LiveData

LiveData 是 Android Jetpack Lifecycle 组件中的内容。属于官方库的一部分，Kotlin/Java 均可使用。
一句话概括 LiveData：LiveData 是可感知生命周期的，可观察的，数据持有者。它的能力和作用很简单：更新 UI。

它有一些可以被认为是优点的特性：
1、观察者的回调永远发生在主线程。
2、仅持有单个且最新的数据。
3、自动取消订阅。
4、提供「可读可写」和「仅可读」两个版本收缩权限。
5、配合 DataBinding 实现「双向绑定」。

以下也是 LiveData 的特性，但我不会将其归类为「设计缺陷」或「LiveData 的缺点」。作为开发者应了解这些特性并在使用过程中正确处理它们。
1、value 是 nullable 的。
2、在 fragment 订阅时需要传入正确的 lifecycleOwner。
3、当 LiveData 持有的数据是「事件」时，可能会遇到「粘性事件」。
4、LiveData 是不防抖的。
5、LiveData 的 transformation 工作在主线程。

LiveData 小结
1、LiveData 作为一个 可感知生命周期的，可观察的，数据持有者，被设计用来更新 UI。
2、LiveData 很轻，功能十分克制，克制到需要配合 ViewModel 使用才能显示其价值。
3、由于 LiveData 专注单一功能，因此它的一些方法使用上是有局限性的，即通过设计来强制开发者按正确的方式编码（如观察者仅在主线程回调，避免了开发者在子线程更新 UI 的错误操作）。
4、由于 LiveData 专注单一功能，如果想在表现层之外使用它，MediatorLiveData 的操作数据的能力有限，仅有的 map 和 switchMap 发生在主线程。可以在 switchMap 中使用协程或 RxJava 处理异步任务，最后在主线程返回 LiveData。如果项目中使用了 RxJava 的 AutoDispose[7]，甚至可以不使用 LiveData，关于 Kotlin 协程的 Flow，我们后文介绍。
5、笔者不喜欢将 LiveData 改造成 bus 使用，让组件做其分内的事（此条属于个人观点）。

StateFlow

StateFlow 与 LiveData 十分像，或者说它们的定位类似。
StateFlow 与 LiveData 有一些相同点：
1、提供「可读可写」和「仅可读」两个版本（StateFlow，MutableStateFlow）。
2、它的值是唯一的。
3、它允许被多个观察者共用 （因此是共享的数据流）。
4、它永远只会把最新的值重现给订阅者，这与活跃观察者的数量是无关的。
5、支持 DataBinding。
它们也有些不同点：

1、必须配置初始值。
2、value 空安全。
3、防抖。

MutableStateFlow 构造方法强制赋值一个非空的数据，而且 value 也是非空的。这意味着 StateFlow 永远有值。

SharedFlow

与 SateFlow 一样，SharedFlow 也有两个版本：SharedFlow 与 MutableSharedFlow。
不同点：
1、MutableSharedFlow 没有起始值。
2、SharedFlow 可以保留历史数据。
3、MutableSharedFlow 发射值需要调用 emit()/tryEmit() 方法，没有 setValue() 方法。

与 MutableStateFlow 不同，MutableSharedFlow 构造器中是不能传入默认值的，这意味着 MutableSharedFlow 没有默认值。
SateFlow 与 SharedFlow 还有一个区别是 SateFlow 只保留最新值，即新的订阅者只会获得最新的和之后的数据。
而 SharedFlow 根据配置可以保留历史数据，新的订阅者可以获取之前发射过的一系列数据。

StateFlow 和 ShareFlow 小结
1、Flow 可分为生产者，消费者，中介三个角色。
2、冷流和热流最大的区别是前者依赖消费者 collect 存在，而热流一直存在，直到被取消。
3、StateFlow 与 LiveData 定位相似，前者必须配置初始值，value 空安全并且默认防抖。
4、StateFlow 与 SharedFlow 的使用场景不同，前者适用于「状态」，后者适用于「事件」