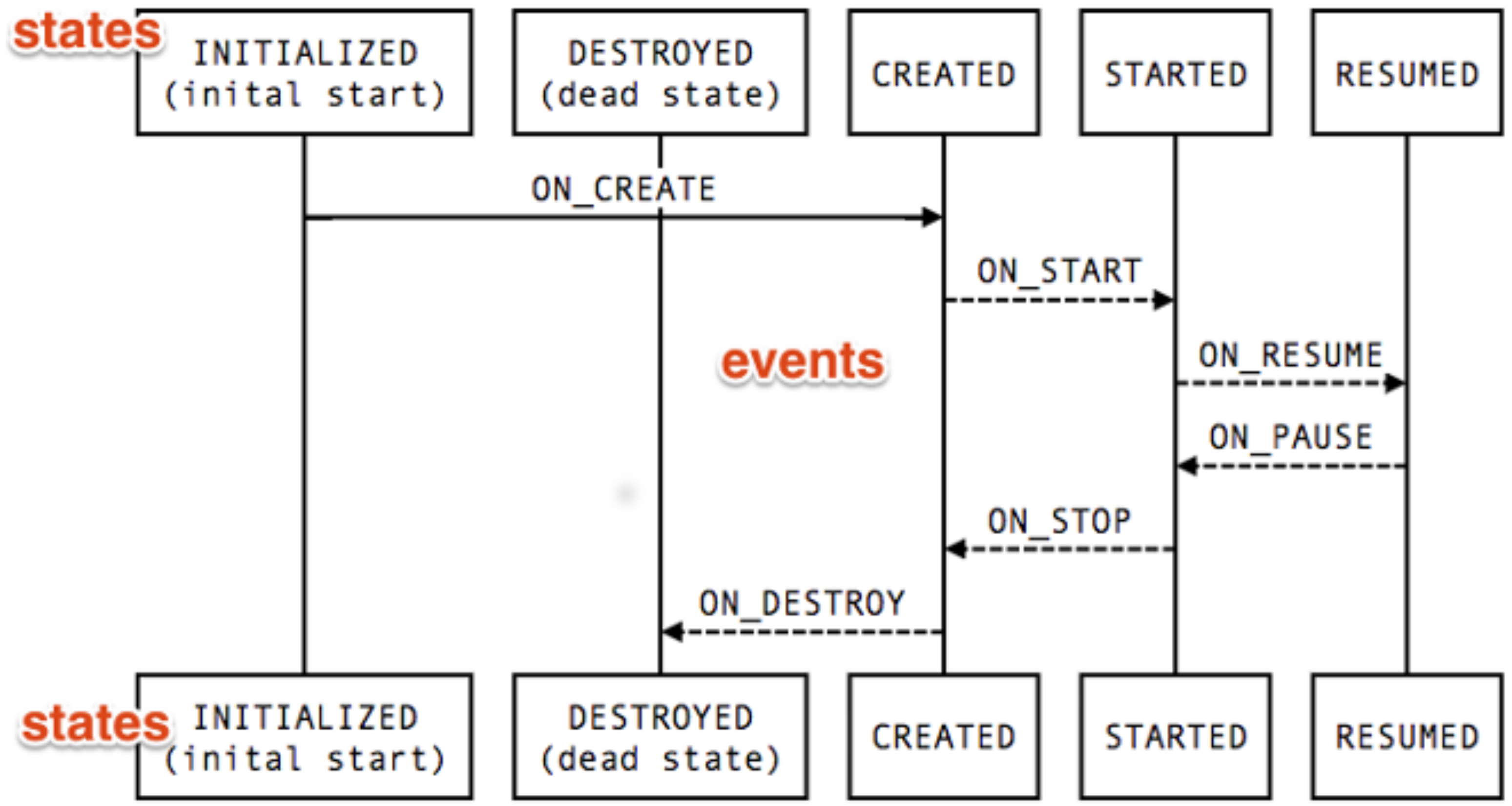
Android Architecture Components (Arch)

