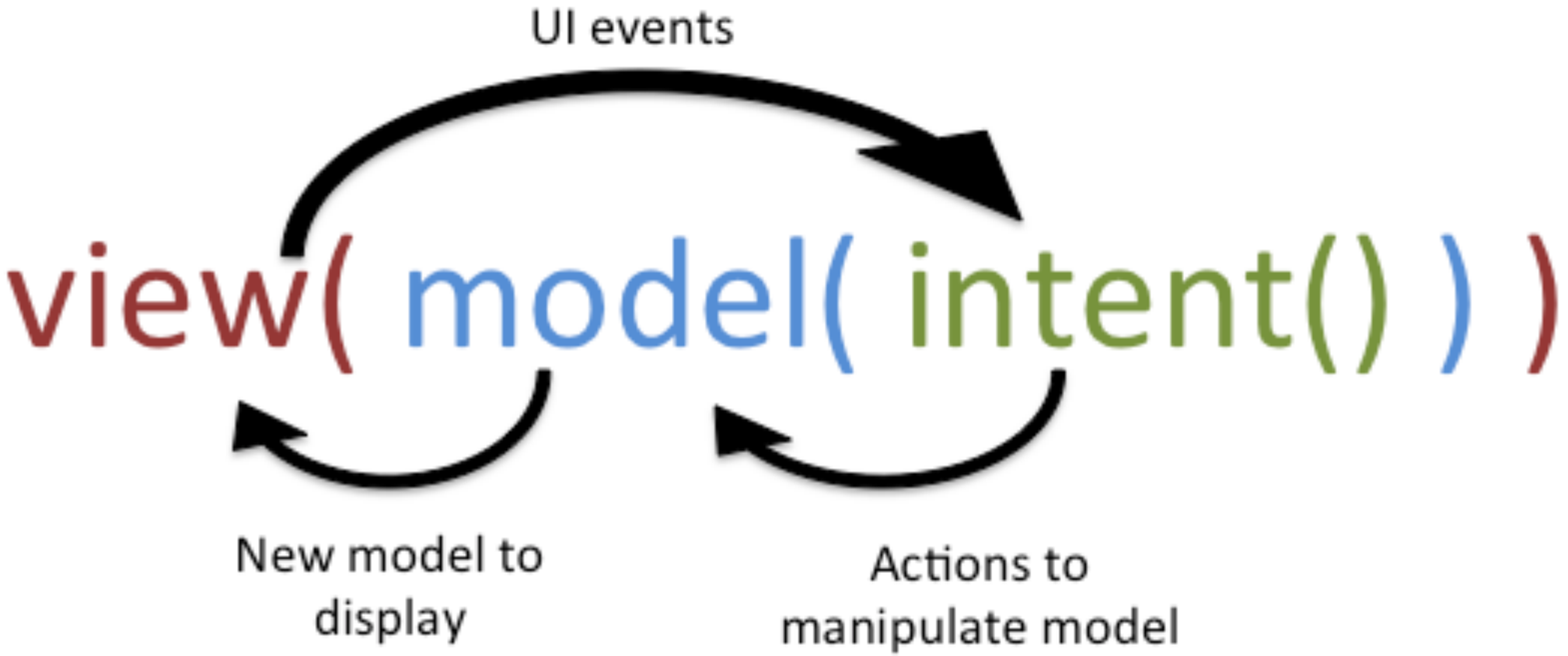
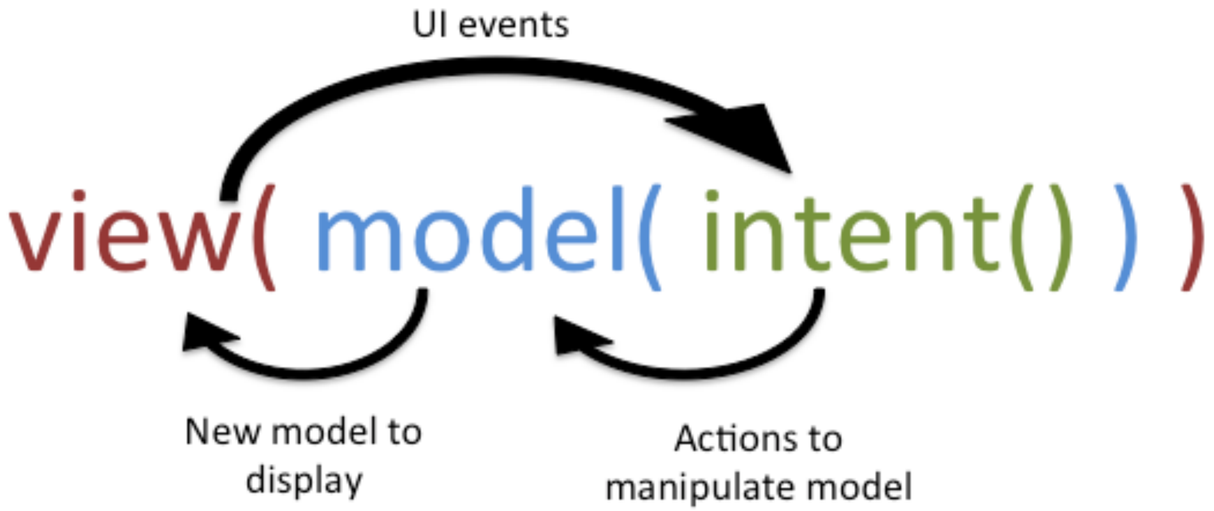
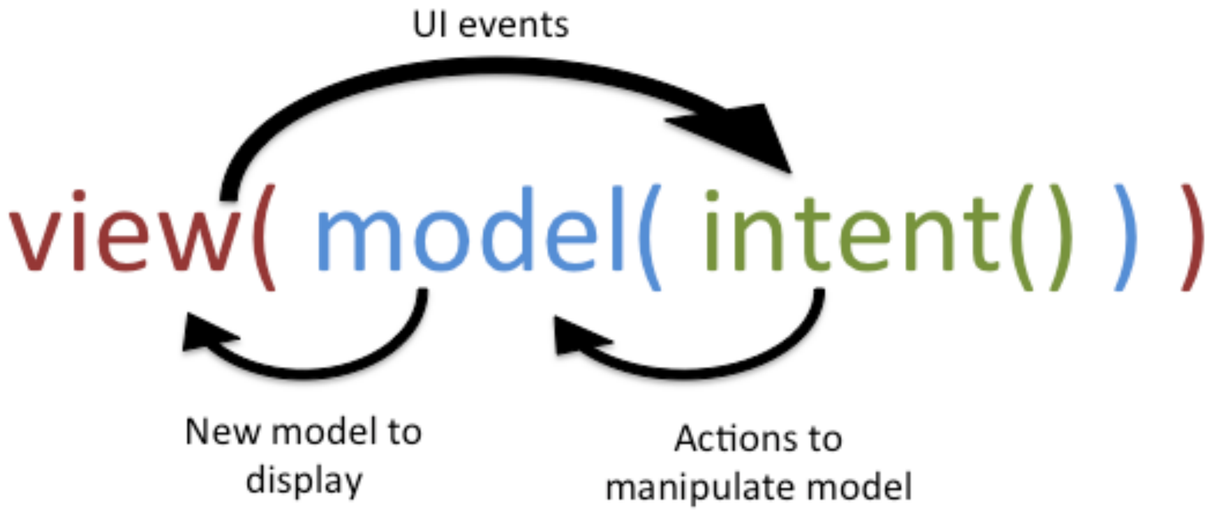
Simple MVI Architecture for Android



