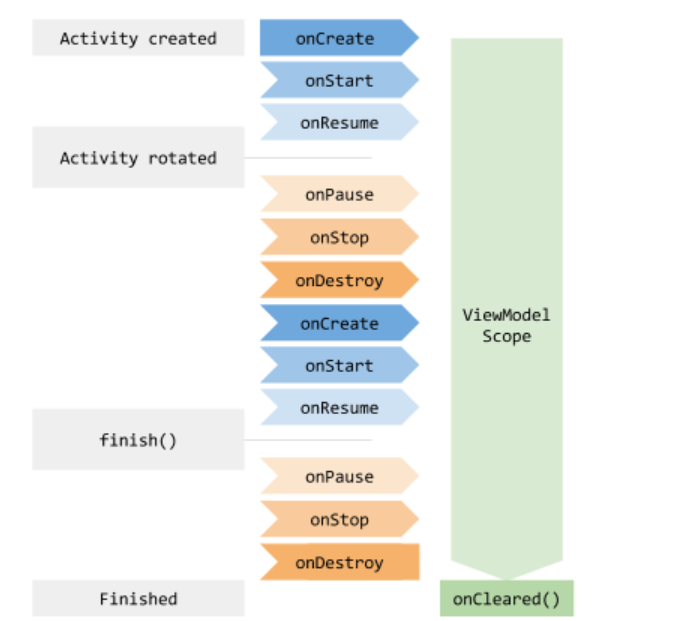


**3**

**View Model**

Link-><https://developer.android.com/topic/libraries/architecture/viewmodel>

* Model for your views such as Activity or Fragment.
* ViewModels are lifecycle aware.
* Data required for your screen is stored at one place i.e. ViewModel. It may involve formatting that data in a particular format,accumulating data, any logic in displaying this data in your UI.



**AGENDA**

* LiveData
* Mutable LiveData & LiveData
* Examples

**LiveData**

Link-> <https://developer.android.com/topic/libraries/architecture/livedata>

* Observable Data Holder Class
* Lifecycle aware

**4**

**Data Binding**

* Support library that allows you to bind UI components in your layouts to data sources in your app using a declarative rather than programmatically.



* Benifiets
* findViewById
* Null Pointer Exception
* Performance & no Memory Leaks

**Data Binding With ViewModel**

**5**

* **Room Database**

Link-><https://developer.android.com/training/data-storage/room>

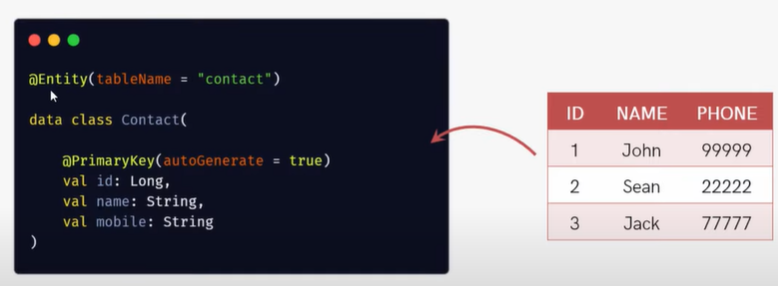
* Need of Room Library
* Examples
* Before that use
* Abstraction over SQLite
* Less Boilerplate
* Compile Time Verification of SQL Queries



**5 way of achieve Room Database**

* Entities(Table)
* DAO(Data Access Objects)
* Database
* Type Convertors
* Migrations