- **Lifecycle**
- **ViewModel**
- **LiveData**
- **Room** an abstraction over SQLite -> covered tomorrow

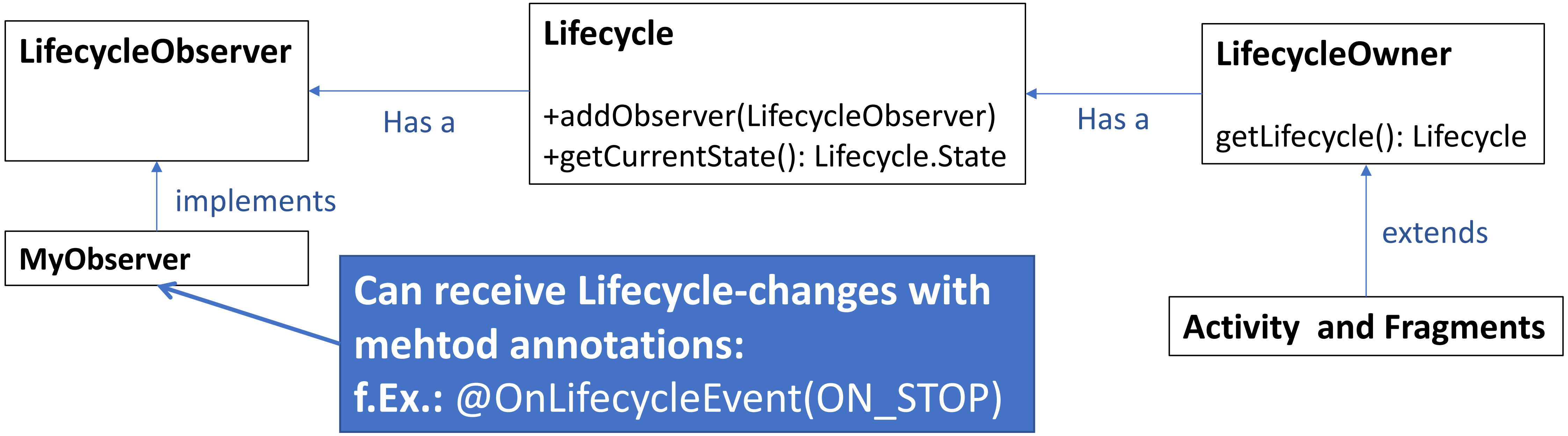
Lifecycle

The Lifecycle Component holds the state of a component:



Lifecycle

- **LifecycleOwner**: interface with one function: getLifecycle()
- **LifecycleObserver**: interface which enables to observe Lifecycle-events


LifecycleObserver
Lifecycle
+addObserver(LifecycleObserver)
+getCurrentState(): Lifecycle.State
LifecycleOwner
getLifecycle(): Lifecycle
Has a
Has a
implements
extends
MyObserver
Activity and Fragments
Can receive Lifecycle-changes with mehtod annotations:
f.Ex.: @OnLifecycleEvent(ON_STOP)