http://hannesdorfmann.com/android/mosby3-mvi-1

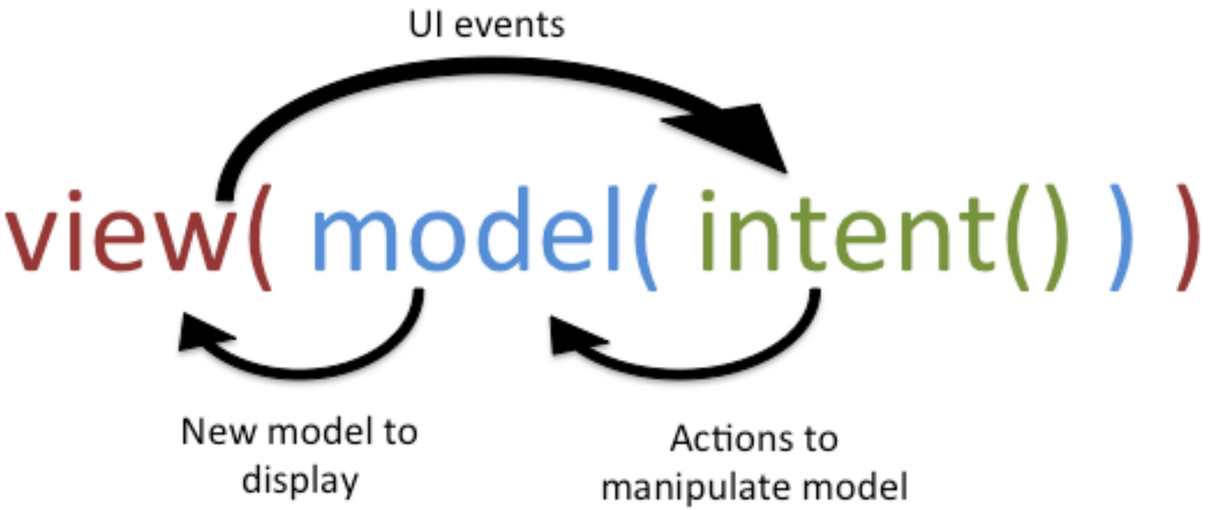

UI events
view(model(intent()))
New model to display
Actions to manipulate model

$$f(x)$$



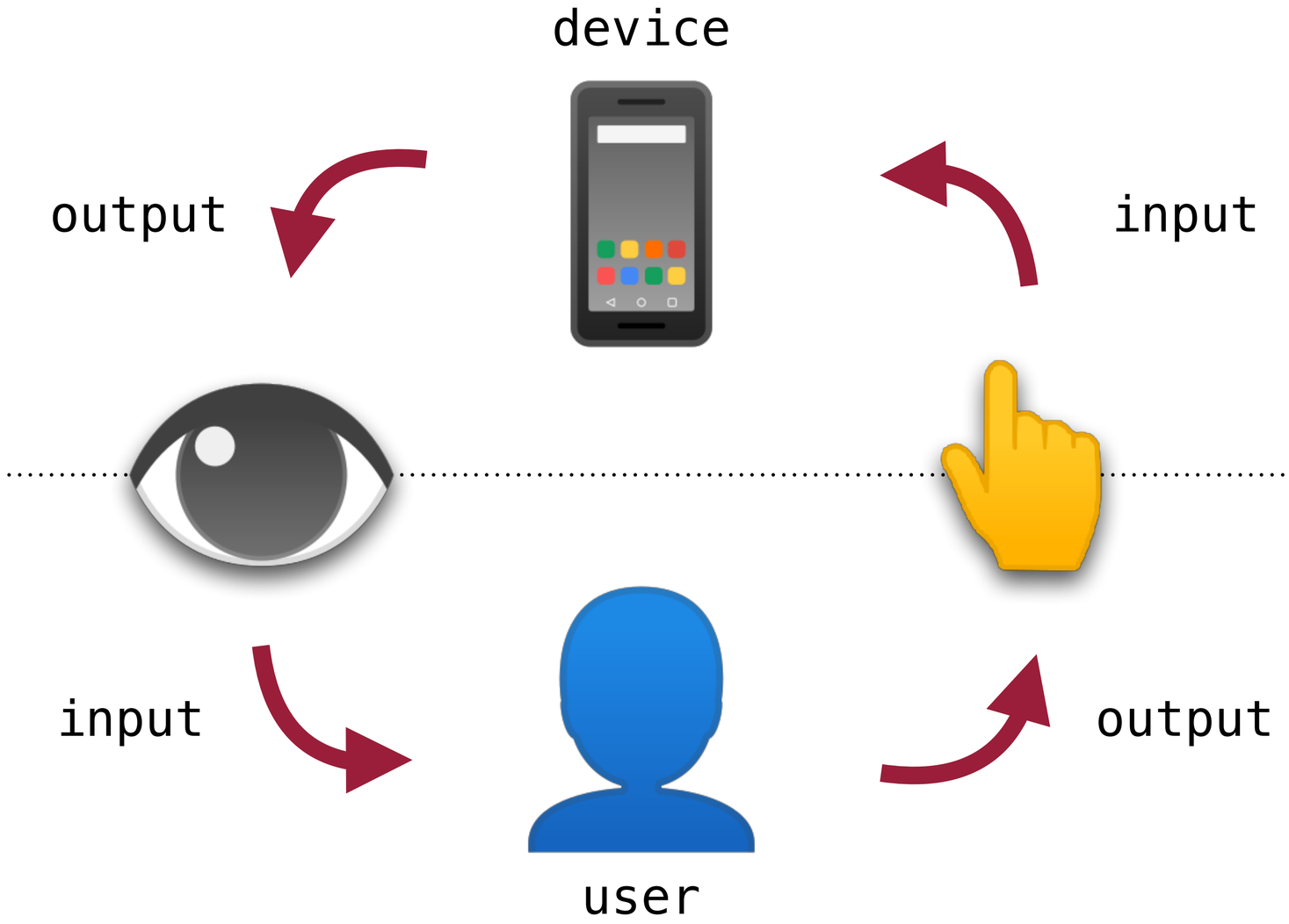
$$f(x)$$

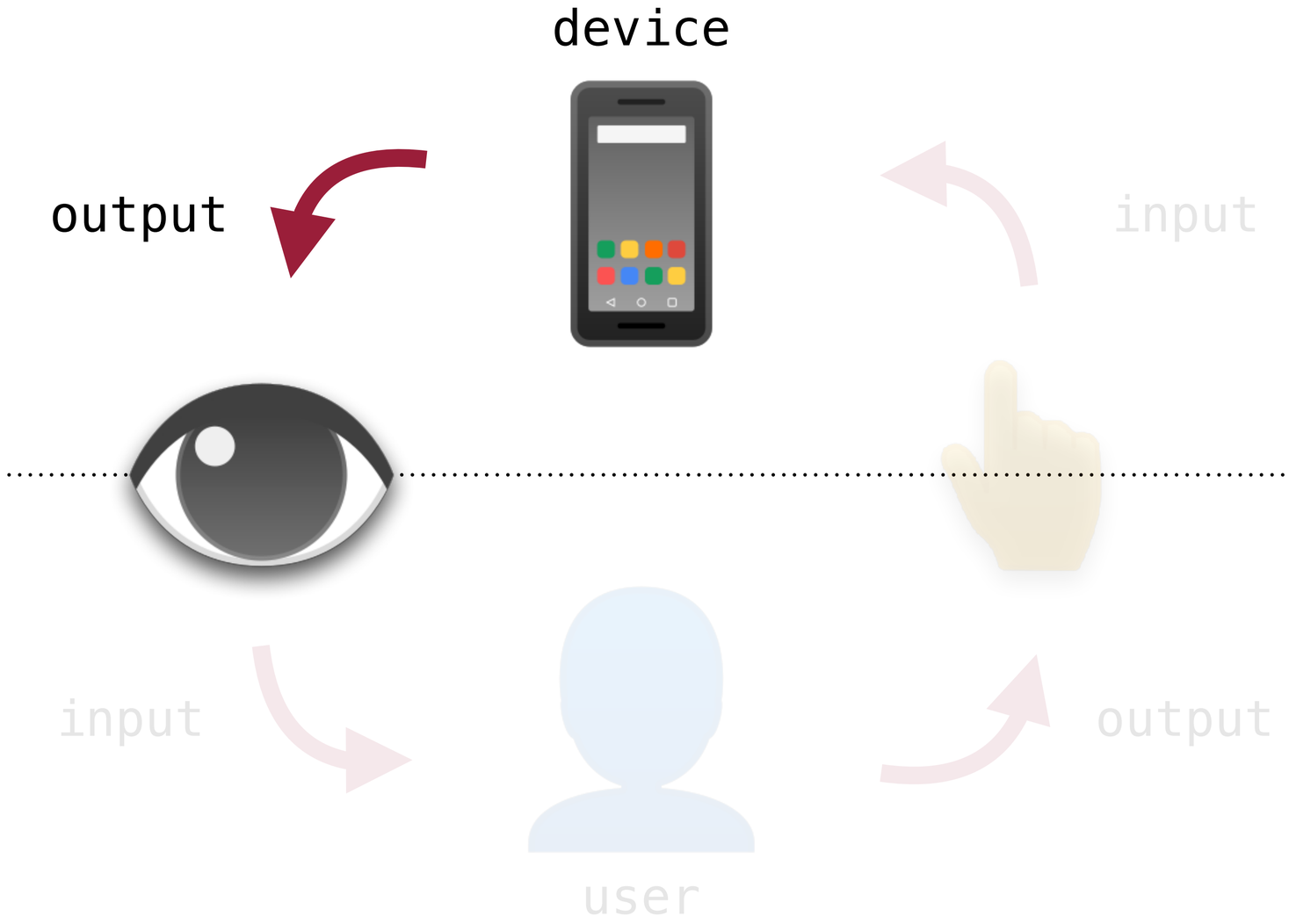
https://xkcd.com/1312

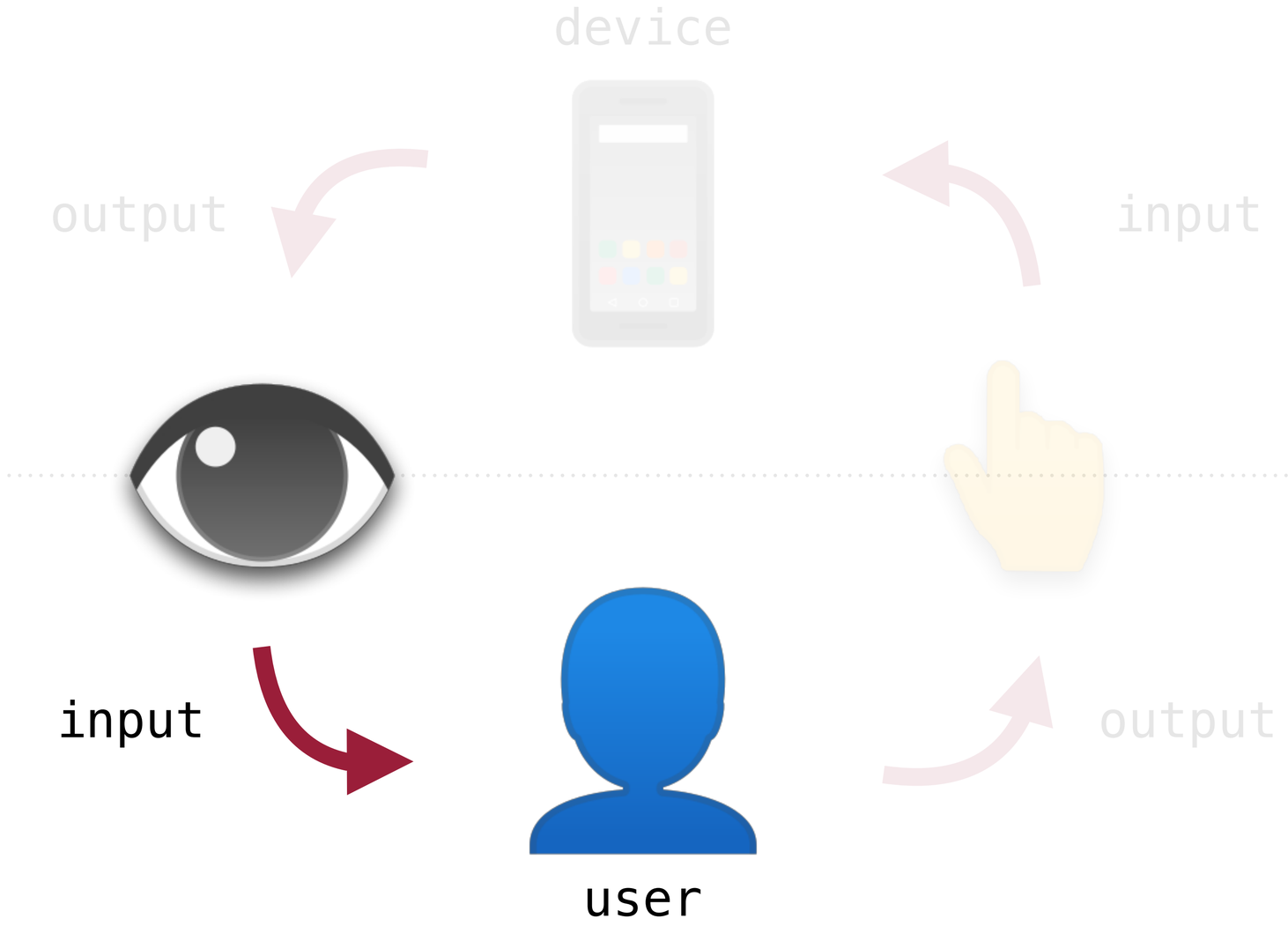

UI events
view(model(intent()))
New model to display
Actions to manipulate model

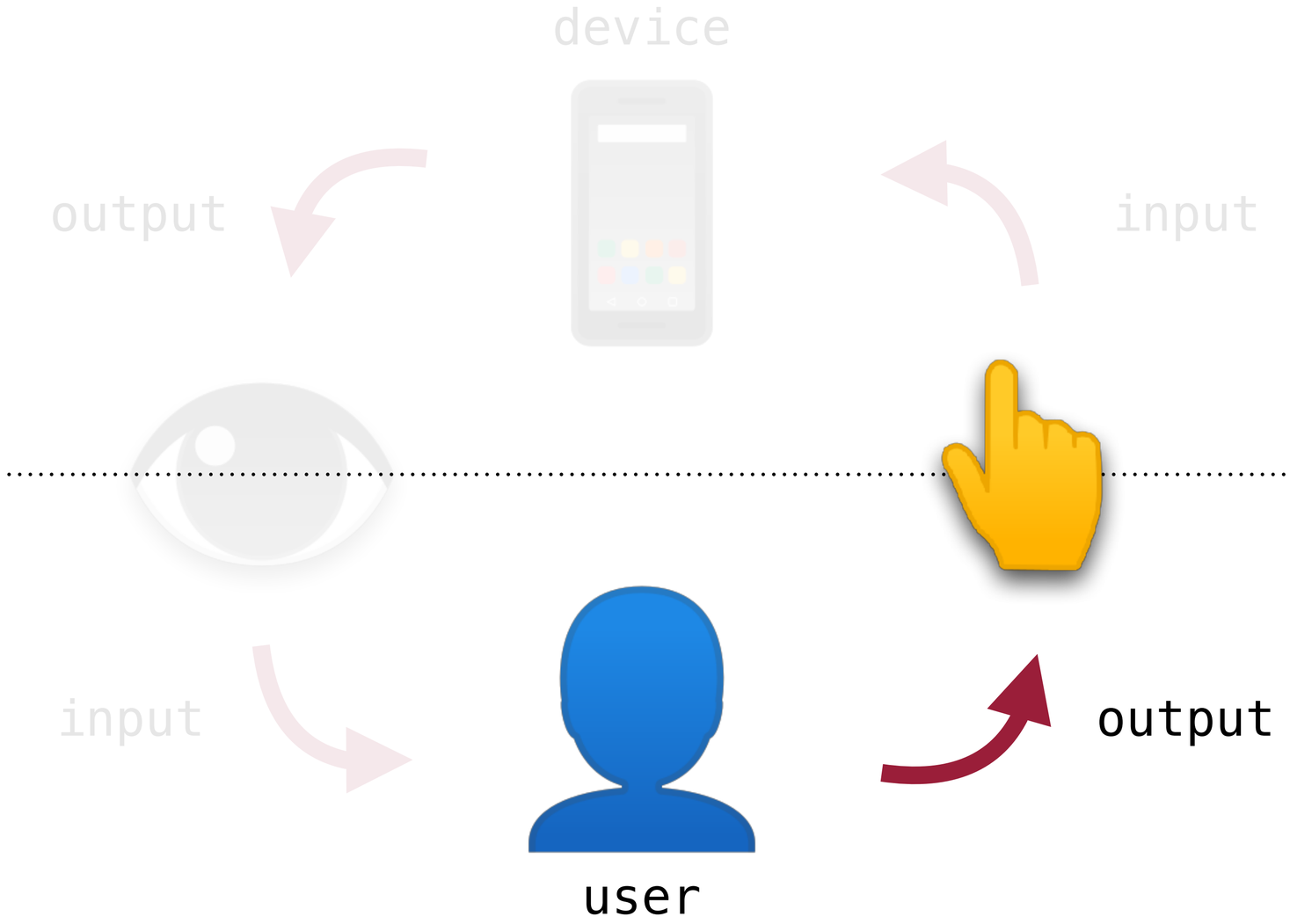
Simple MVI Architecture for Android

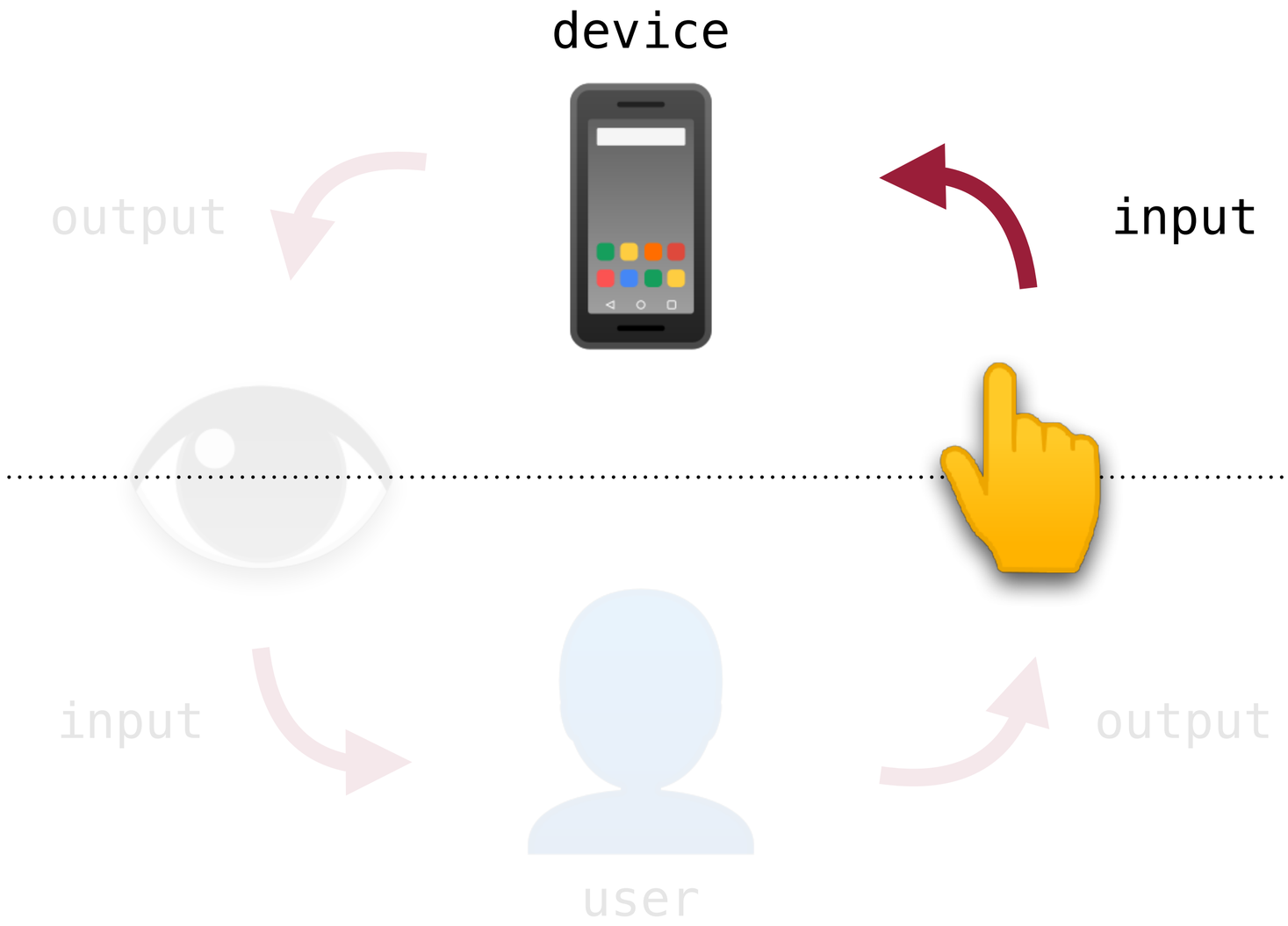
Simple MVI Architecture for Android

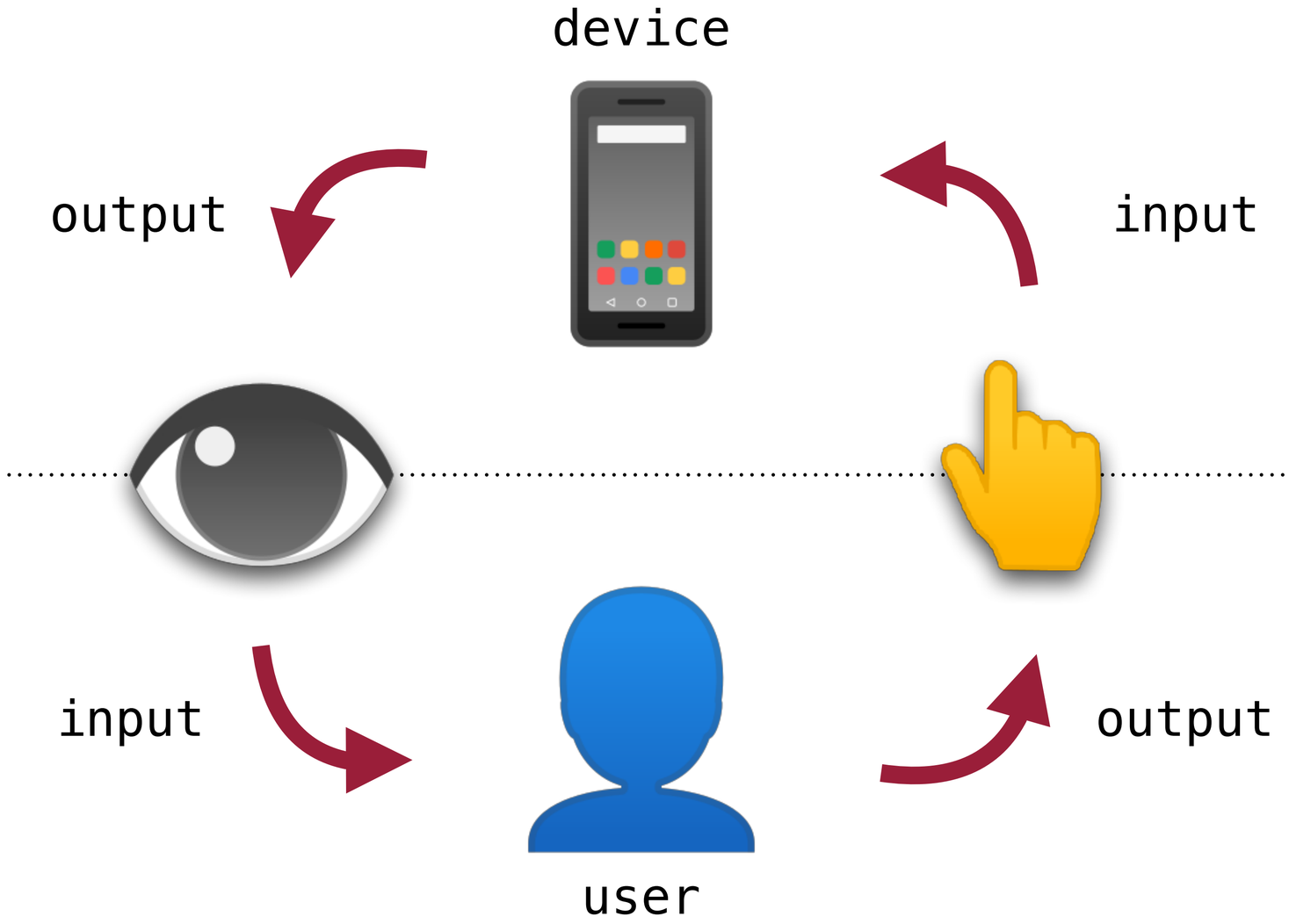

device
output
input
input
output
user

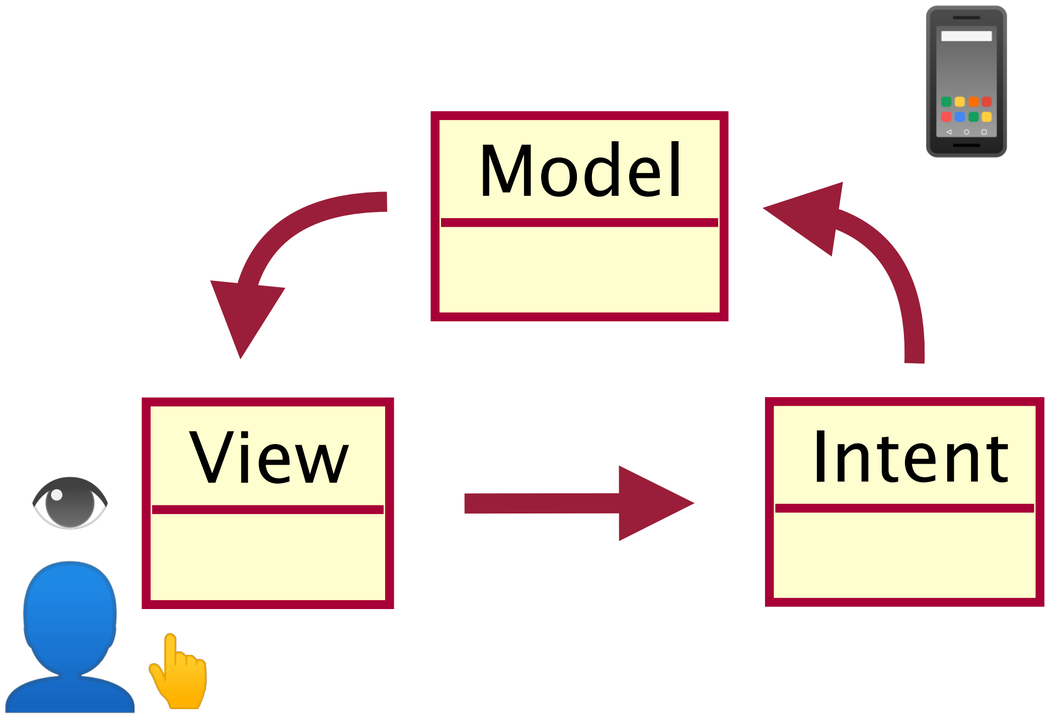

device
output
input
input
output
user

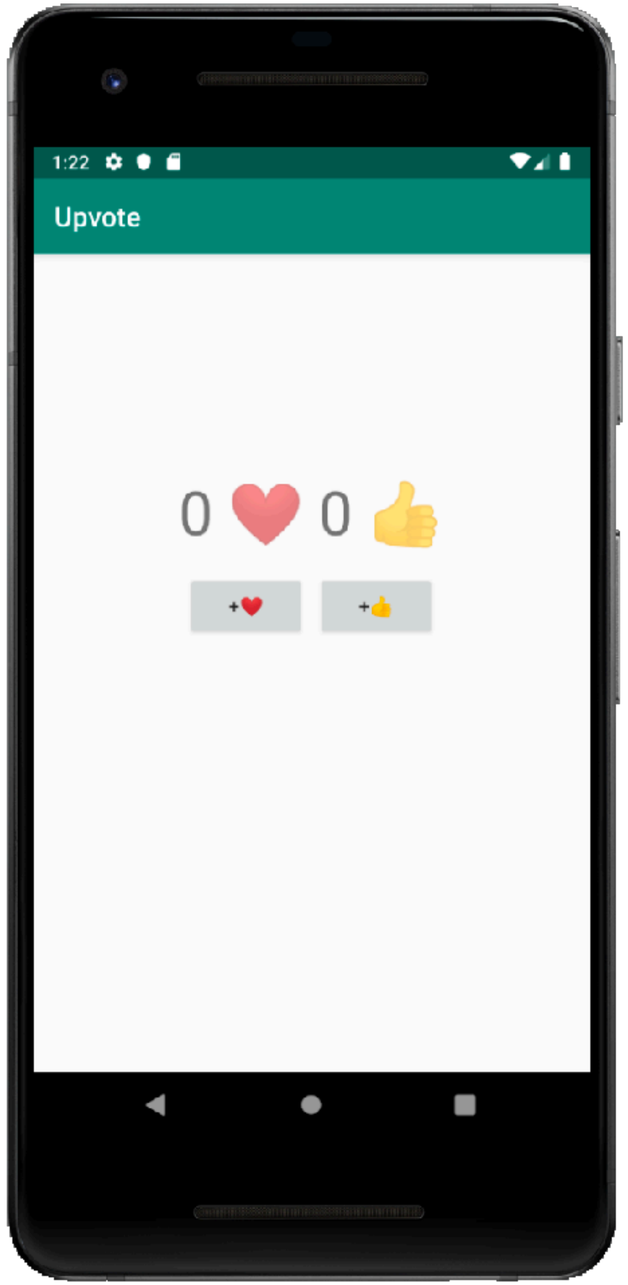

device
output
input
input
output
user

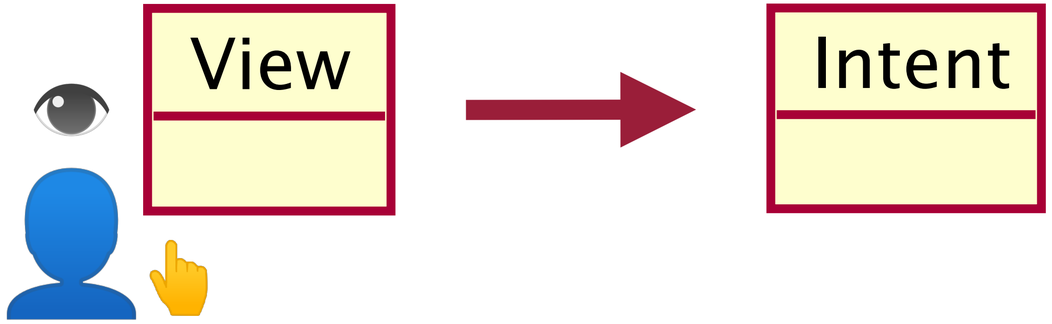

device
output
input
input
output
user

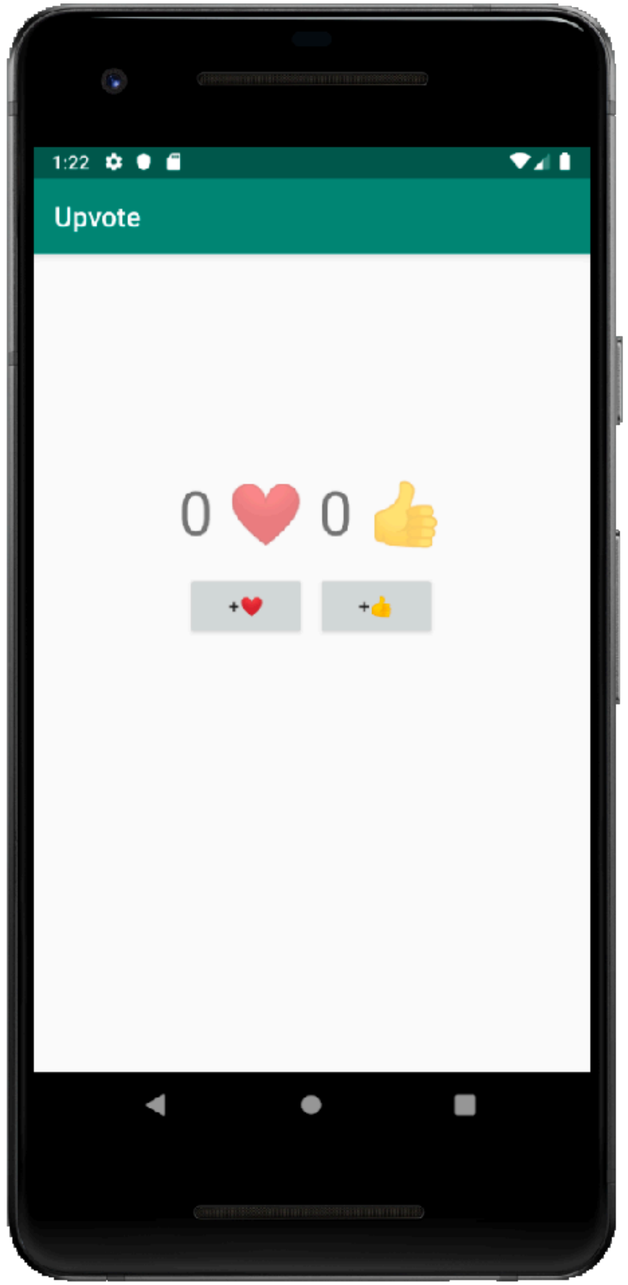

device
output
input
input
output
user

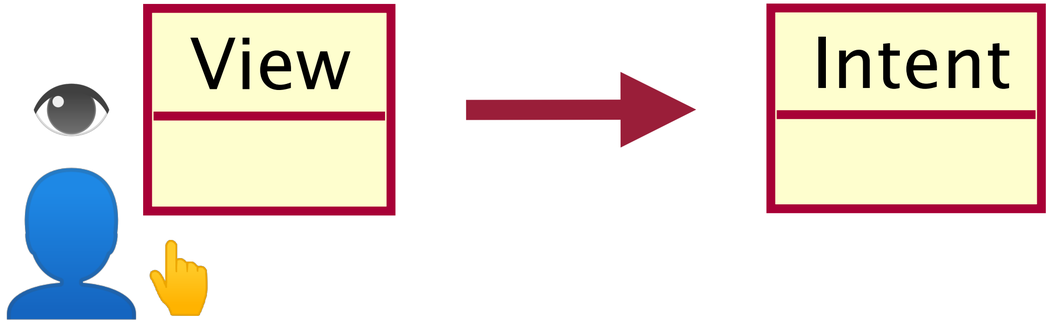

device
output
input
input
output
user