**Entities**

****

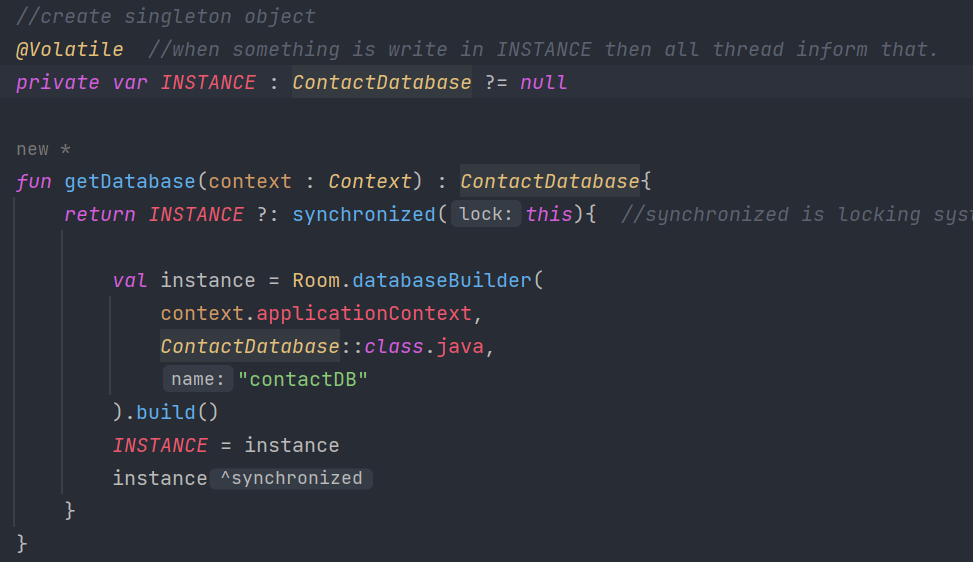
**6**

**DAO**

* Data Access Objects
* Interface containing methods to access database – CRUD operations
* We can define multiple DAOs