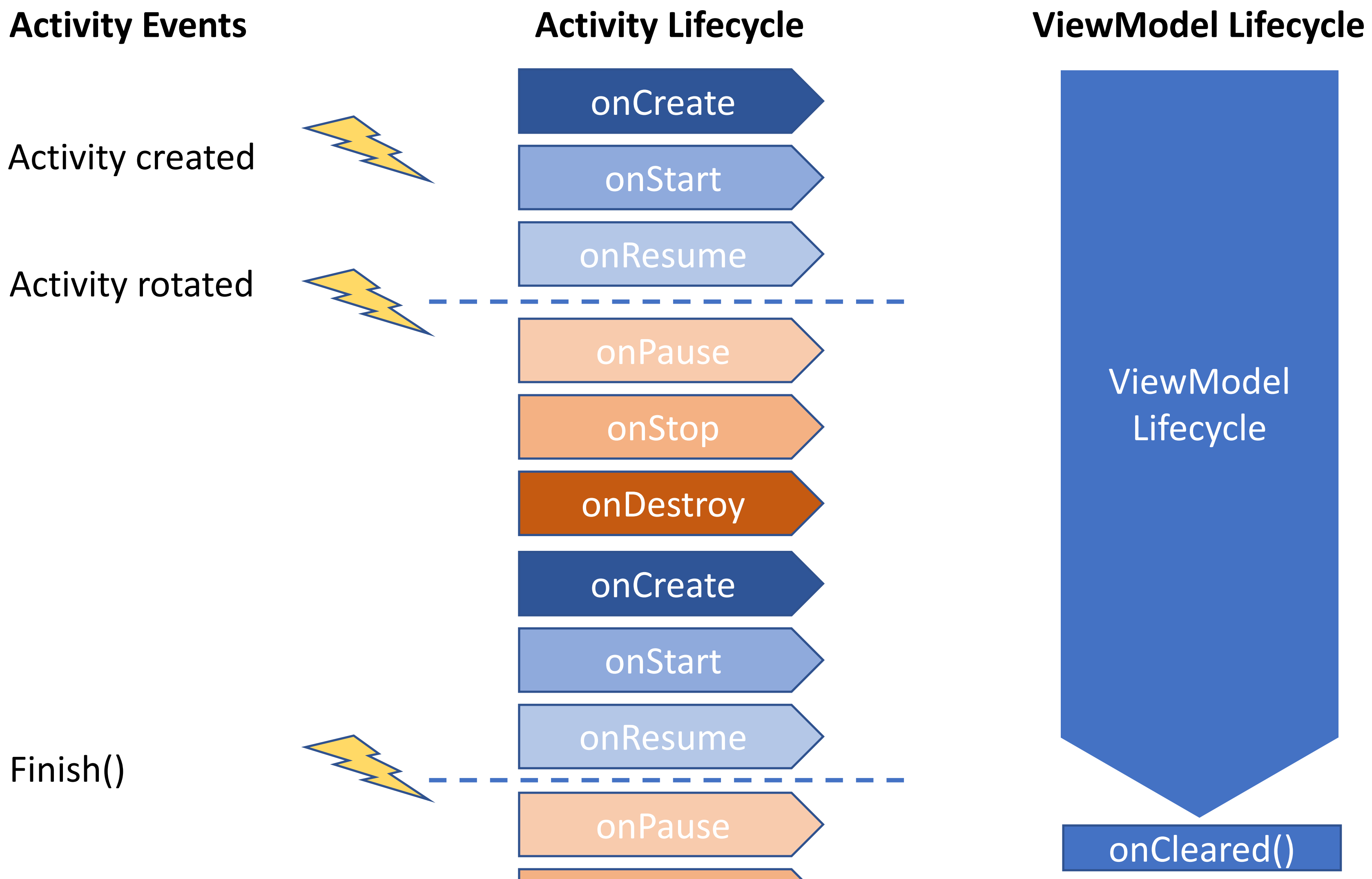
ViewModel

- Enables us to use the MVVM – Architecture pattern
- Designed to store and manage UI-related data
- React to user actions
- Removes responsibility from UI-Controllers (Activities and Fragments)
  - Better testability, no bloated Activities or Fragments
- Is Lifecycle-aware

Creating our own LoginViewModel

```
class LoginViewModel: ViewModel{…
```

ViewModel & Configuration Changes



- Configuration Changes
- Pressing Back (Pop Fragment)
- Pressing Back (Pop Activity)
- Kill the app from recents
- Framework kills the app

ViewModel & Inter-Fragment Communication

- Scopes of ViewModel
  - Activity

```
ViewModelProviders.of(activity,
viewModelFactory).get(NewsViewModel::class.java)
```

=> Share data between fragments without Bundle or Parcelable ☺

  - Fragment

```
ViewModelProviders.of(fragment,
viewModelFactory).get(NewsViewModel::class.java)
```

Viewmodel Constraints

- No reference to a View
- No reference to a View’s context
- No imports from android.* unless android.arch
- Pure Java/Kotlin libraries
- Per screen -> one ViewModel

LiveData & LifeCycle

- ViewModel does not hold a reference to View (Activity or Fragment)
- How to pass data to the View?