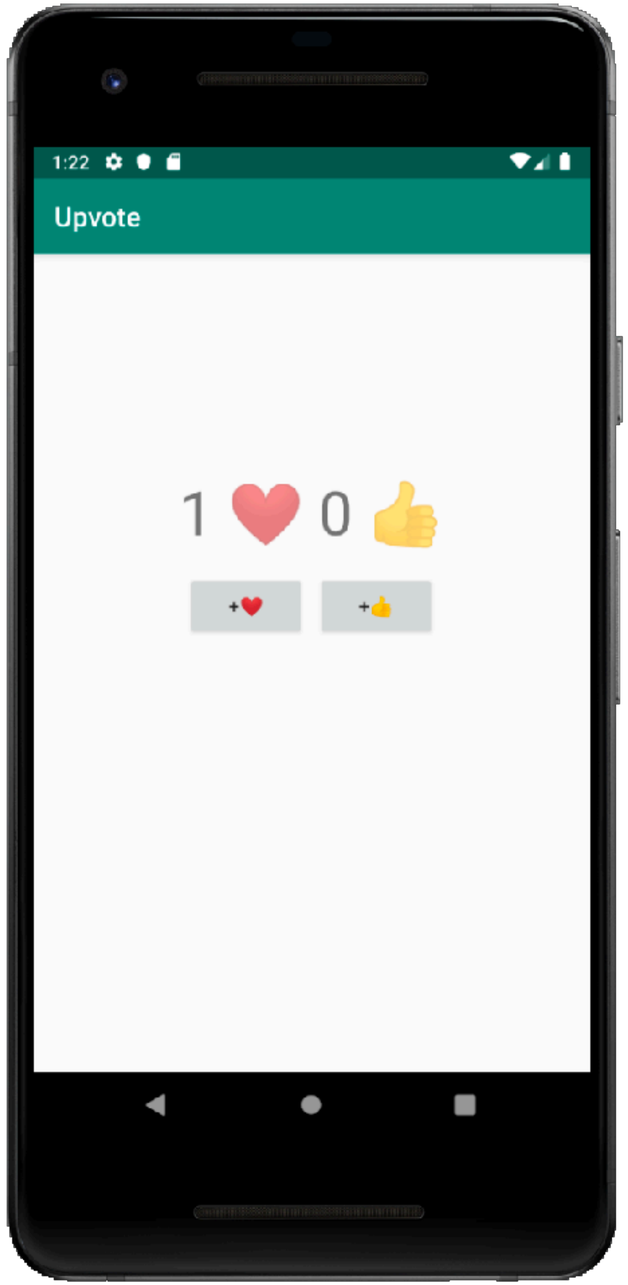

Model
View
Intent

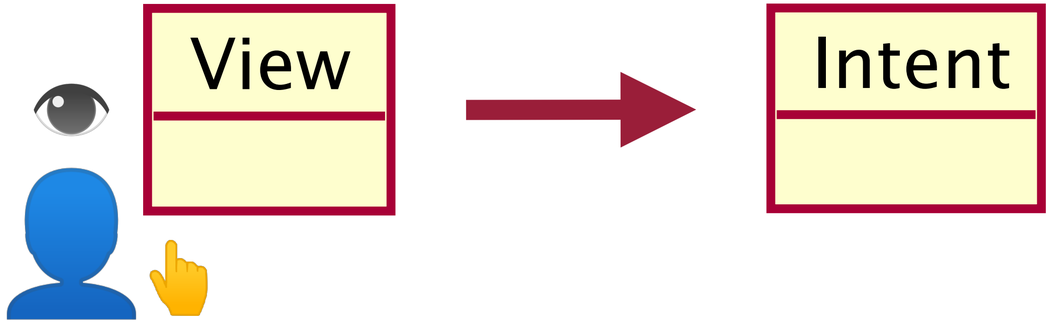

1:22
Upvote
0
0
+
+

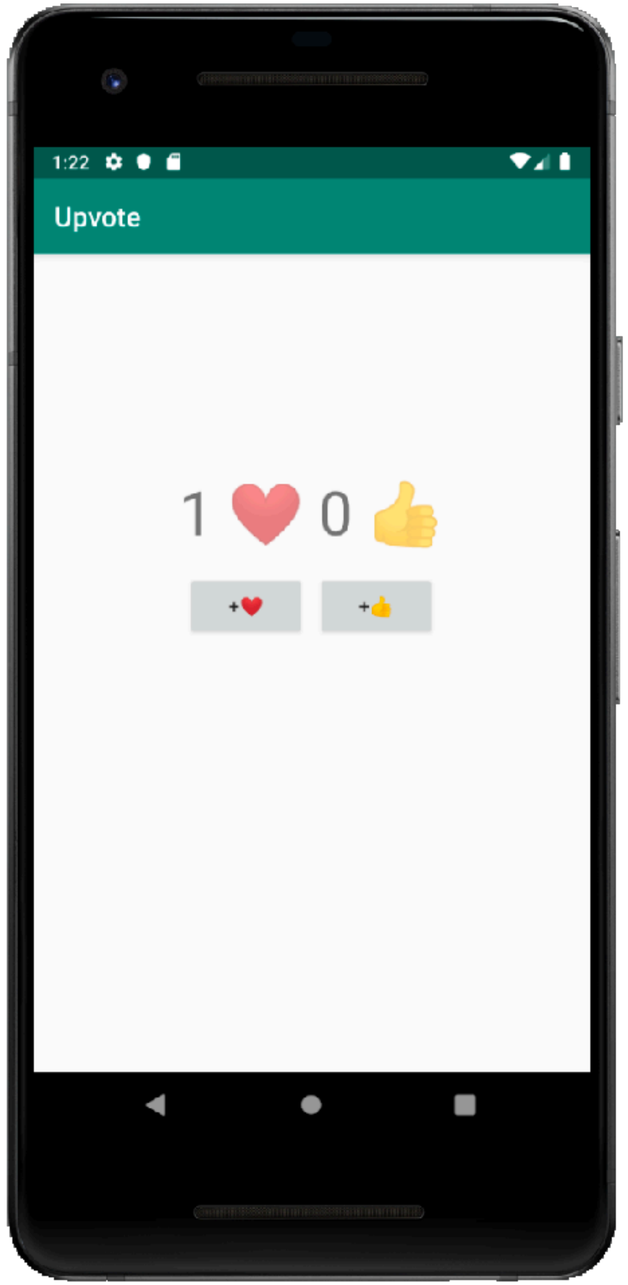

View
Intent

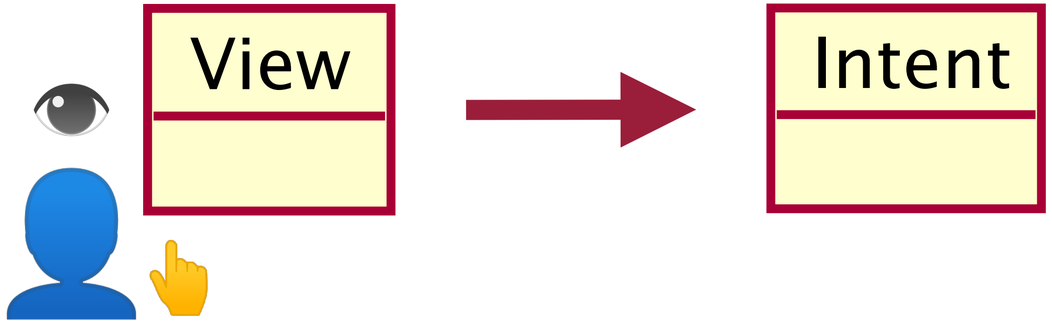

1:22
Upvote
0
0
+
+

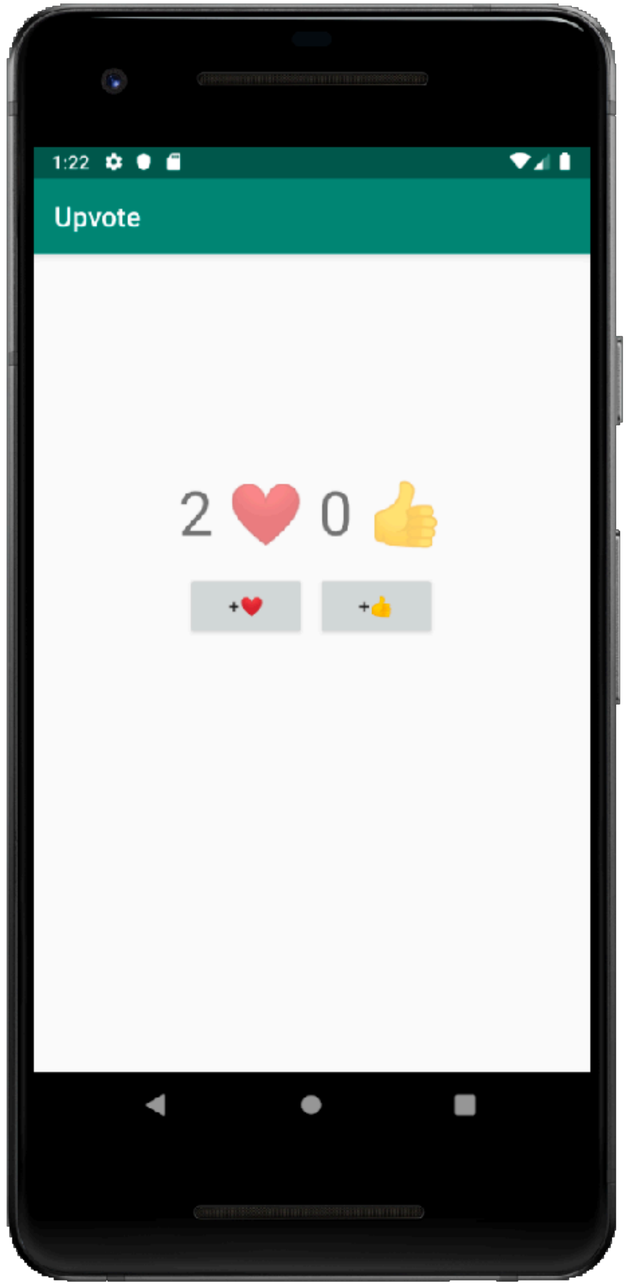

View
Intent

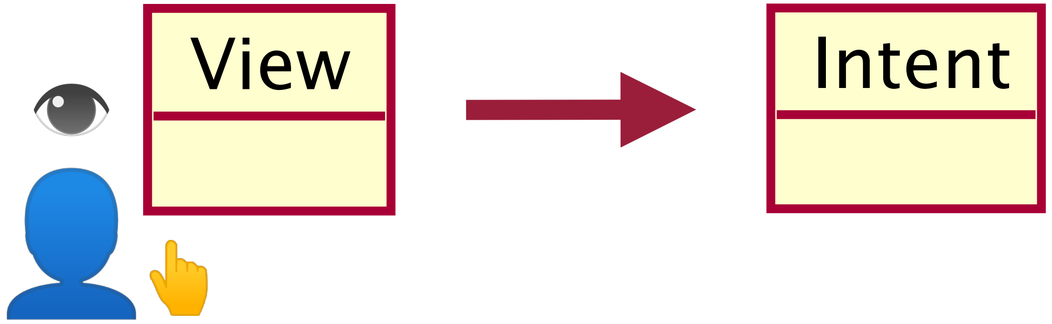

1:22
Upvote
1
0
+
+

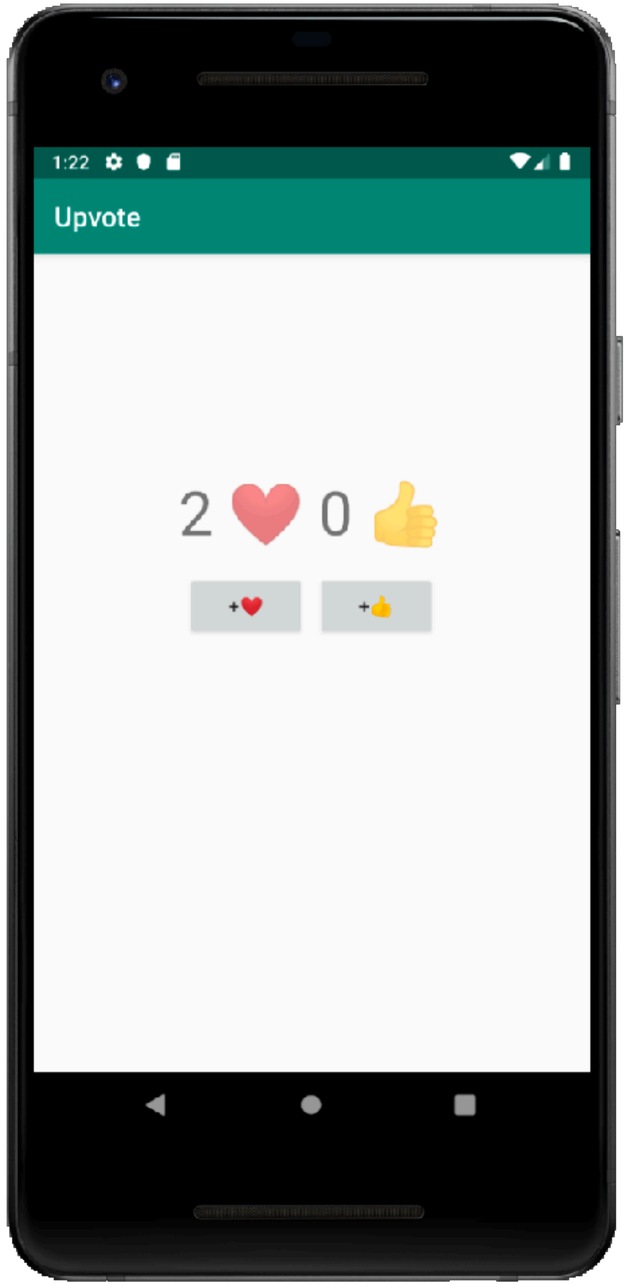

View
Intent

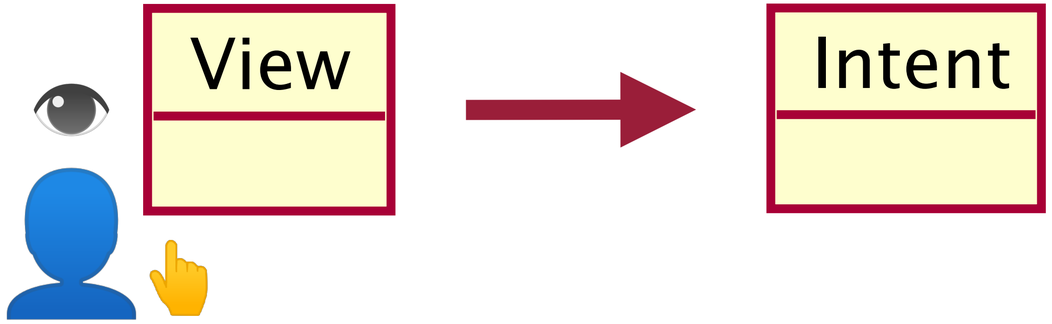

1:22
Upvote
1
0
+
+

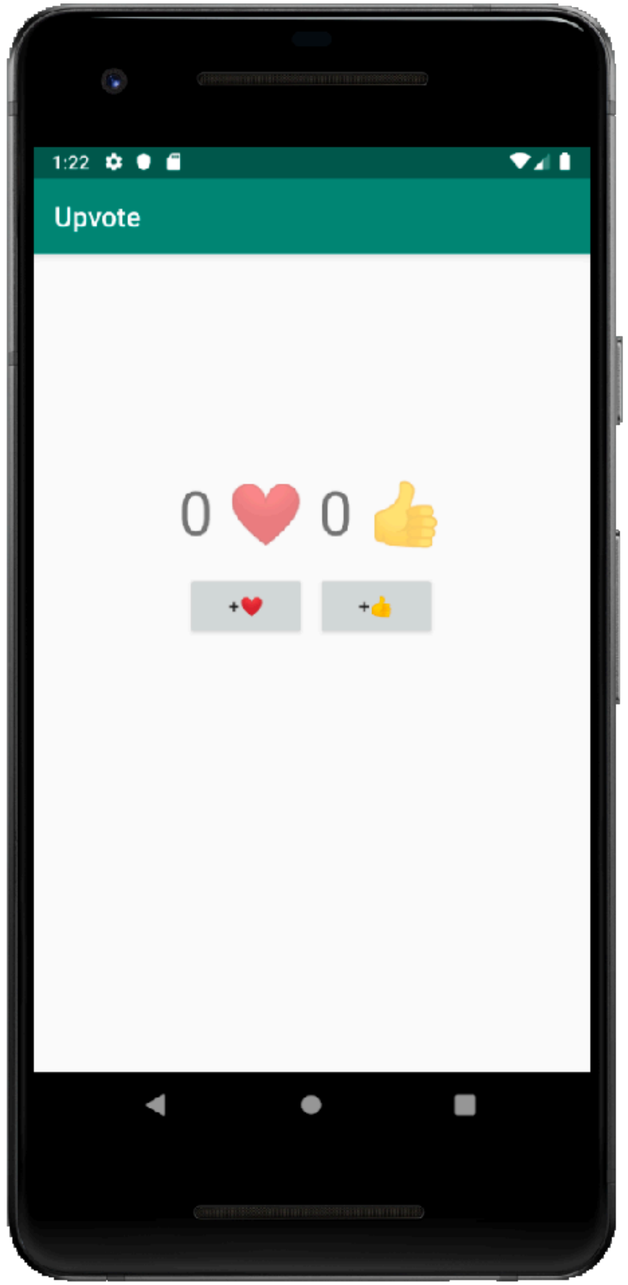

View
Intent


1:22
Upvote
2
0
+
+

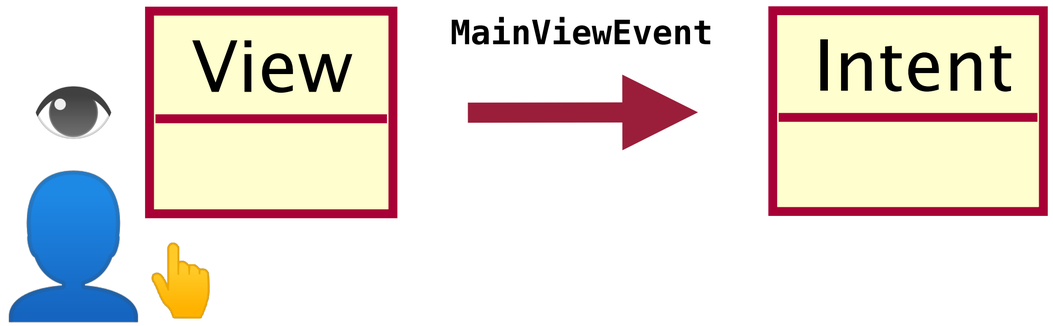

View
Intent

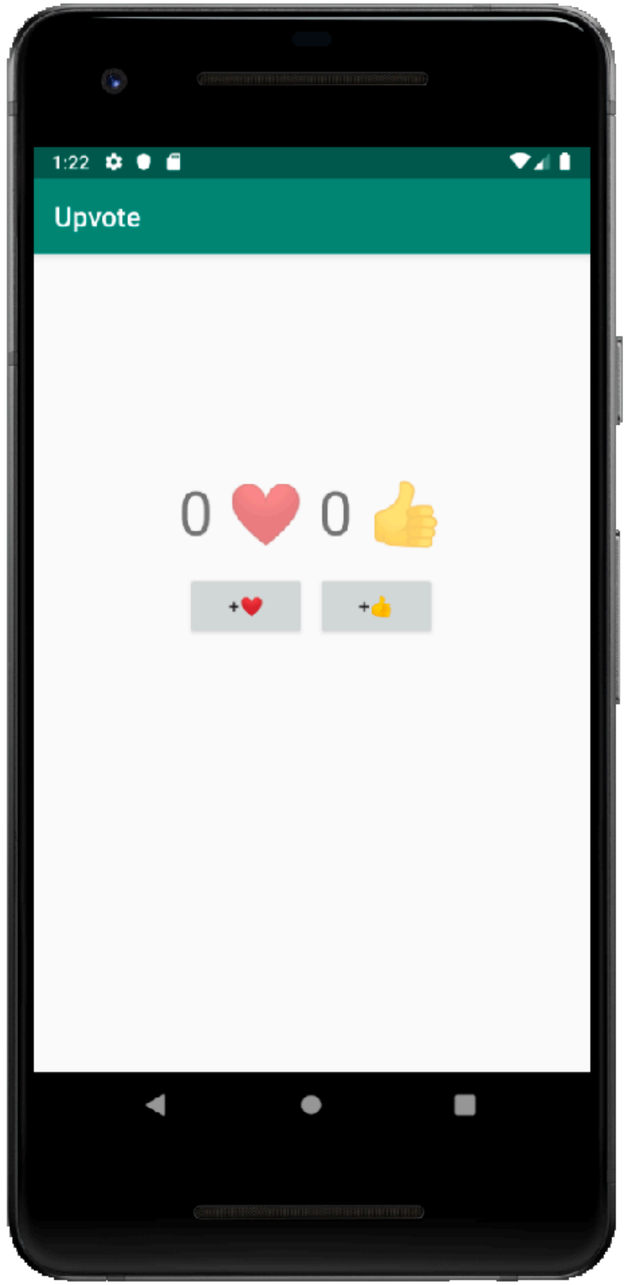

1:22
Upvote
2
0
+
+


View
Intent

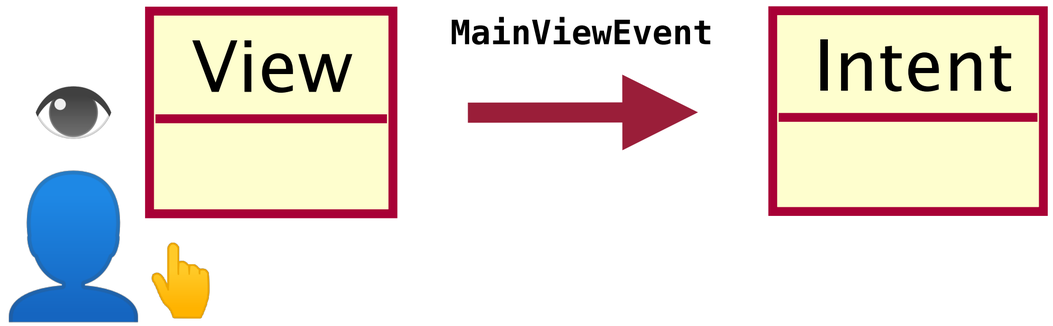


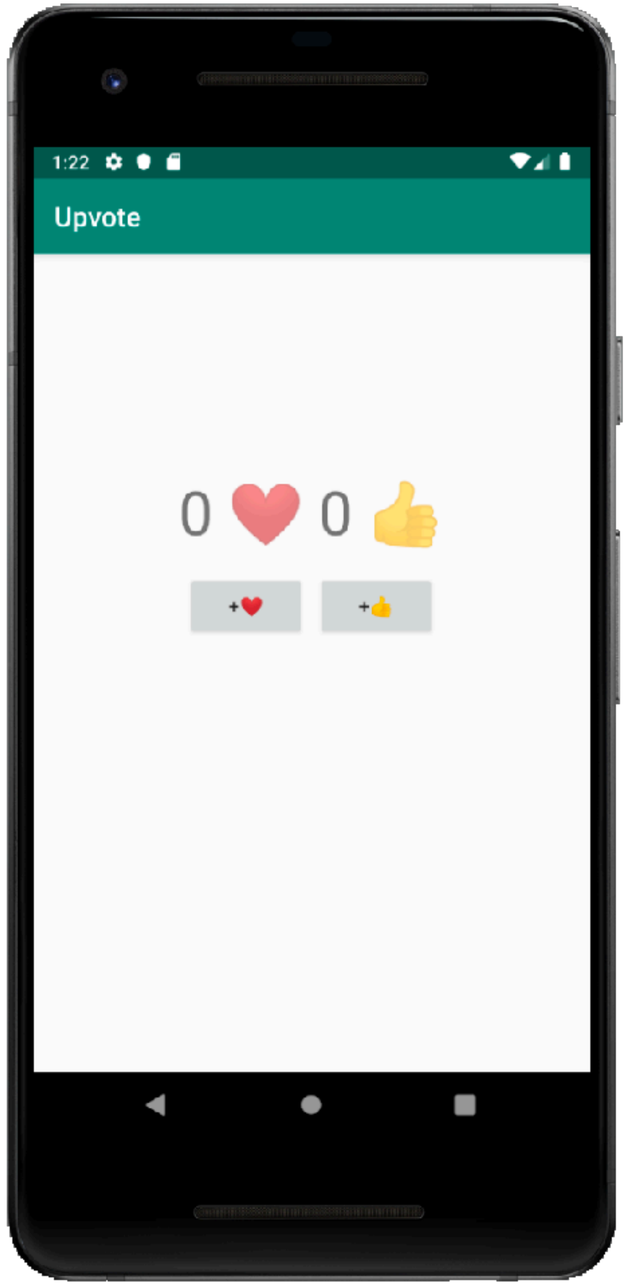

```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```





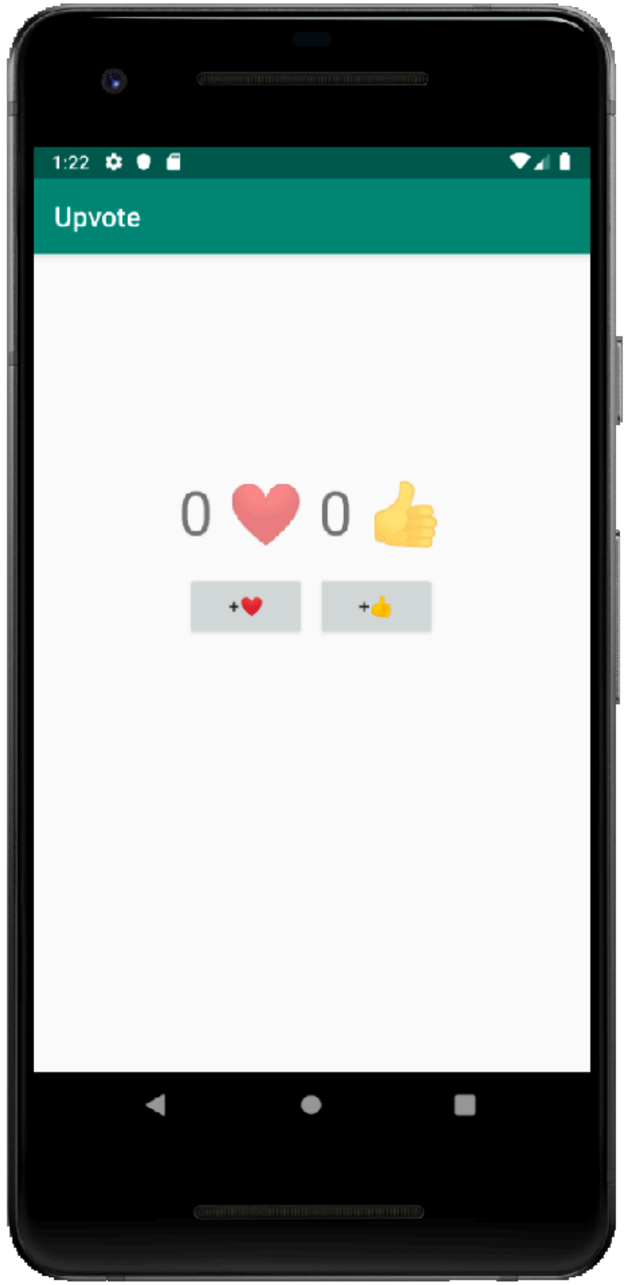

```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```





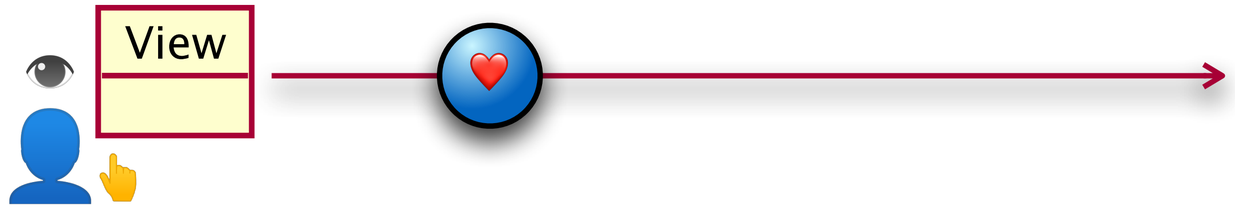

```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```

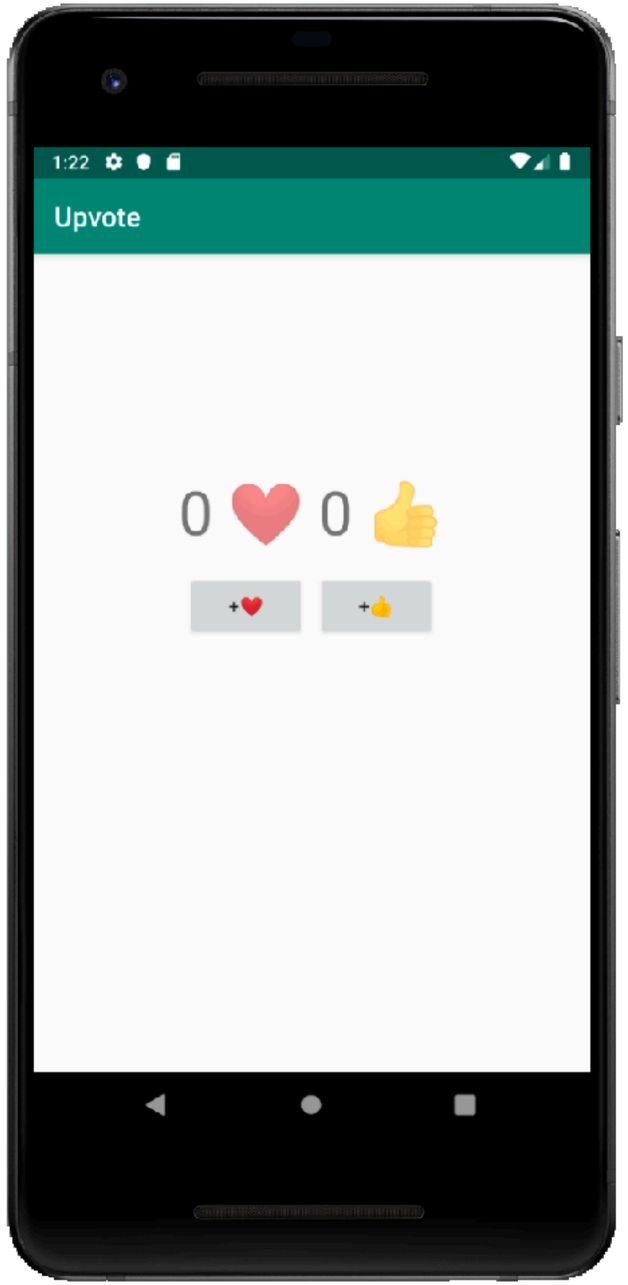
- MainViewEvent