**Database**

****

**7**

**Type Convertors**

* SQLite only supports-

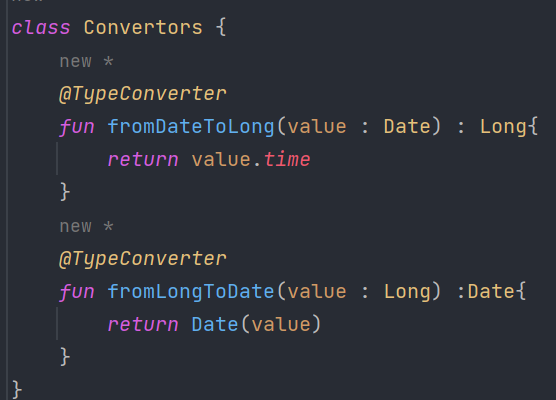
1.NULL

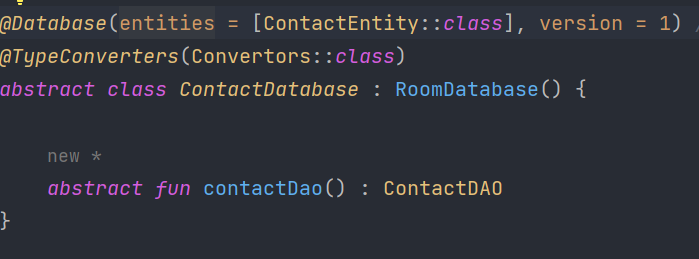
2.INTEGER

3.REAL

4.TEXT

5.BLOB



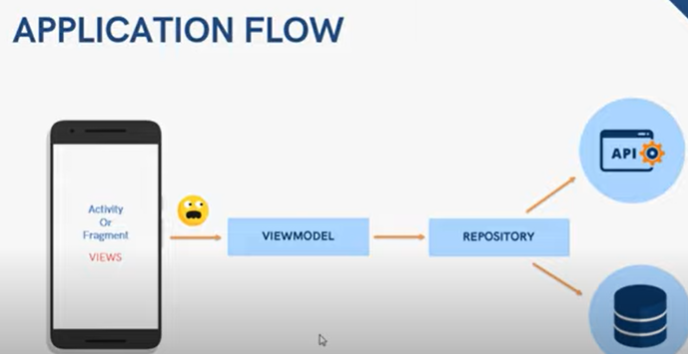


**Migration**

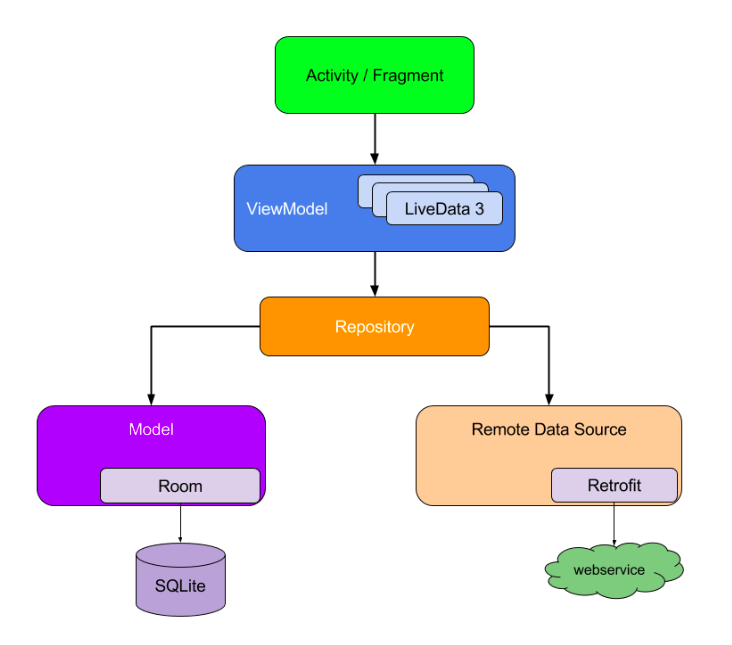
Up gradation in Database with version change without lose data.

* **Repository Design Pattern**
* Not Specific to Android – It is a common design pattern.
* Reusable solution to a commonly occurring problem in software design.

**8**



* Repository mediates between the domain layer & data mapping layers.
* Client constructs the query & submits it to the repository.
* Repository encapsulates all of this logic of resolving that query.



**AGENDA**

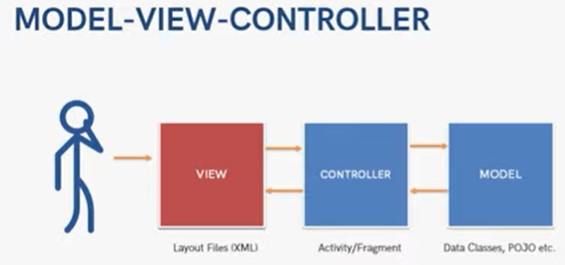
* **Android Architecture Patterns**
* **MVC,MVP,MVVM in Android**
* **Examples**

**9**

**NEED OF ARCHITECTURE**

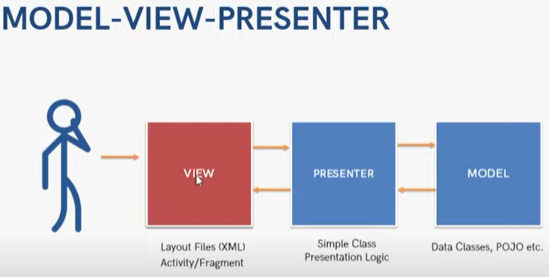
* Scalability & Maintainability of an application.
* To achieve this-we implement architecture pattern – MVC,MVP,MVVM etc.
* Architecture patterns mainly focuses on-
* Separation of Concerns.
* Unit Testing

**MVC (Model-View-Controller)(Default pattern in Android)**



* Highly depend view and controller
* Speration of Concerns violate in this case.

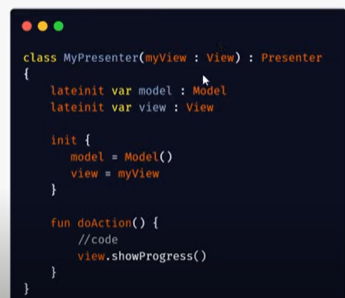
**MVP(Model-View-Presenter)**



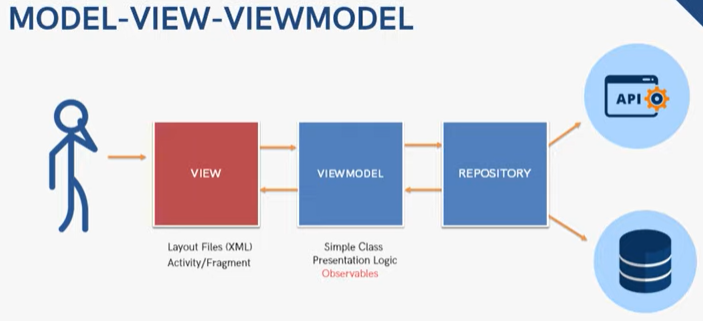
* Independent view and presenter(file)
* Sepration of concerns not violate in this case.

**10**

**Example of MVP (Android Specific code is not allowed in presenter )**



**MVVM(Model-View-ViewModel)**



* Room Database and/or Retrofit Setup
* Repository
* LiveData,ViewModel with ViewModelFactory
* Activity/Fragment with Data Binding