=> LifeData objects

- LiveData
  - Observer pattern
  - Lifecycle aware data-objects (only update data, when active lifecycle state)
  - LiveData.observe(Lifecycle.Owner, Lifecycle.Observer)

LiveData & LifeCycle

- LifeData is a Lifecycle-aware data-objects with observer-pattern
  - In the ViewModel:

```
private val someMutableLifeData: MutableLiveData<ViewState> =
MutableLiveData()
val someLifeData:LiveData<ViewState> = someMutableLifeData
```

  - In Fragment or Activity

```
newsViewModel.someLifeData.observe(this,this::handle)
```

LiveData Transformations - SwitchMap

- Transform LiveData of one type to LiveData of another type

- **Transformations.switchMap()**

```
val userId : MutableLiveData<String> = MutableLiveData()

val user : LiveData<User> = Transformations.switchMap(userId, {id ->
getUser(id)})

fun getUser(userId: String): MutableLiveData<User>{...}
```

LiveData Transformations - Map

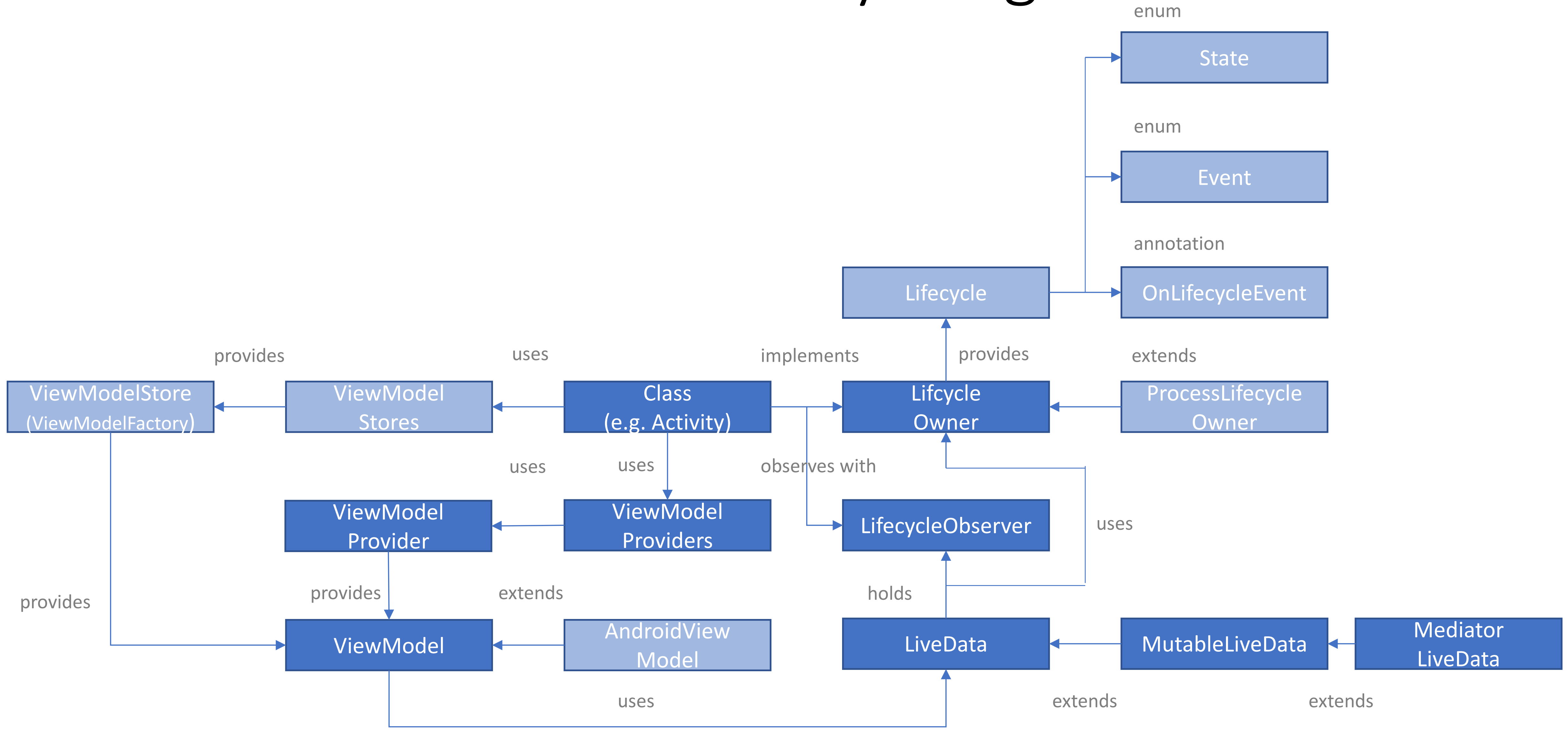
- Make changes to the LiveData before exposing it