  - +👍 ThumbsUpClick
  - +❤️ LoveItClick




```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```

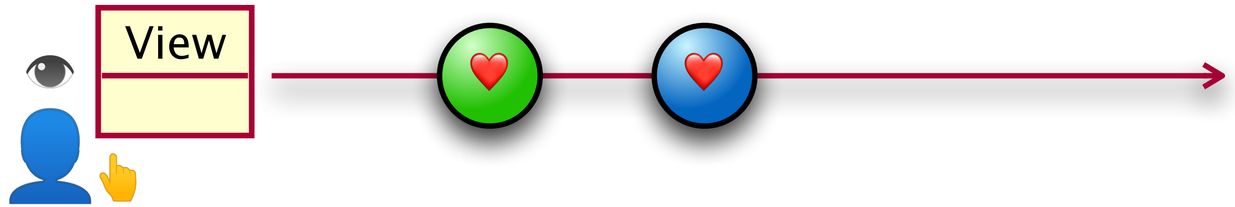
- MainViewEvent
  - +👍 ThumbsUpClick
  - +❤️ LoveItClick

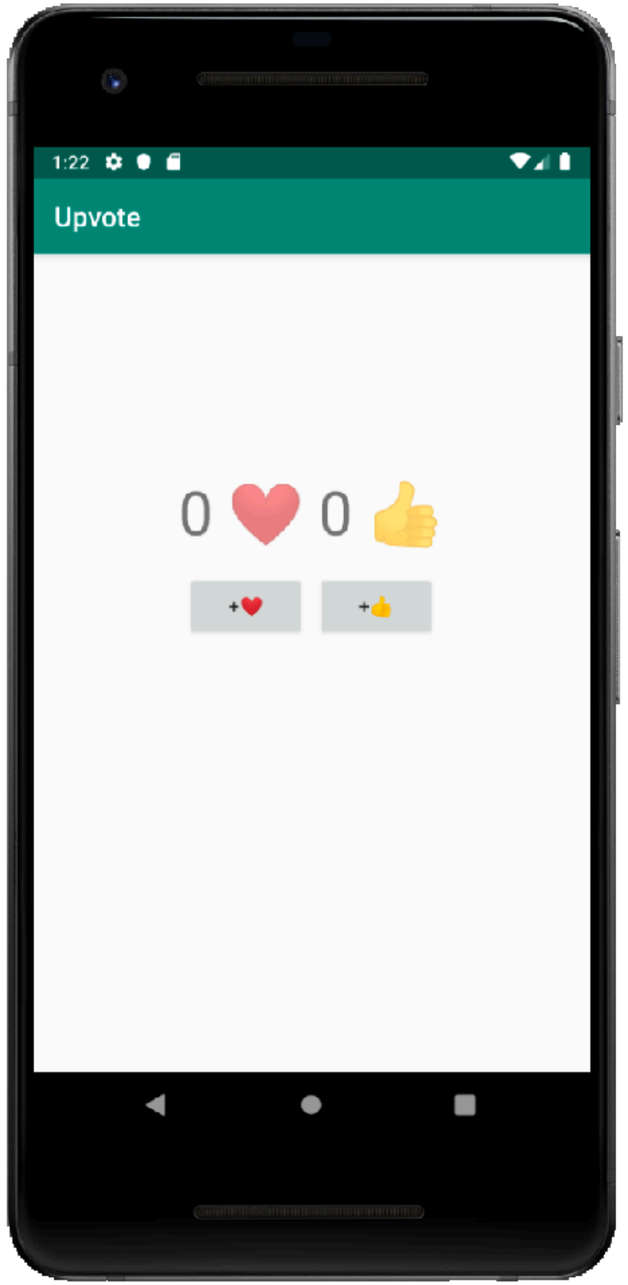





```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```

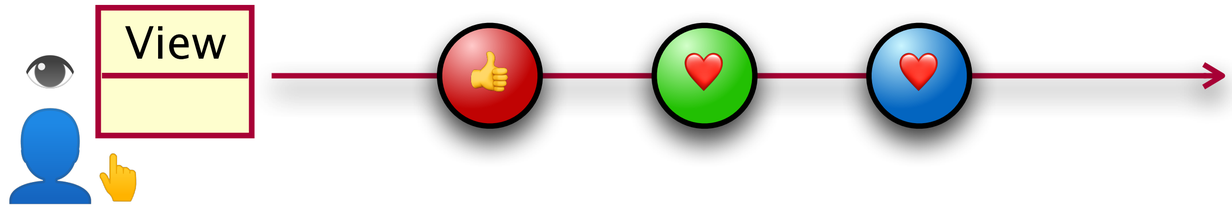
- MainViewEvent
  - +👍 ThumbsUpClick
  - +❤️ LoveItClick

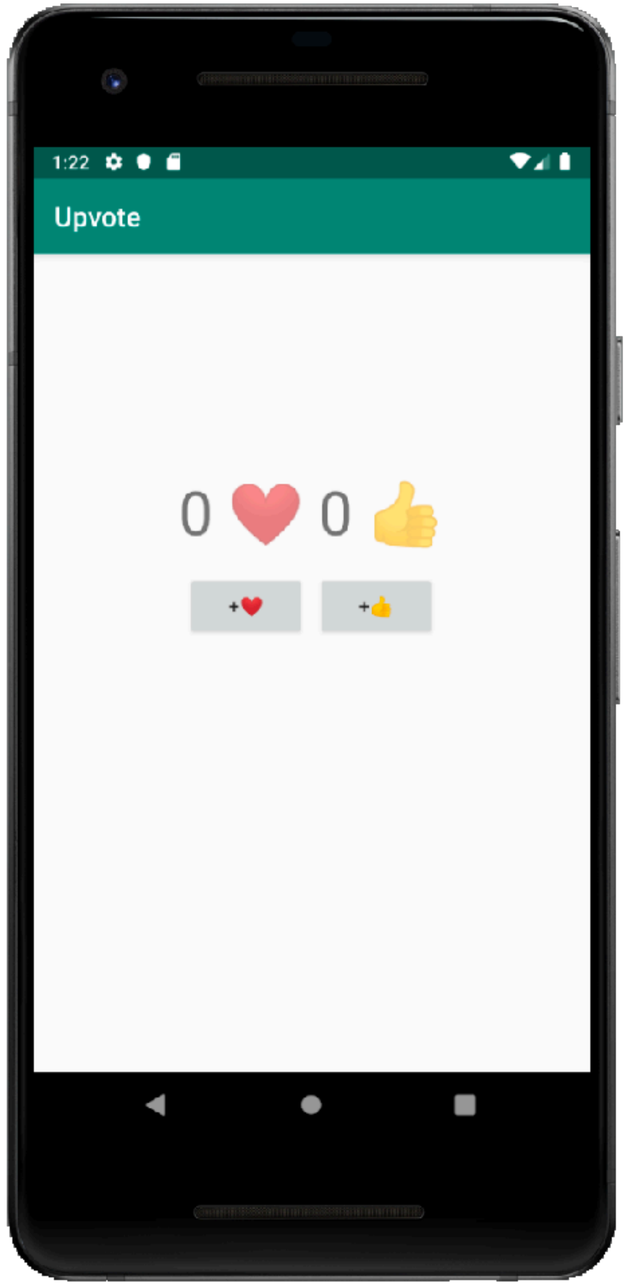


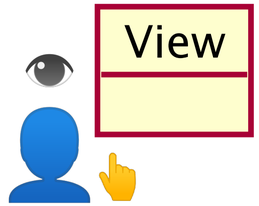



```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```

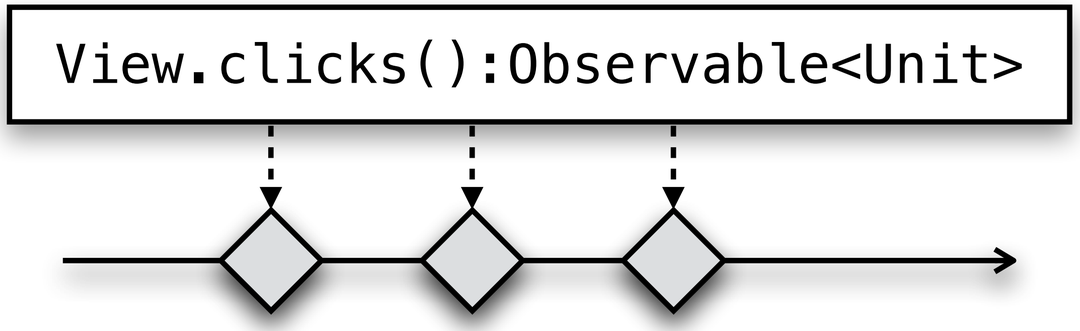
- MainViewEvent
  - +👍 ThumbsUpClick
  - +❤️ LoveItClick

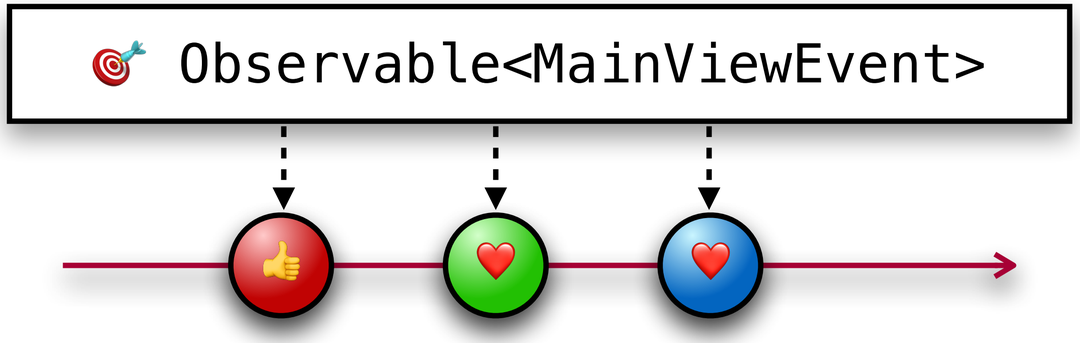


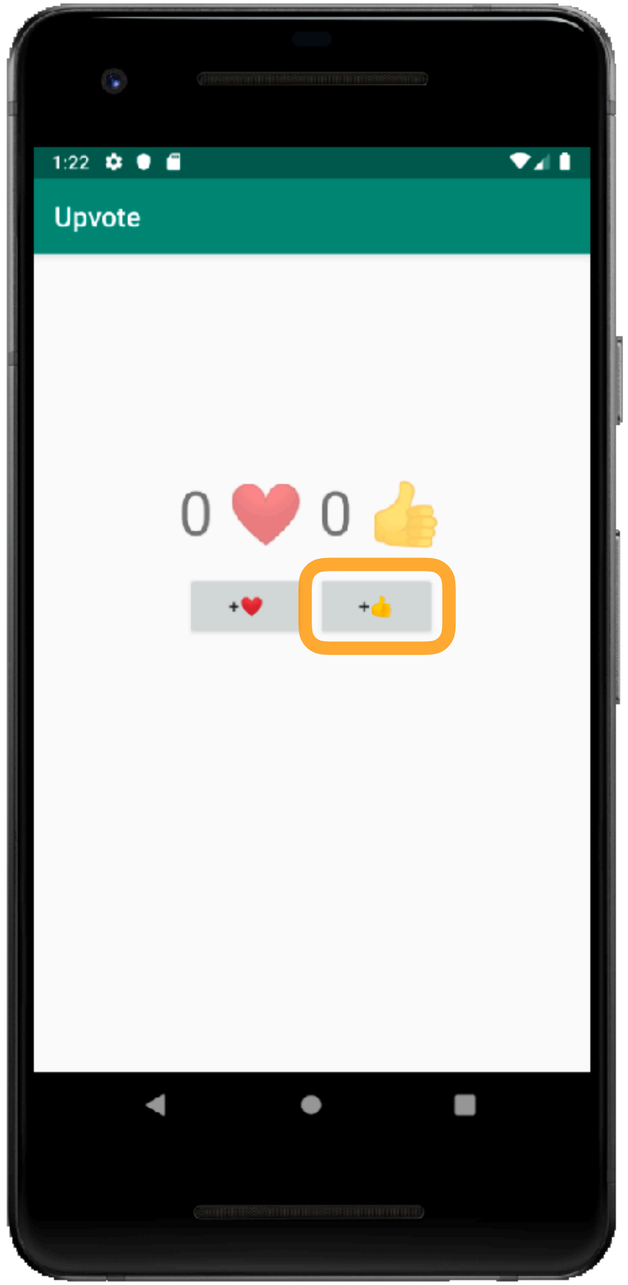


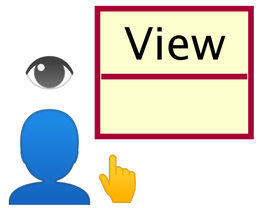



```
package com.jakewharton.rxbinding2.view
```





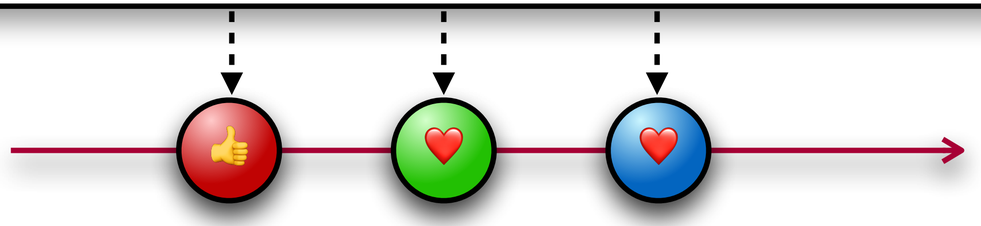


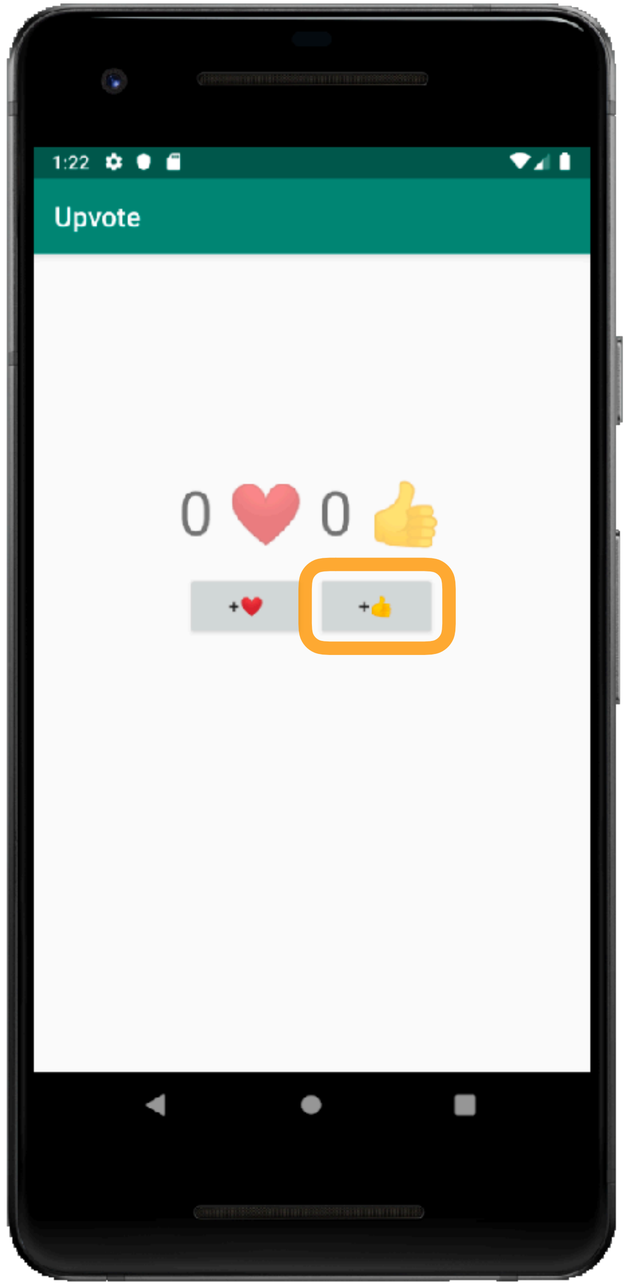


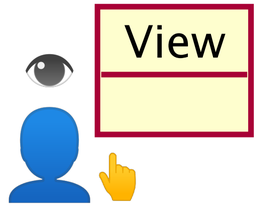

```
package com.jakewharton.rxbinding2.view
```

```
thumbButton.clicks()
```

```
🎯 Observable<MainViewEvent>
```





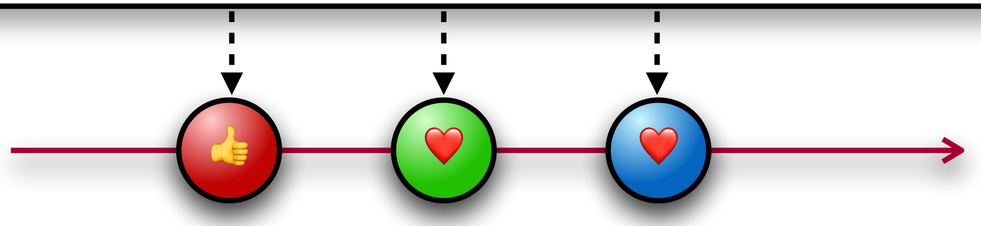


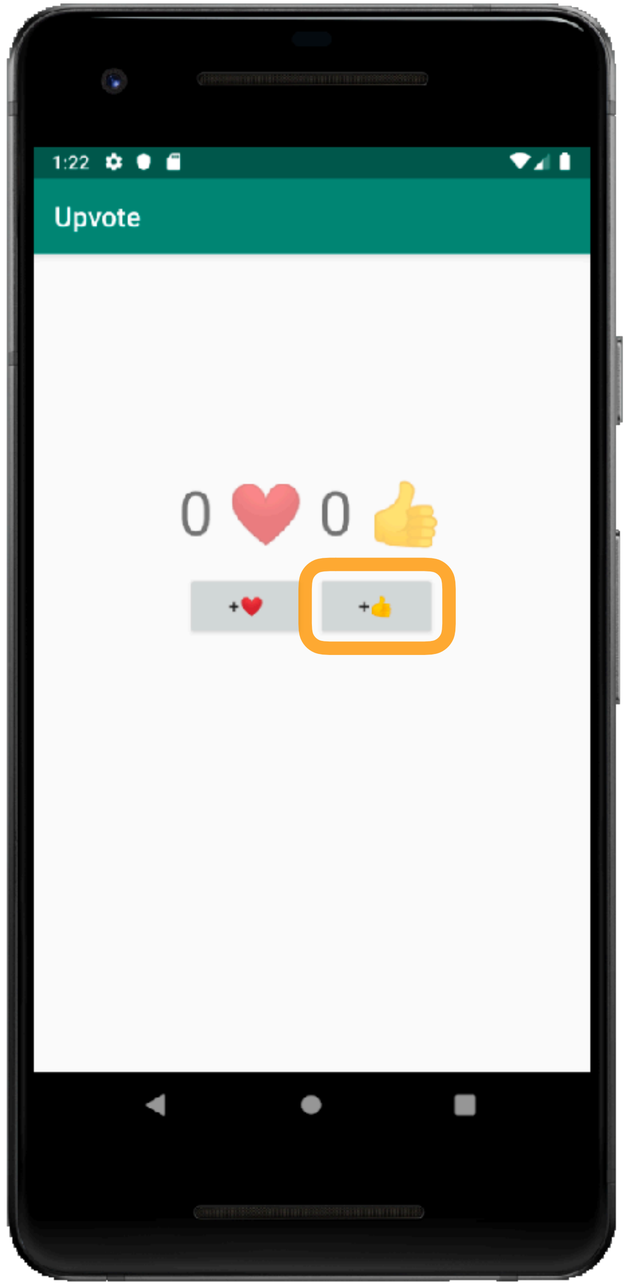

```
package com.jakewharton.rxbinding2.view
```

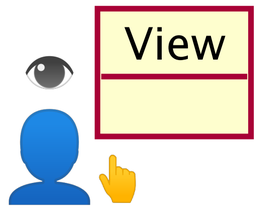
```
thumbButton.clicks()
```

```
map { MainViewEvent.ThumbsUpClick }
```

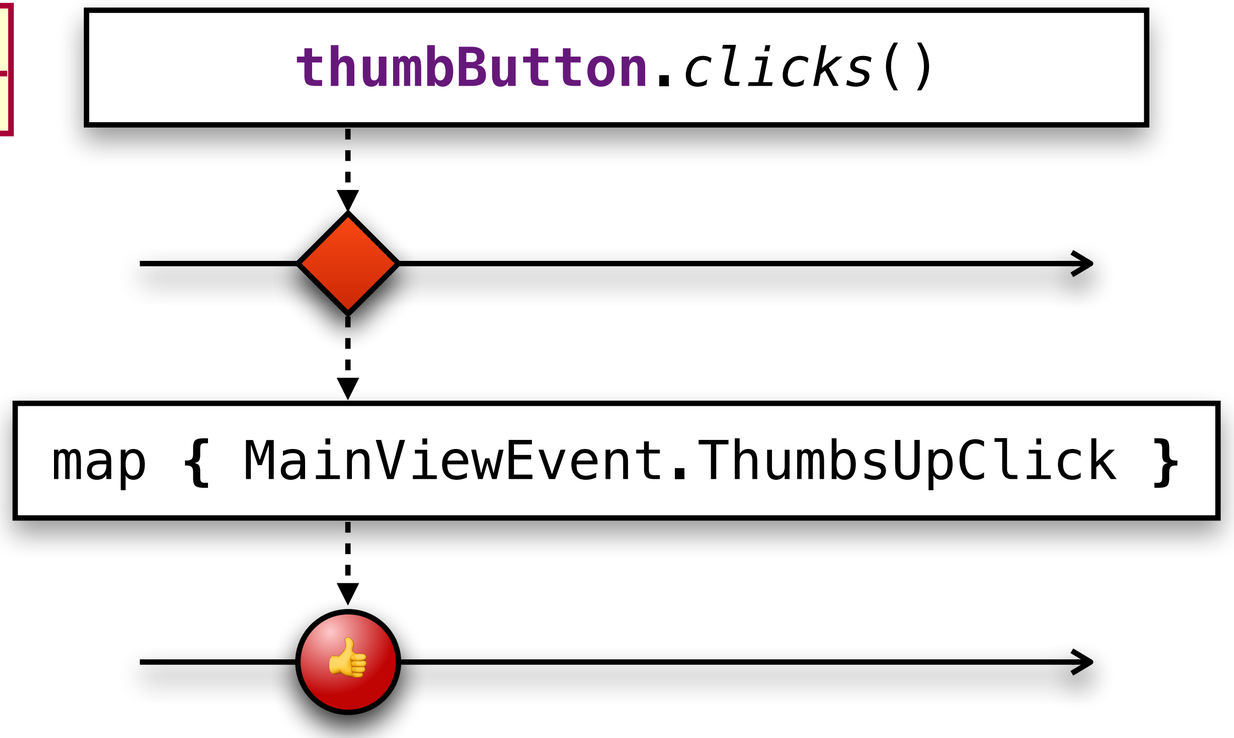
```
🎯 Observable<MainViewEvent>
```

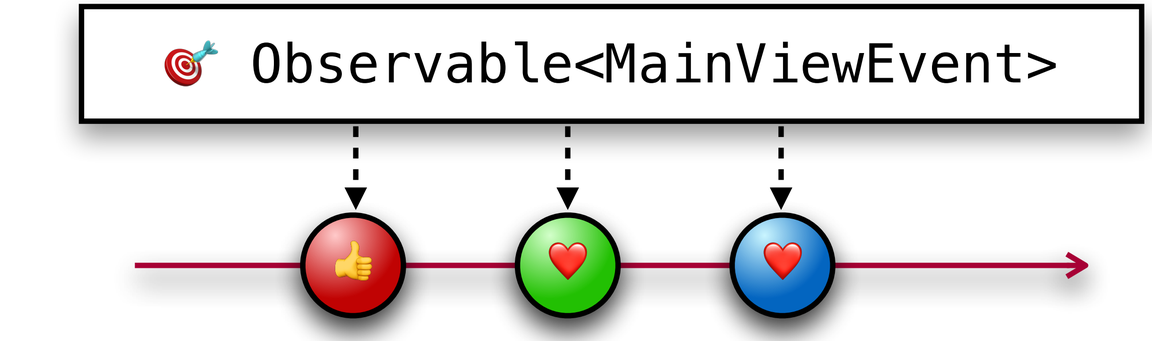


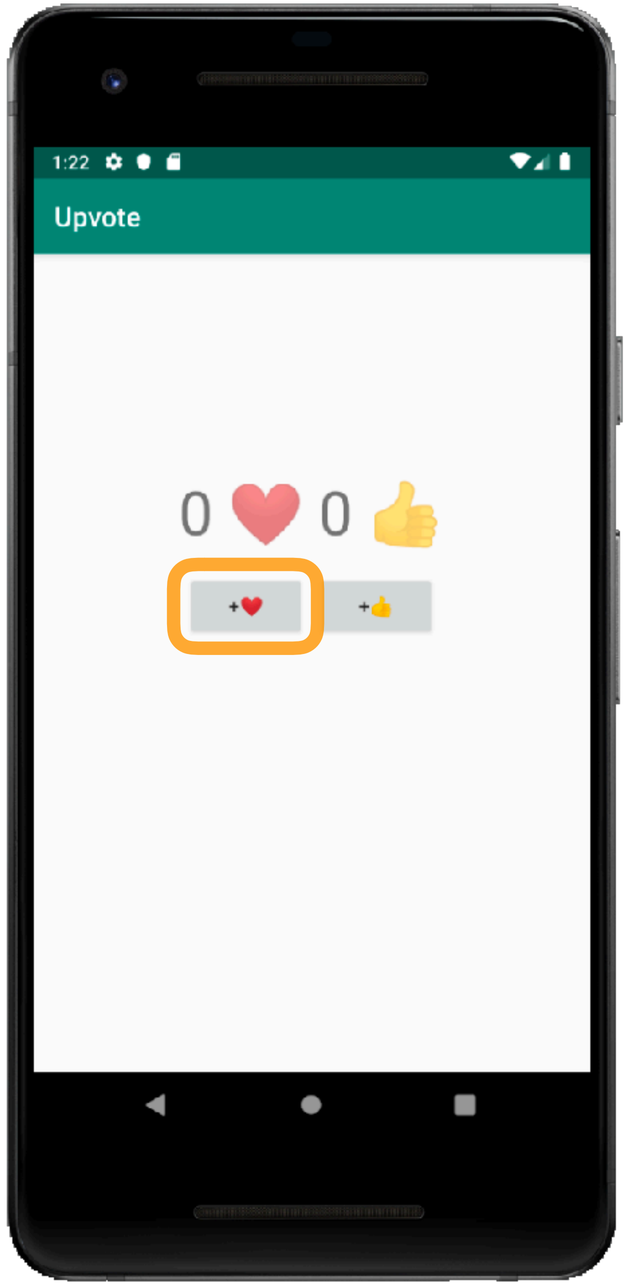


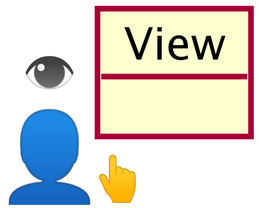



```
package com.jakewharton.rxbinding2.view
```





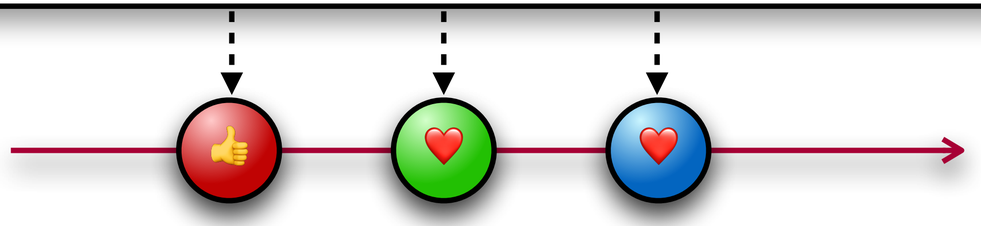


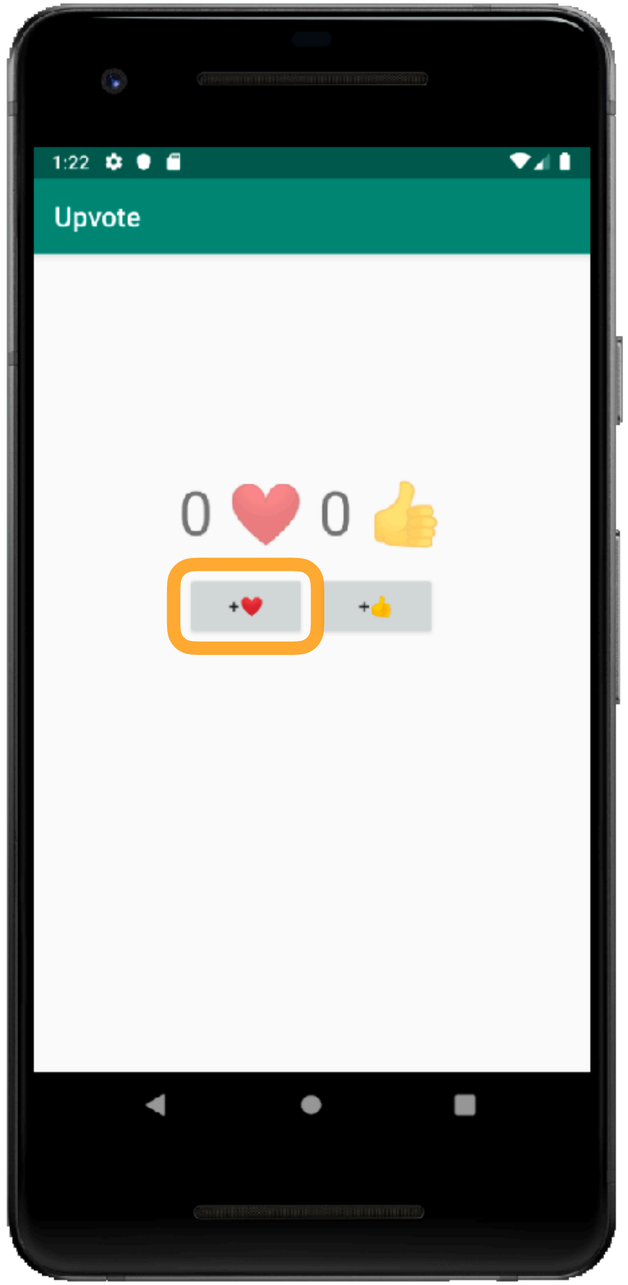


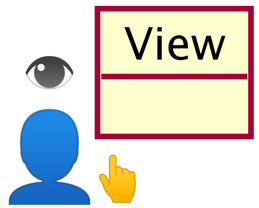

```
package com.jakewharton.rxbinding2.view
```

```
heartButton.clicks()
```

```
🎯 Observable<MainViewEvent>
```





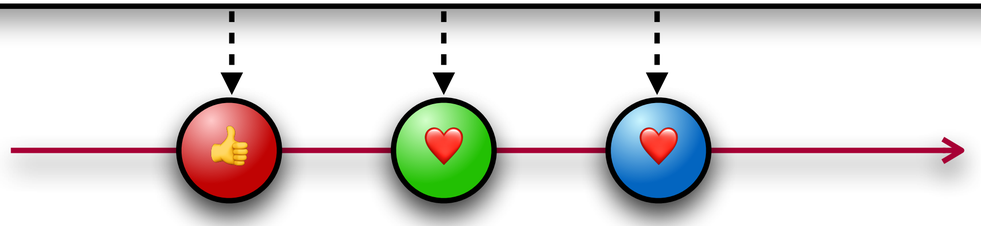


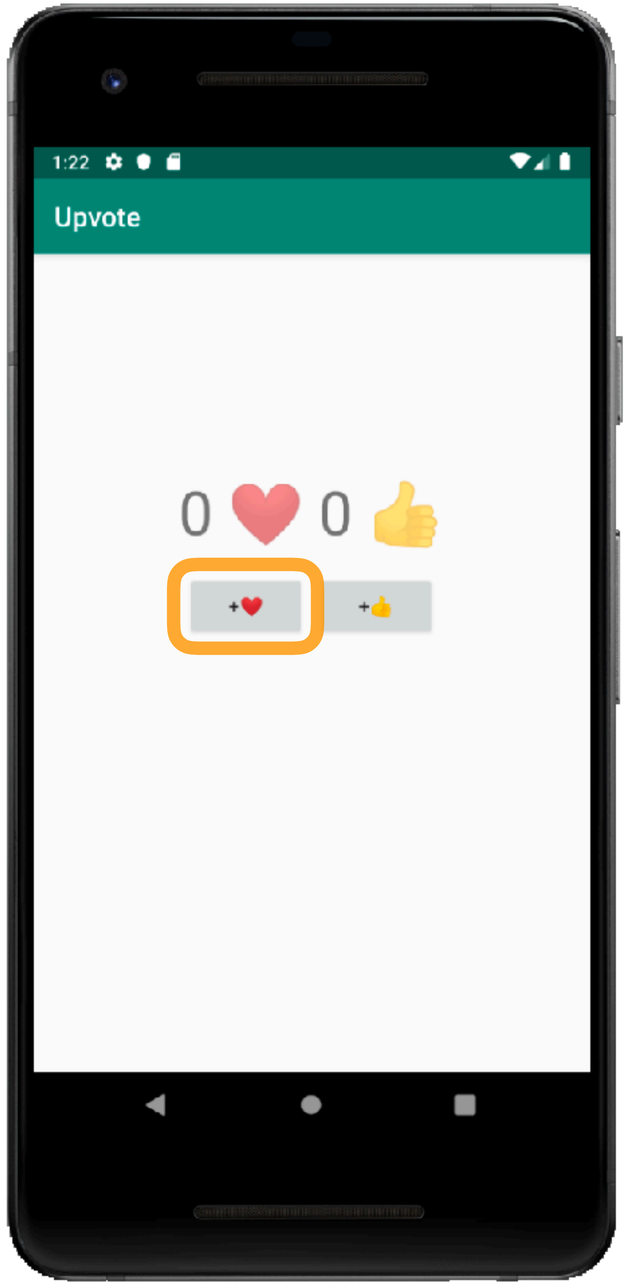

```
package com.jakewharton.rxbinding2.view
```

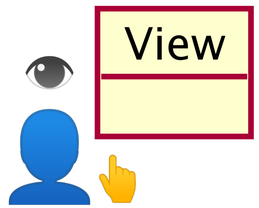
```
heartButton.clicks()
```

```
map { MainViewEvent.LoveItClick }
```