- **Transformations.map()**

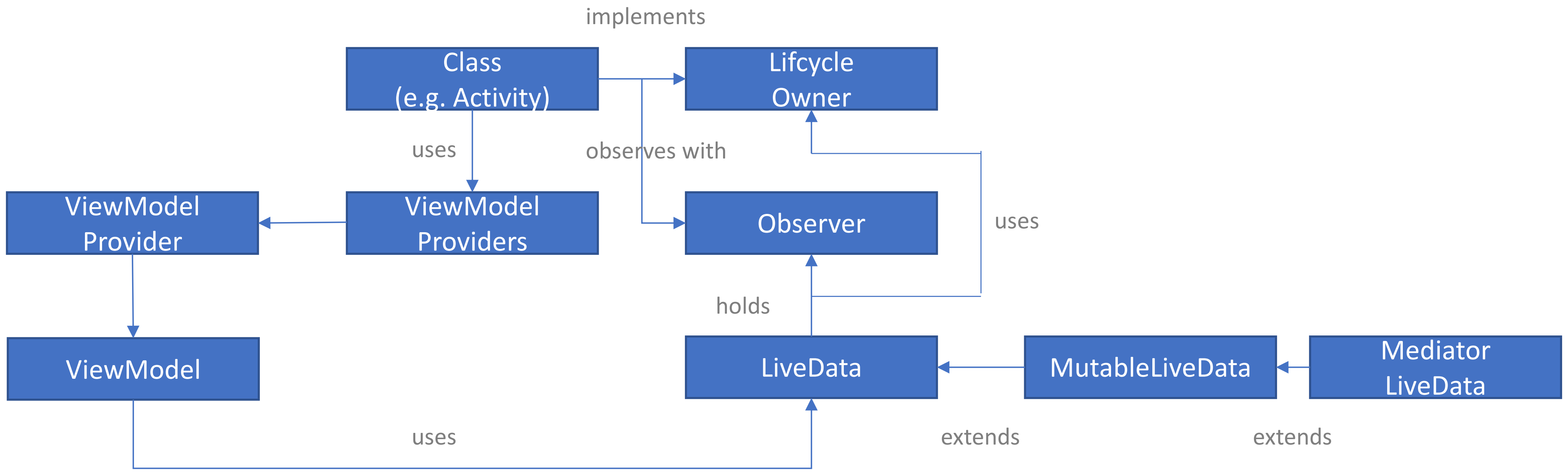
```
val redditData: LiveData<MutableList<RedditNewsData>> = mutableRedditData

private val redditAutors = Transformations.map(redditData,{ input:
MutableList<RedditNewsData>?->
    input?.map { redditNewsData -> redditNewsData.author }})
```

Class overview – how everything is connected



Class overview – Things we deal with



Links und Quellen

- https://developer.android.com/topic/libraries/architecture/livedata.html