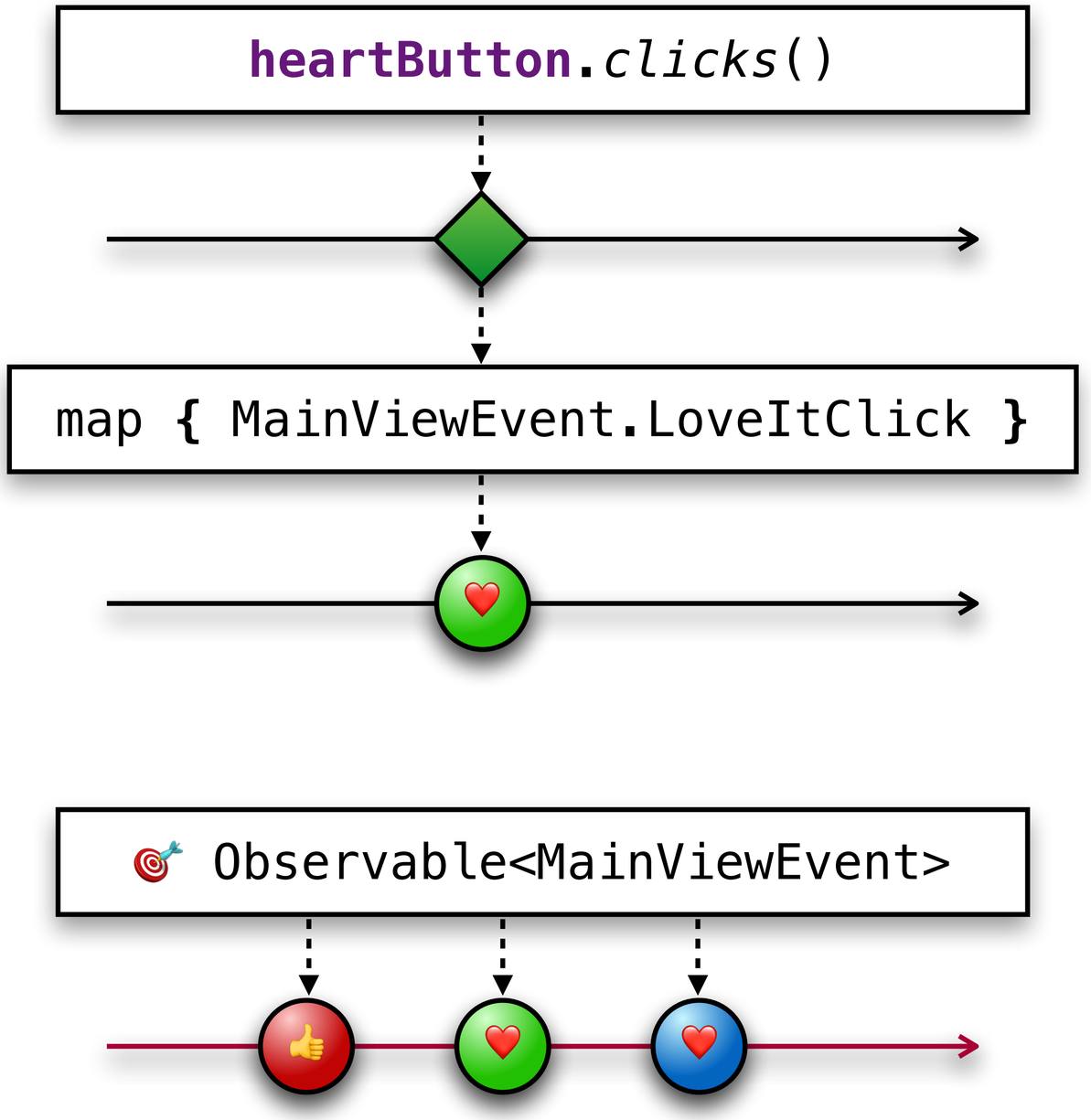
```
🎯 Observable<MainViewEvent>
```

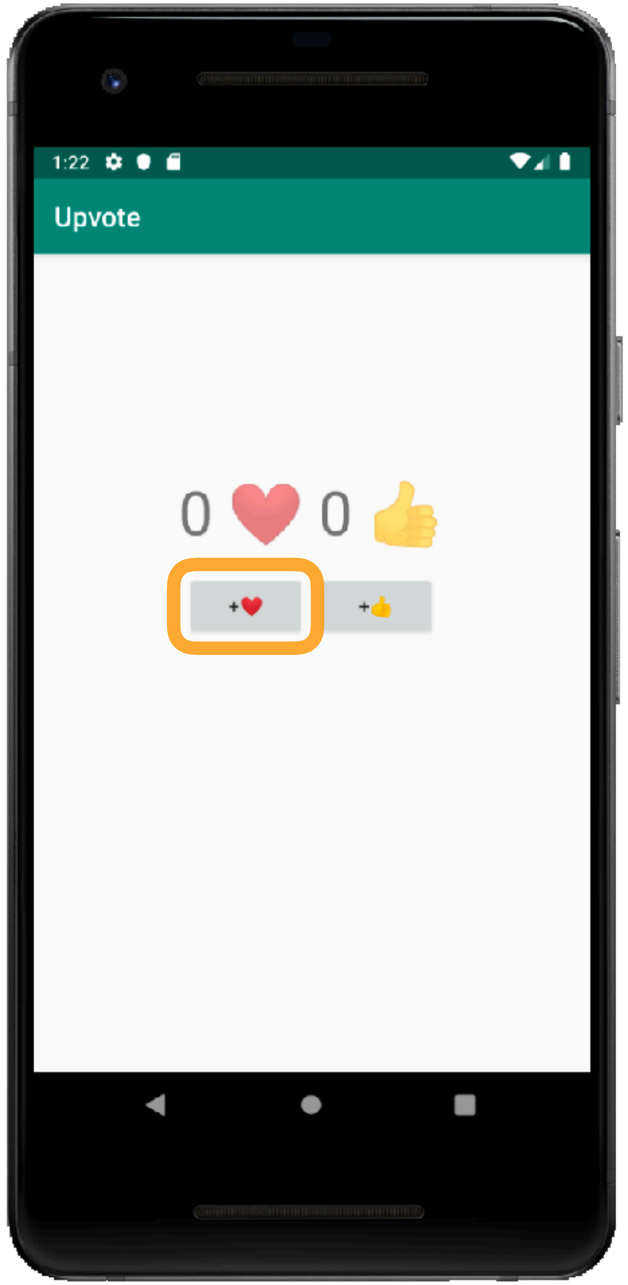


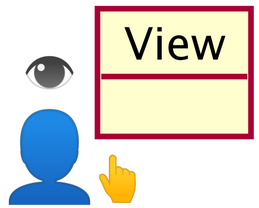




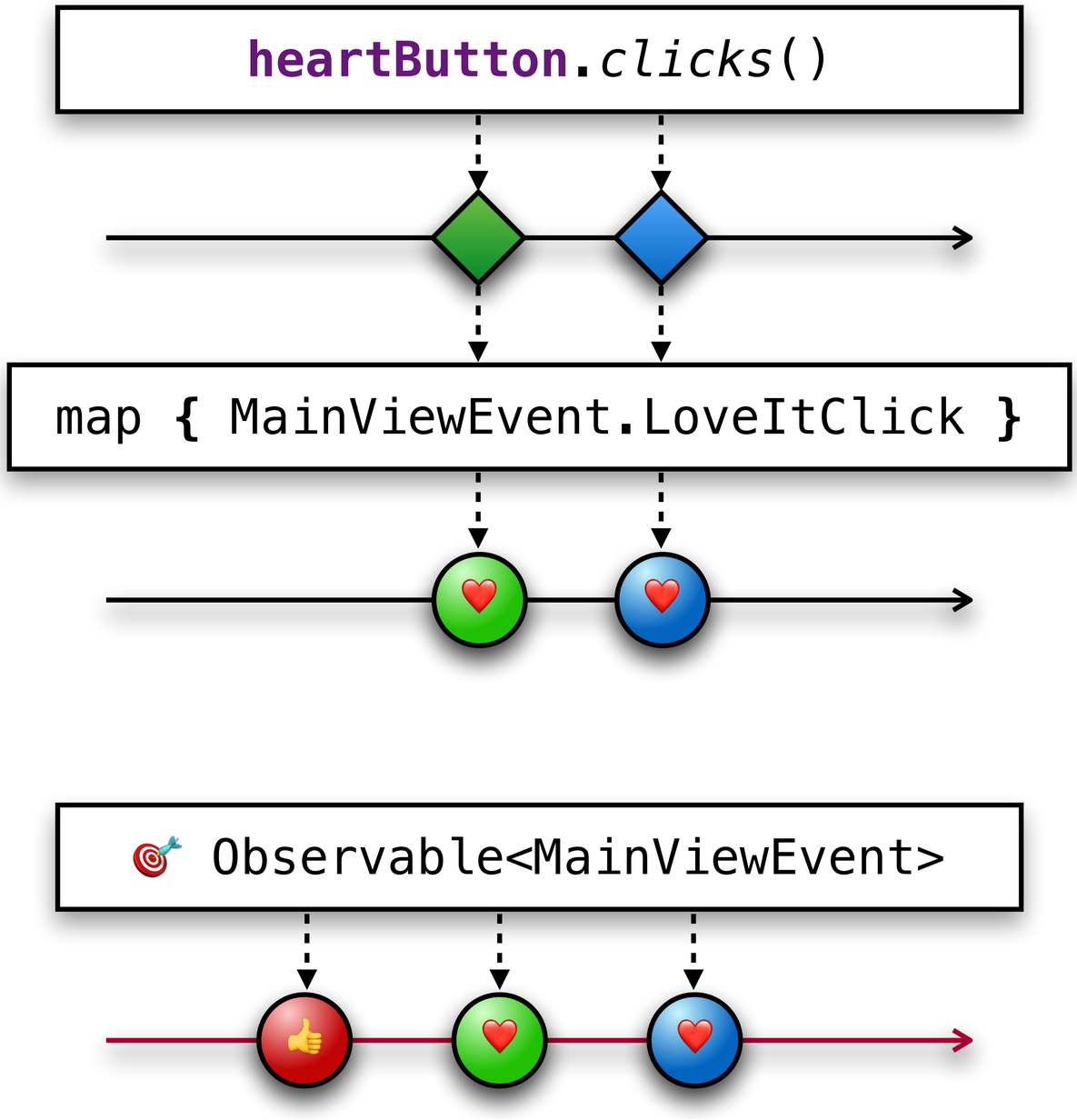

```
package com.jakewharton.rxbinding2.view
```

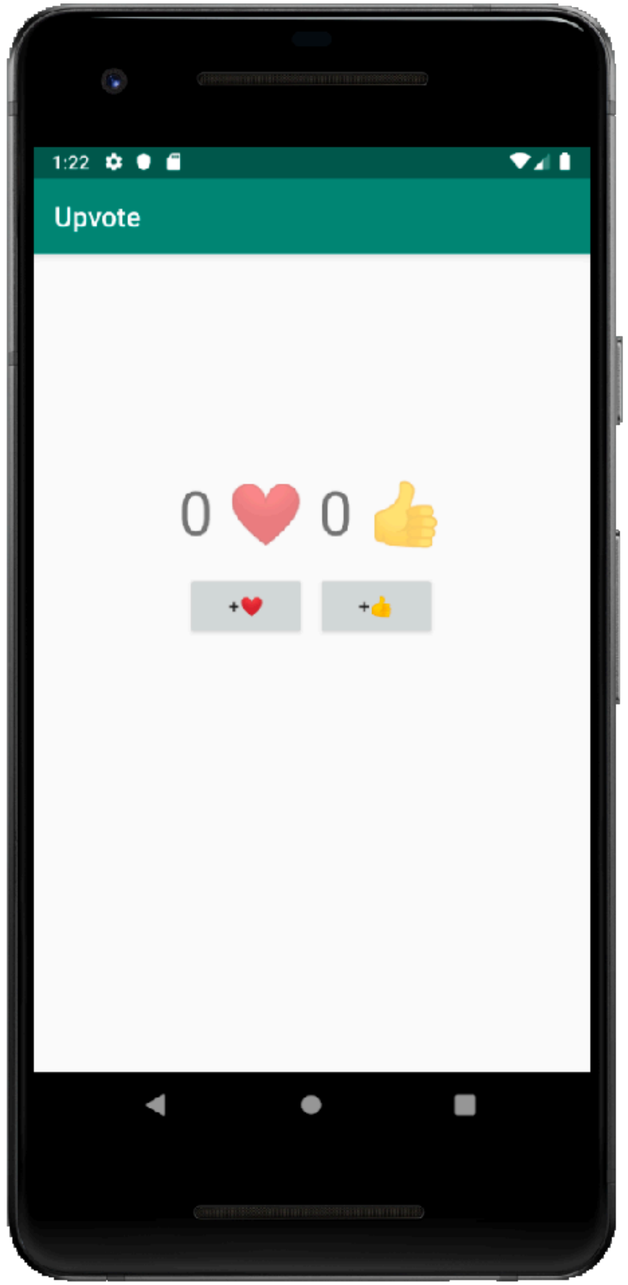


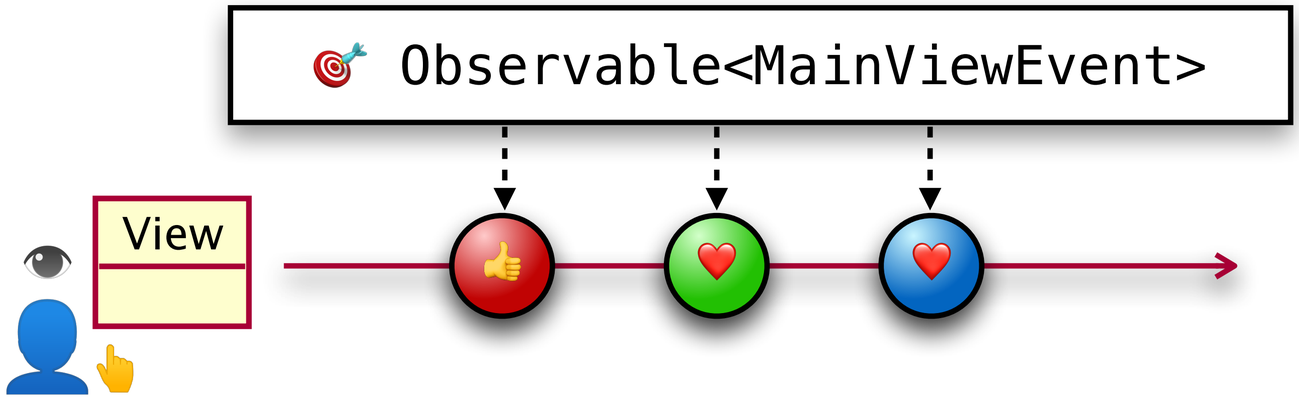


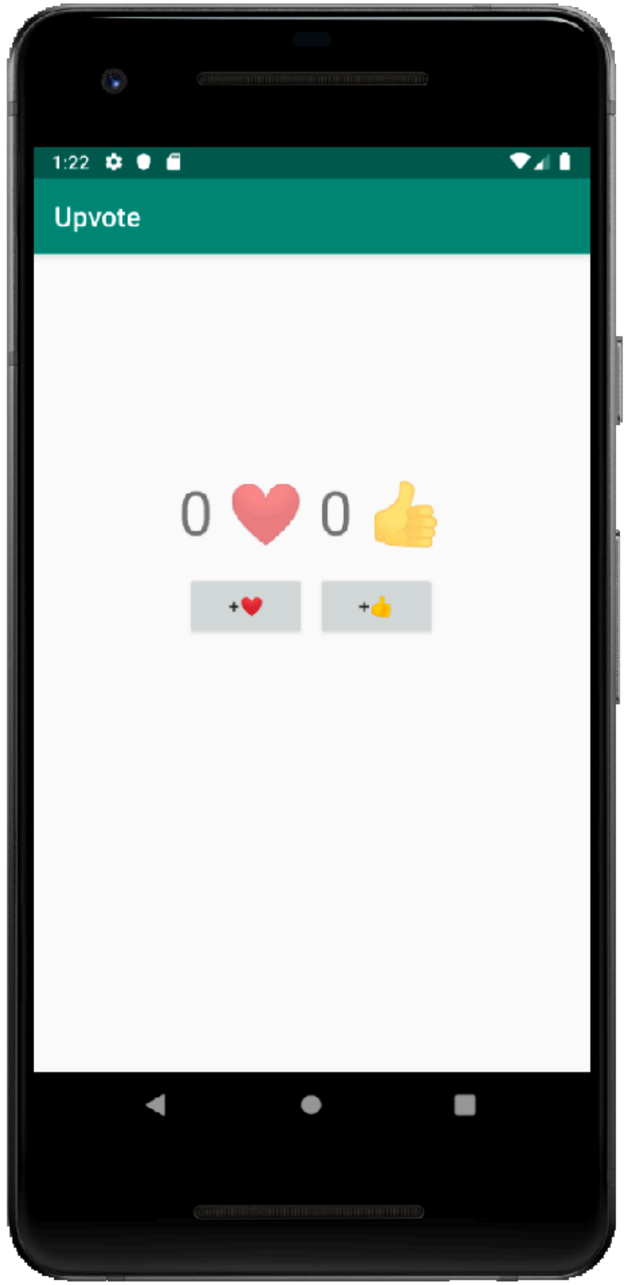



```
package com.jakewharton.rxbinding2.view
```



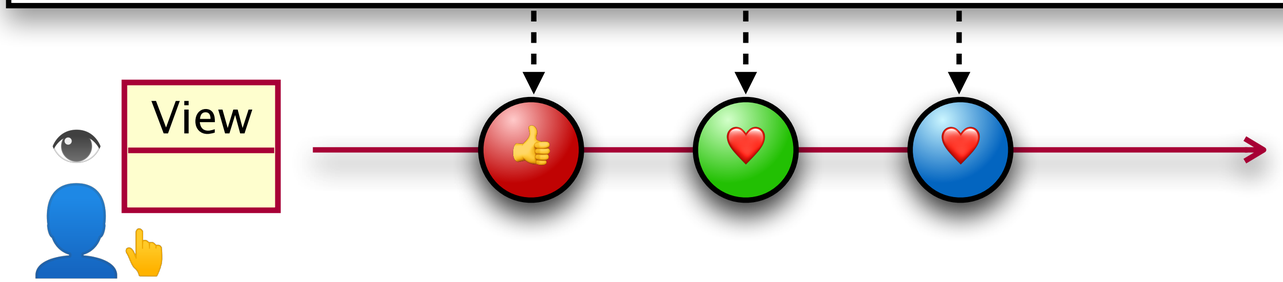

1:22
Upvote
0
0
+
+

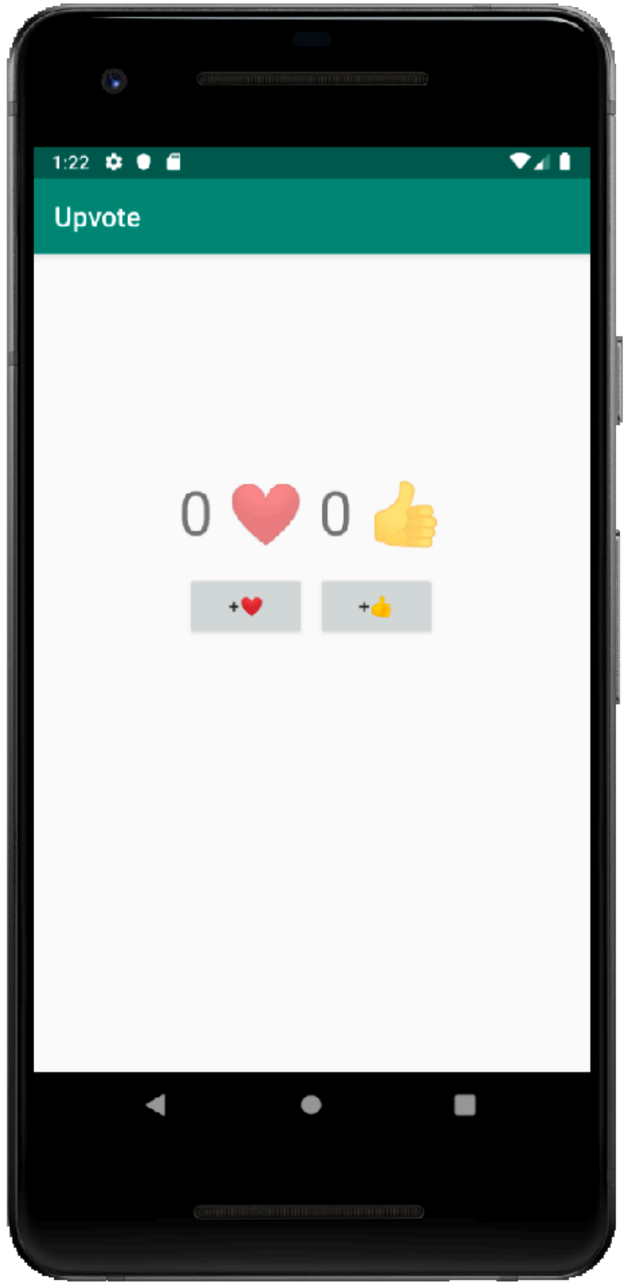

Observable<MainViewEvent>
View



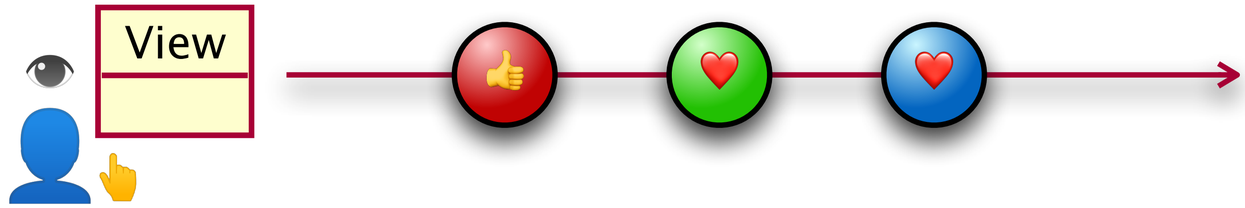
🎯 Observable<MainViewEvent>

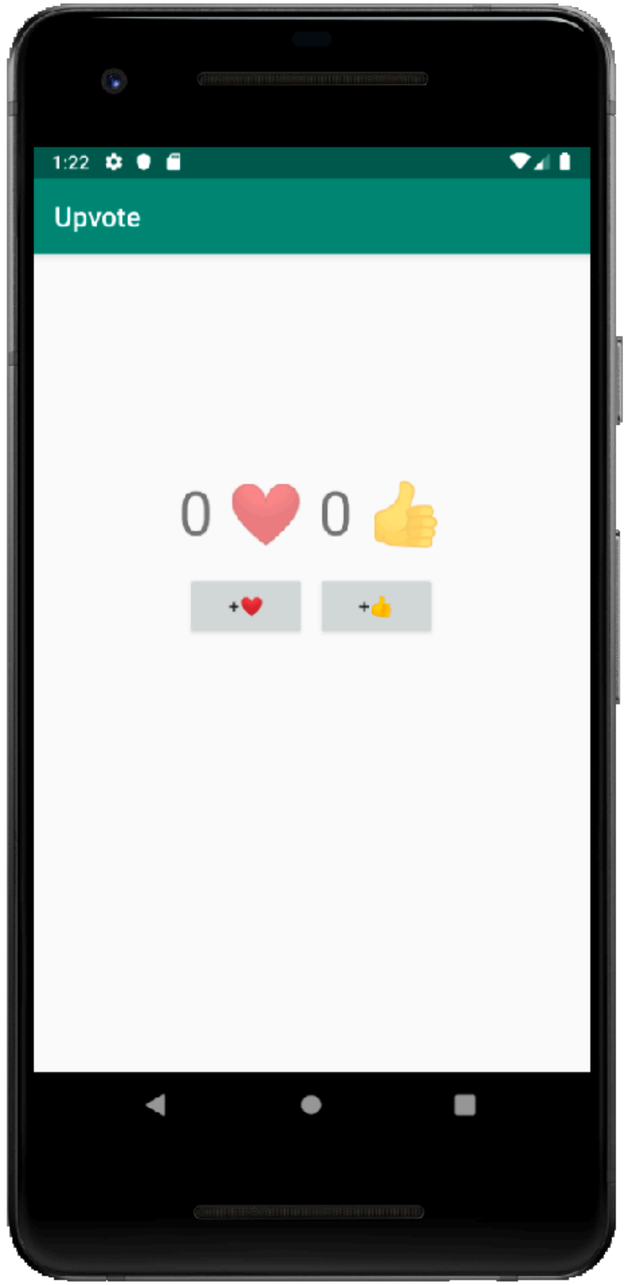
```
Observable.merge(
    heartButton.clicks().map { LoveItClick },
    thumbButton.clicks().map { ThumbsUpClick }
)
```



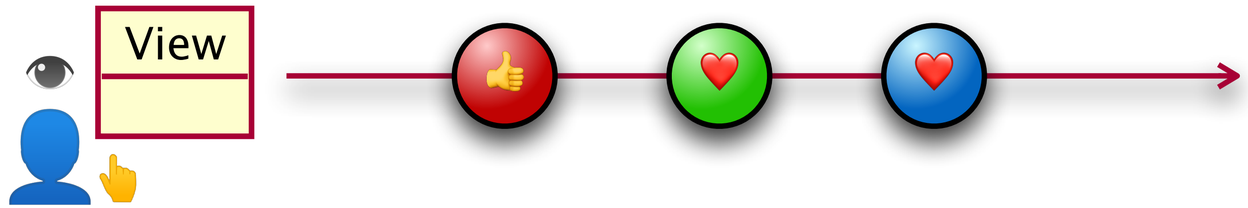


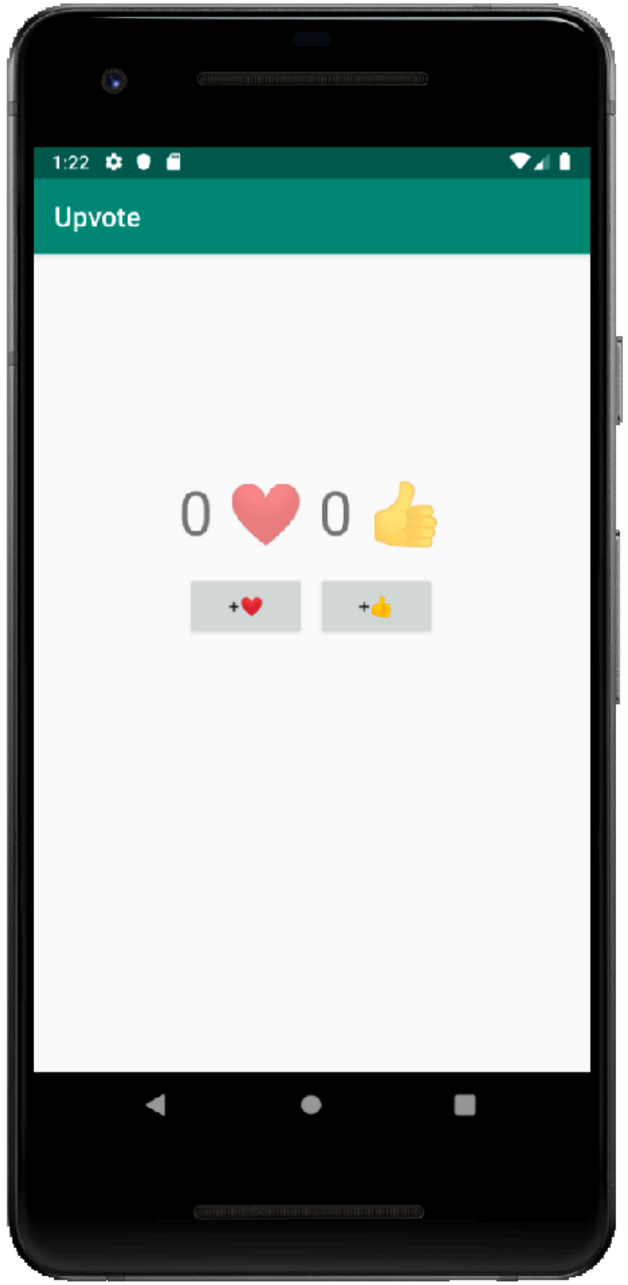

```
class MainActivity {

    override fun viewEvents(): Observable<MainViewEvent> {
        return Observable.merge(
                heartButton.clicks().map { LoveItClick },
                thumbButton.clicks().map { ThumbsUpClick }
        )
    }

}
```



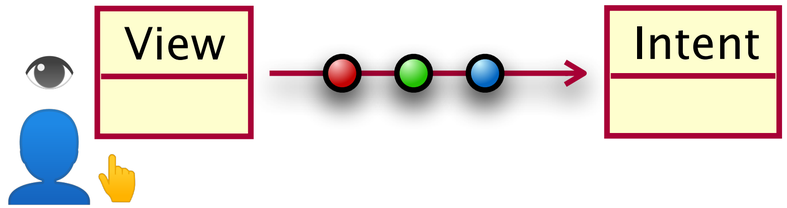


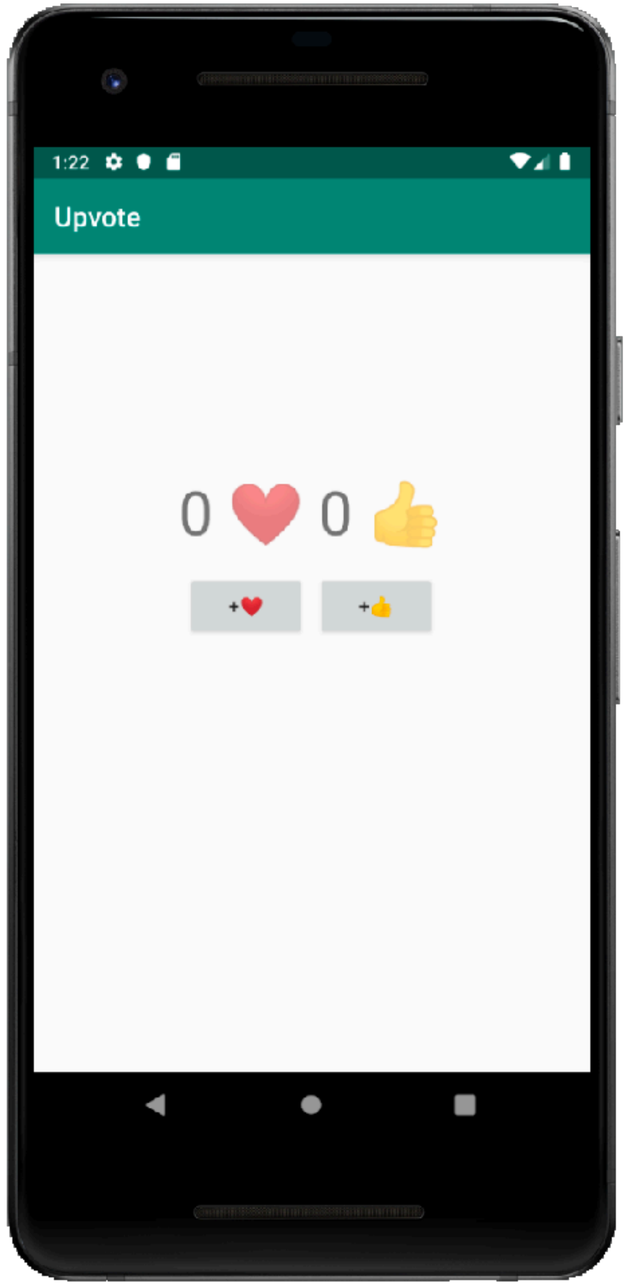

```kotlin
/**
 * This allows us to group all the viewEvents from
 * one view in a single Observable.
 */
interface ViewEventObservable<E> {
    fun viewEvents(): Observable<E>
}

class MainActivity : ViewEventObservable<MainViewEvent> {

    override fun viewEvents(): Observable<MainViewEvent> {
        return Observable.merge(
                heartButton.clicks().map { LoveItClick },
                thumbButton.clicks().map { ThumbsUpClick }
        )
    }

}
```



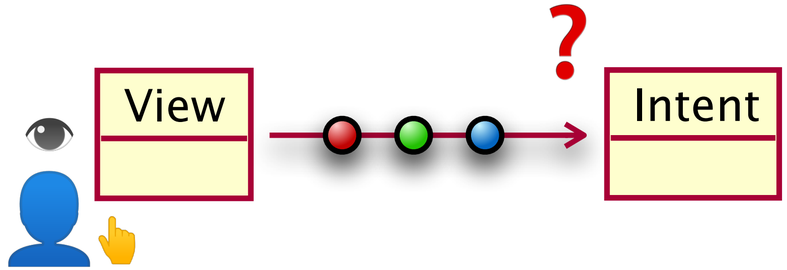


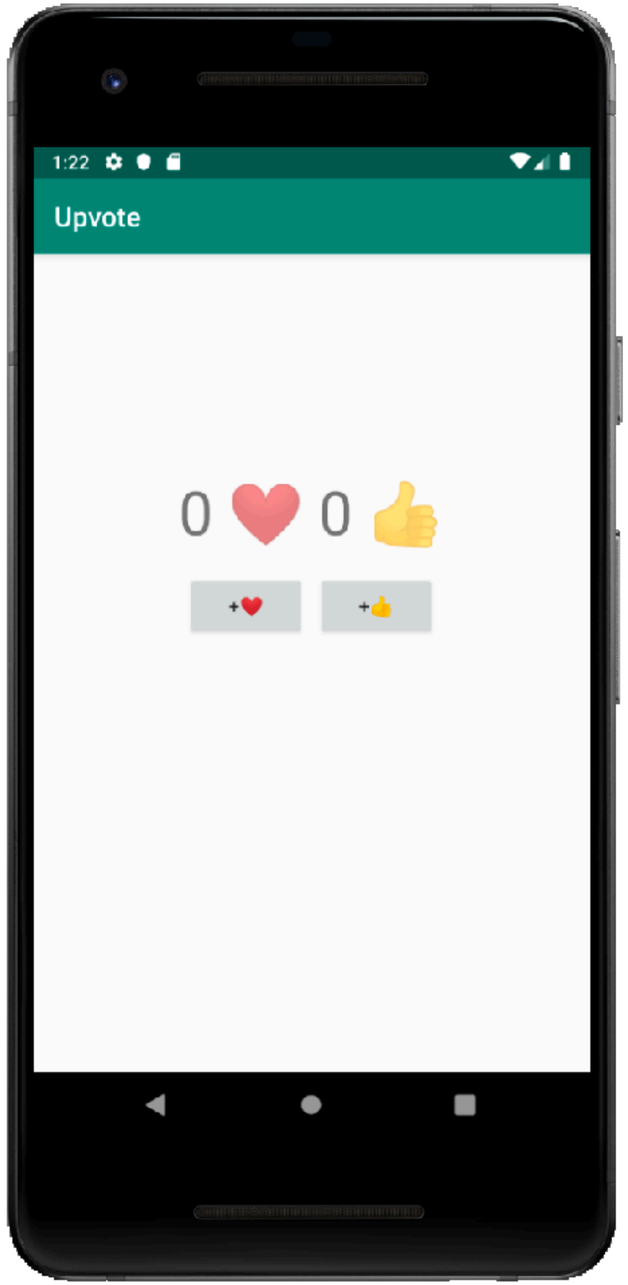

```
class MainActivity : ViewEventObservable<MainViewEvent> {

    private val disposables = CompositeDisposable()

    override fun onResume() {
        super.onResume()
        disposables += viewEvents().subscribe(   )
    }

    override fun onPause() {
        super.onPause()
        disposables.clear()
    }

    override fun viewEvents(): Observable<MainViewEvent> {
        return Observable.merge(
                heartButton.clicks().map { LoveItClick },
                thumbButton.clicks().map { ThumbsUpClick }
        )
    }

}
```



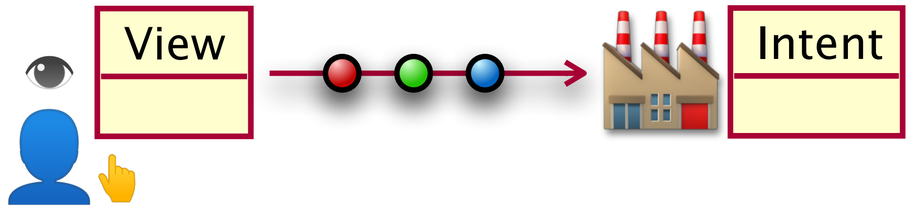


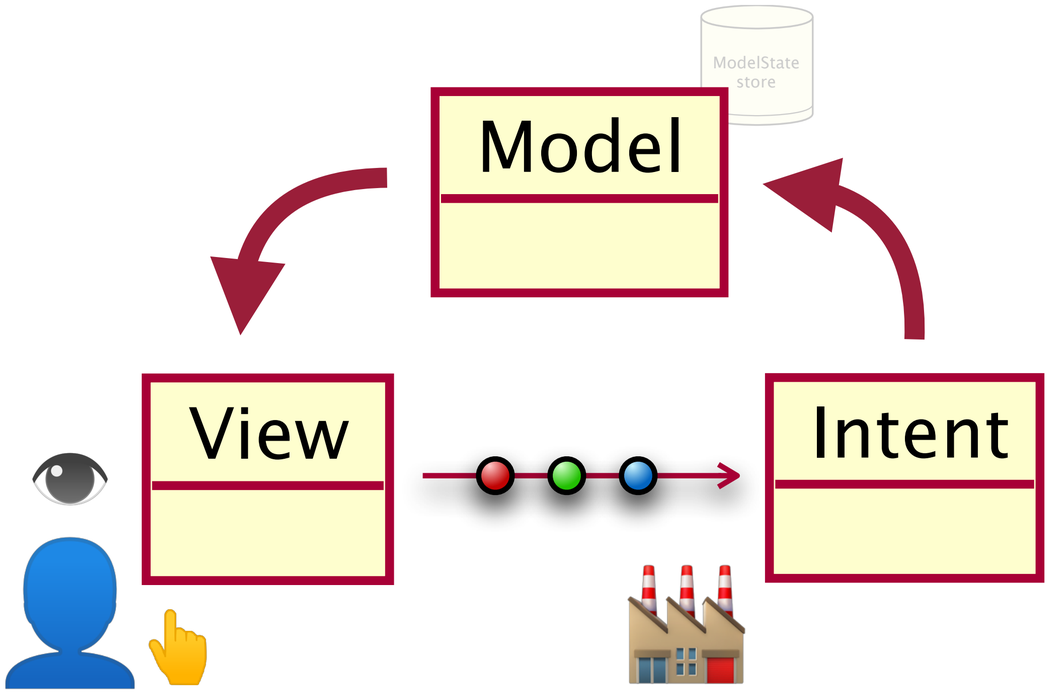

```
class MainActivity : ViewEventObservable<MainViewEvent> {

    private val disposables = CompositeDisposable()

    override fun onResume() {
        super.onResume()
        disposables += viewEvents().subscribe( ? )
    }

    override fun onPause() {
        super.onPause()
        disposables.clear()
    }

    override fun viewEvents(): Observable<MainViewEvent> {
        return Observable.merge(
                heartButton.clicks().map { LoveItClick },
                thumbButton.clicks().map { ThumbsUpClick }
        )
    }

}
```

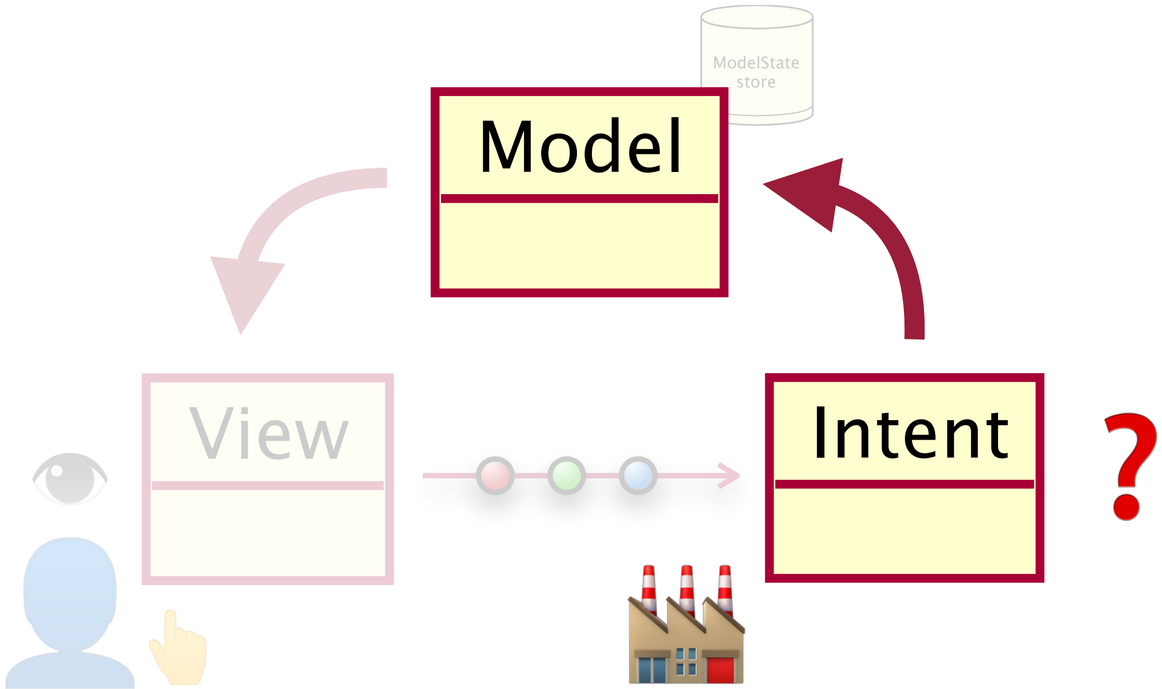


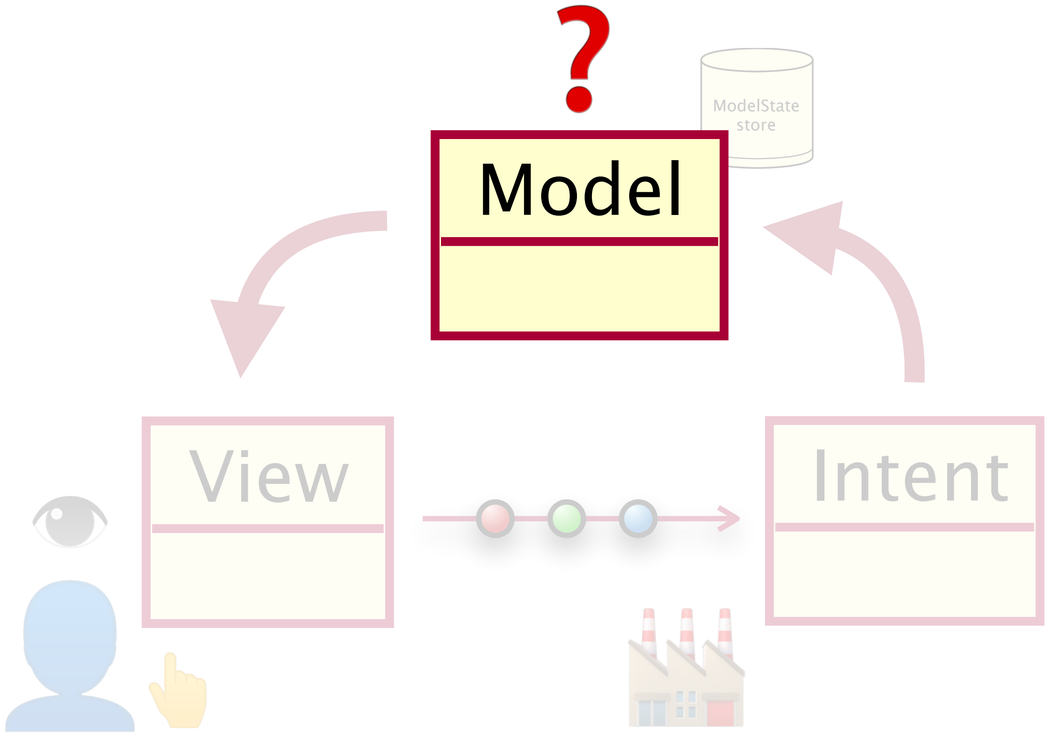


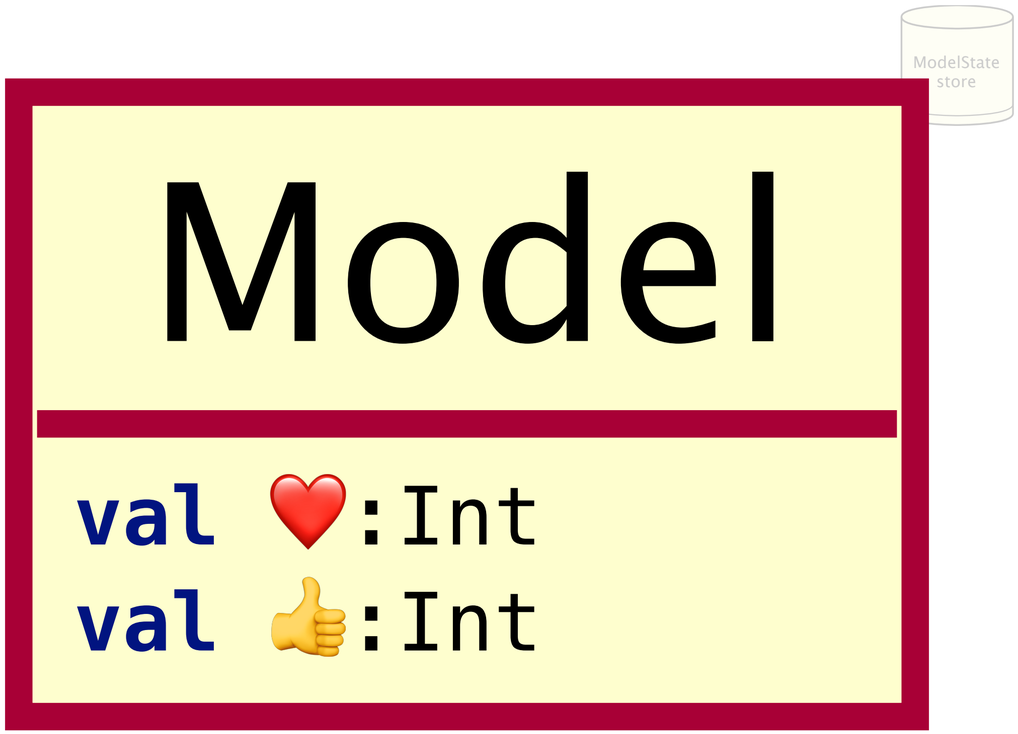

```
class MainActivity : ViewEventObservable<MainViewEvent> {

    private val disposables = CompositeDisposable()

    override fun onResume() {
        super.onResume()
        disposables += viewEvents().subscribe(
            MainViewIntentFactory::process
        )
    }

    override fun onPause() {
        super.onPause()
        disposables.clear()
    }

    override fun viewEvents(): Observable<MainViewEvent> {
        return Observable.merge(
                heartButton.clicks().map { LoveItClick },
                thumbButton.clicks().map { ThumbsUpClick }
        )
    }

}
```

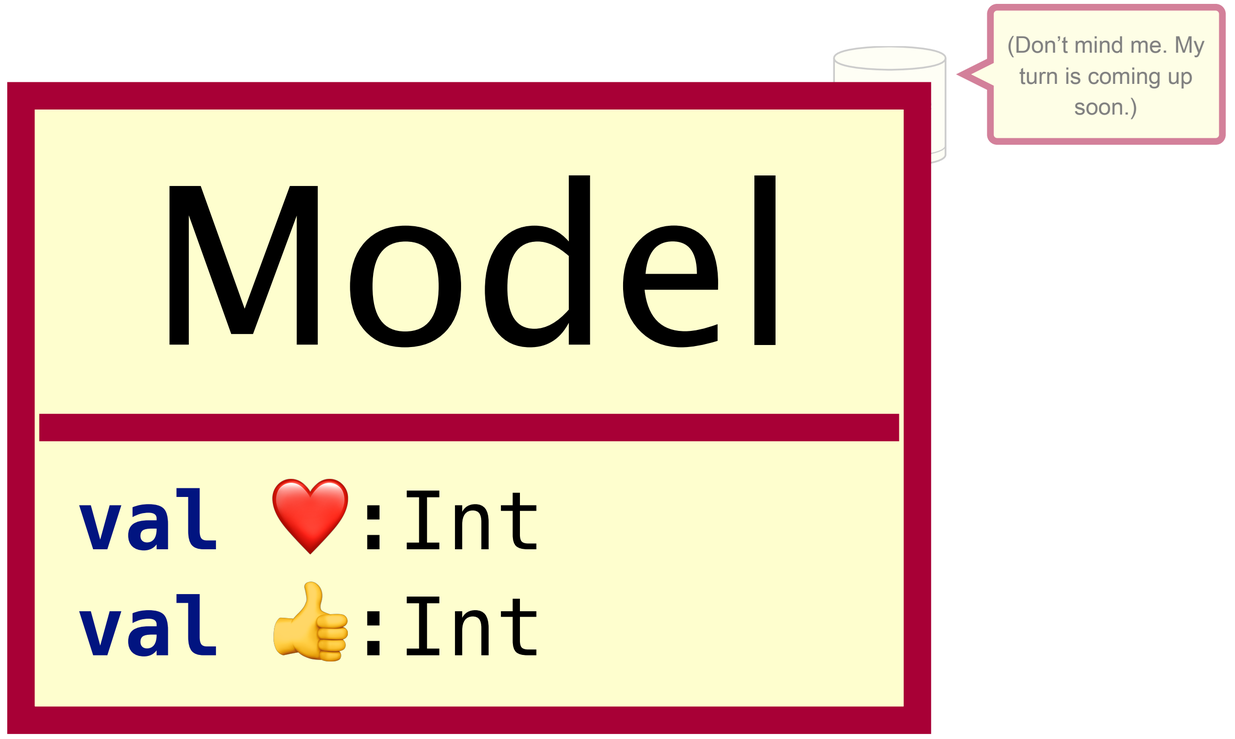



ModelState
store
Model
View
Intent


ModelState
store
Model
View
Intent
?

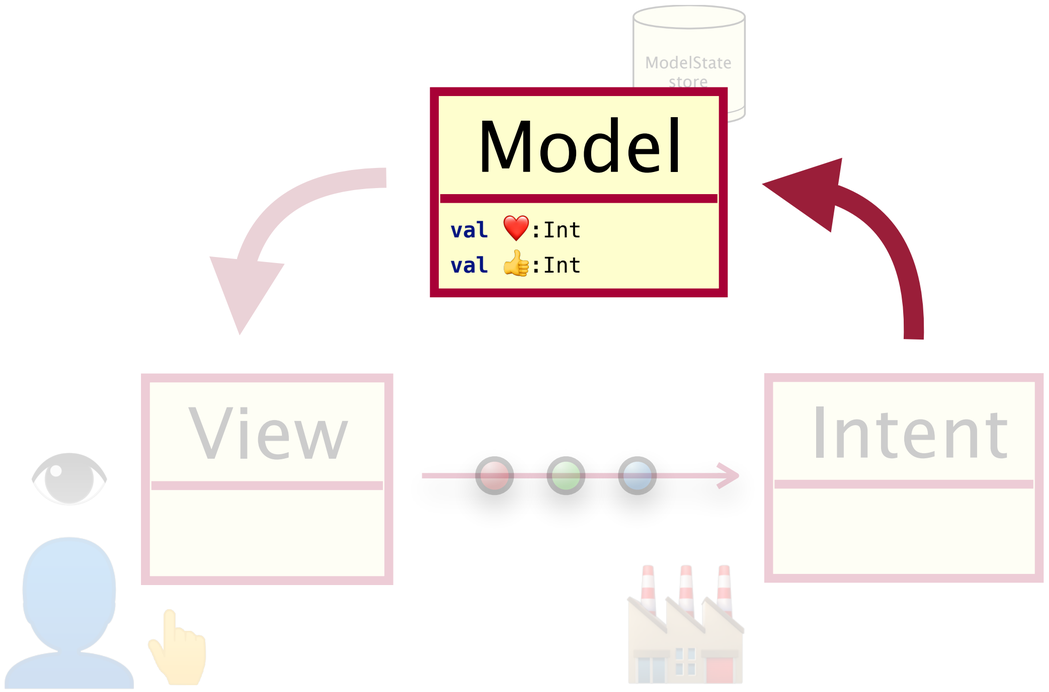

?
ModelState
store
Model
View
Intent


ModelState
store
Model
val ❤️:Int
val 👍:Int

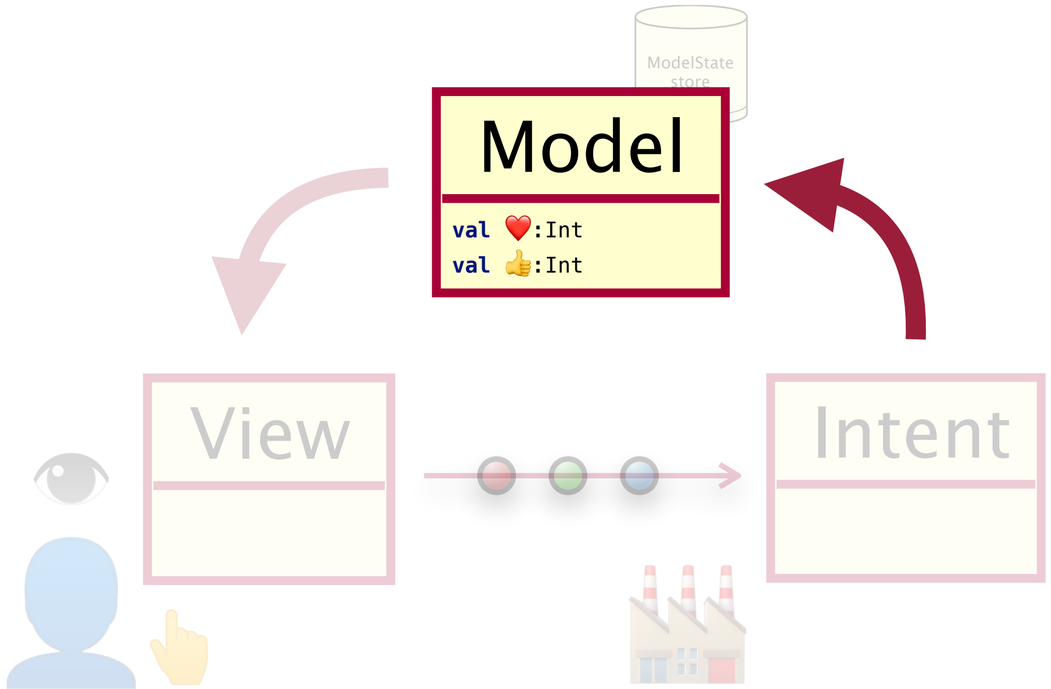



```
data class UpvoteModel(val hearts:Int, val thumbs:Int)
```

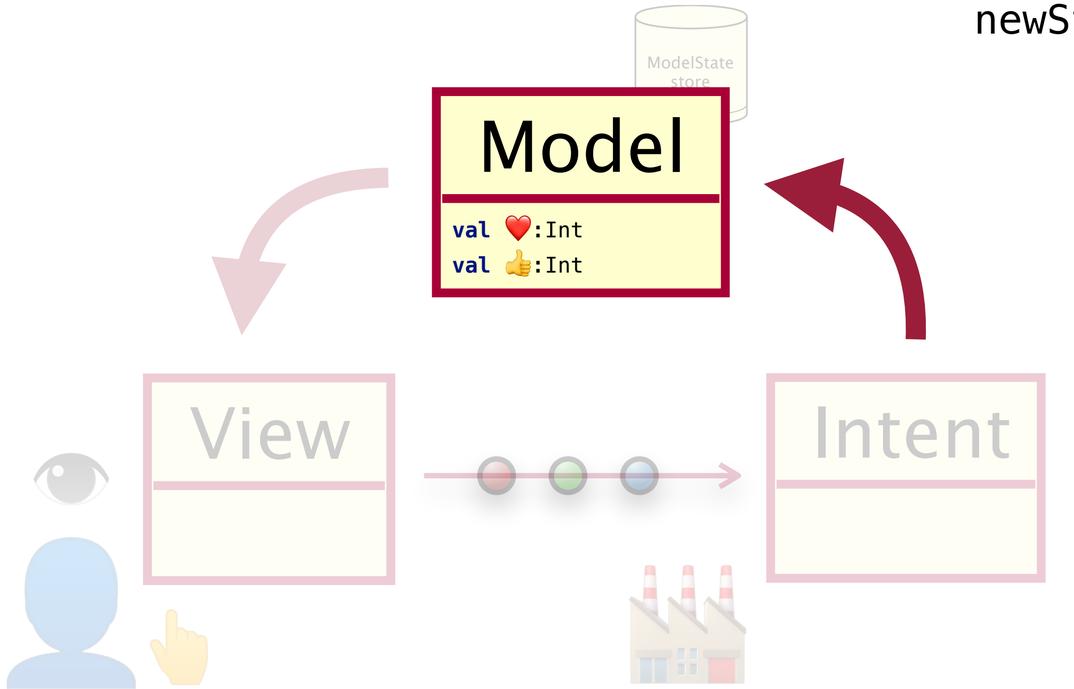
```
data class UpvoteModel(val hearts:Int, val thumbs:Int)
```



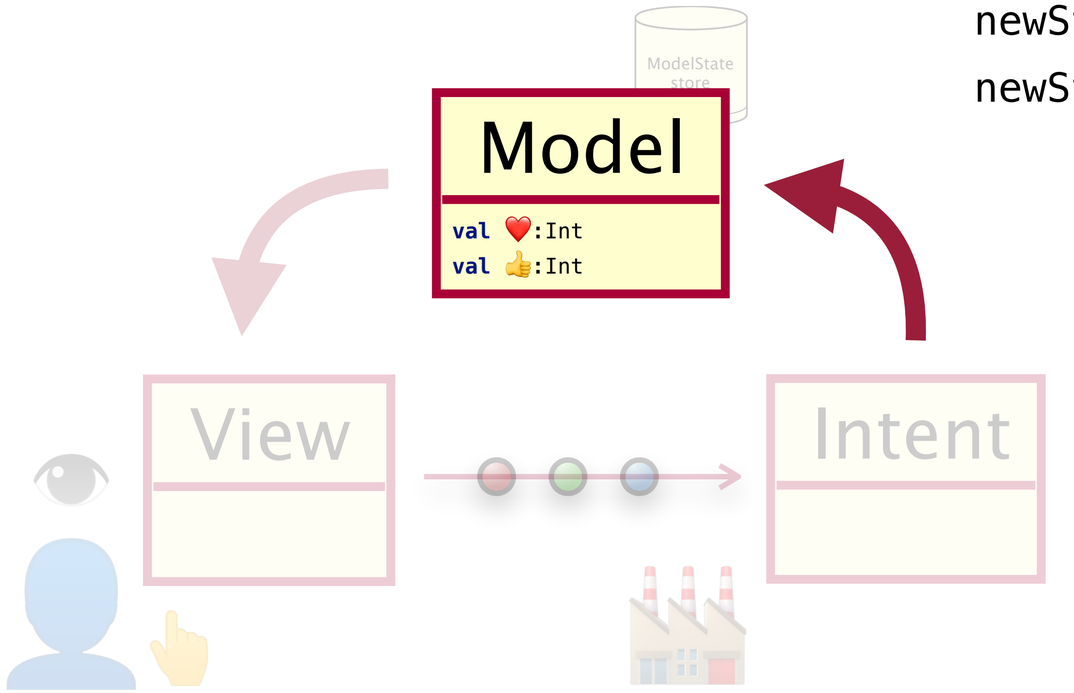
```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

newState = oldState.copy(thumbs = thumbs + 1) // 0, 1
```



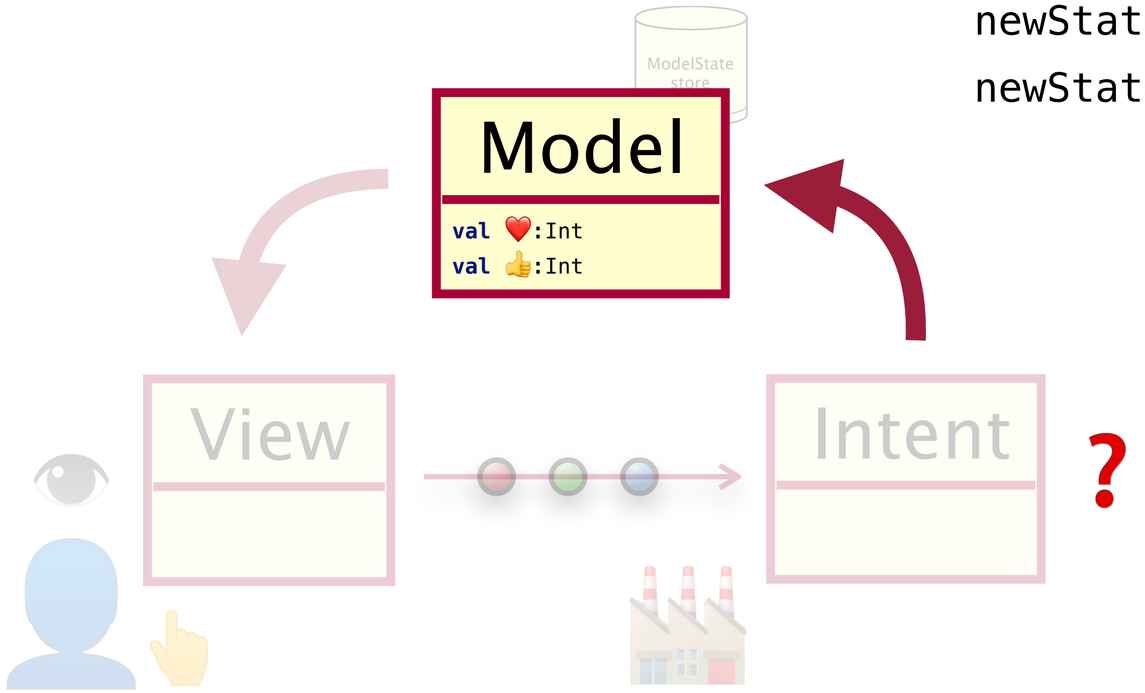
```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

newState = oldState.copy(thumbs = thumbs + 1) // 0, 1
newState = newState.copy(hearts = hearts + 1) // 1, 1
```



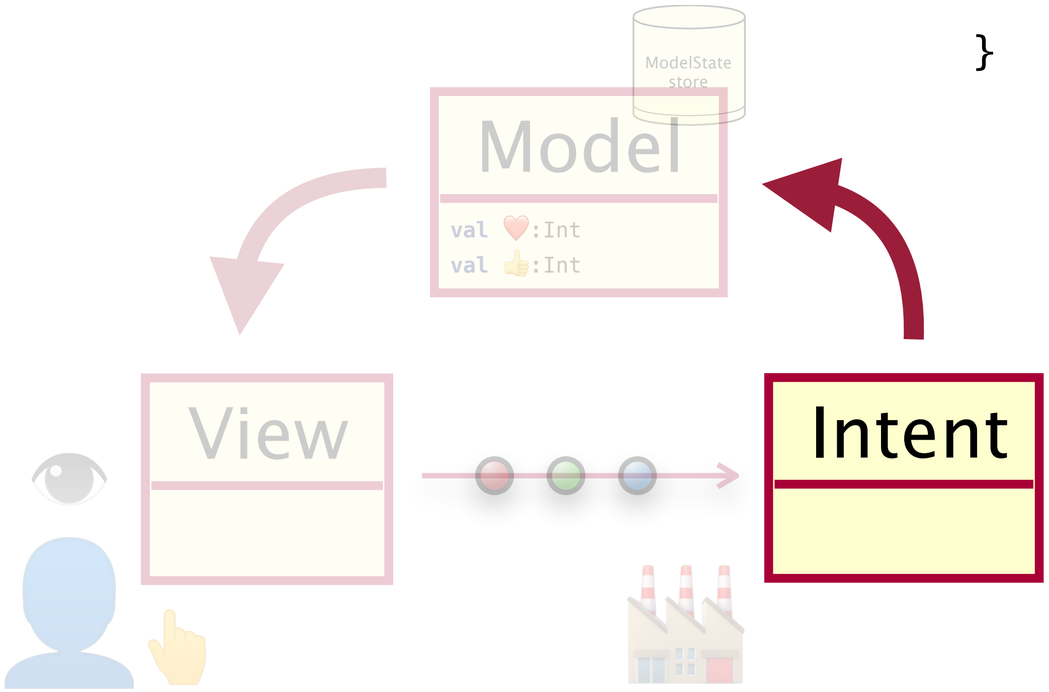
```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

newState = oldState.copy(thumbs = thumbs + 1) // 0, 1
newState = newState.copy(hearts = hearts + 1) // 1, 1
newState = newState.copy(hearts = hearts + 1) // 2, 1
```

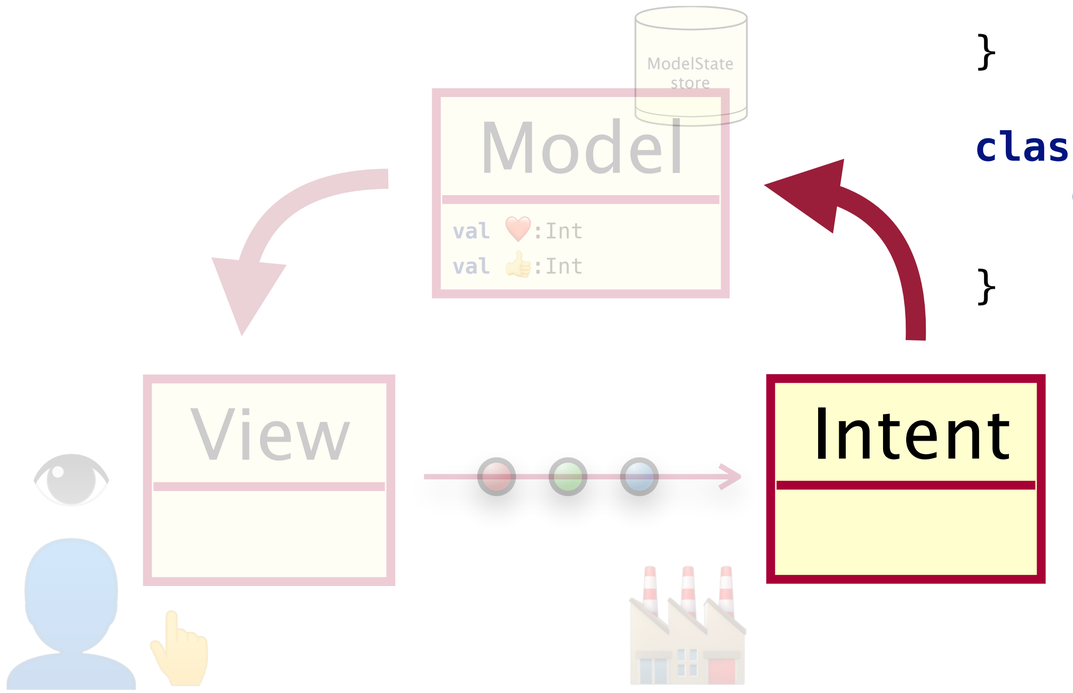


```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

newState = oldState.copy(thumbs = thumbs + 1) // 0, 1
newState = newState.copy(hearts = hearts + 1) // 1, 1
newState = newState.copy(hearts = hearts + 1) // 2, 1
```

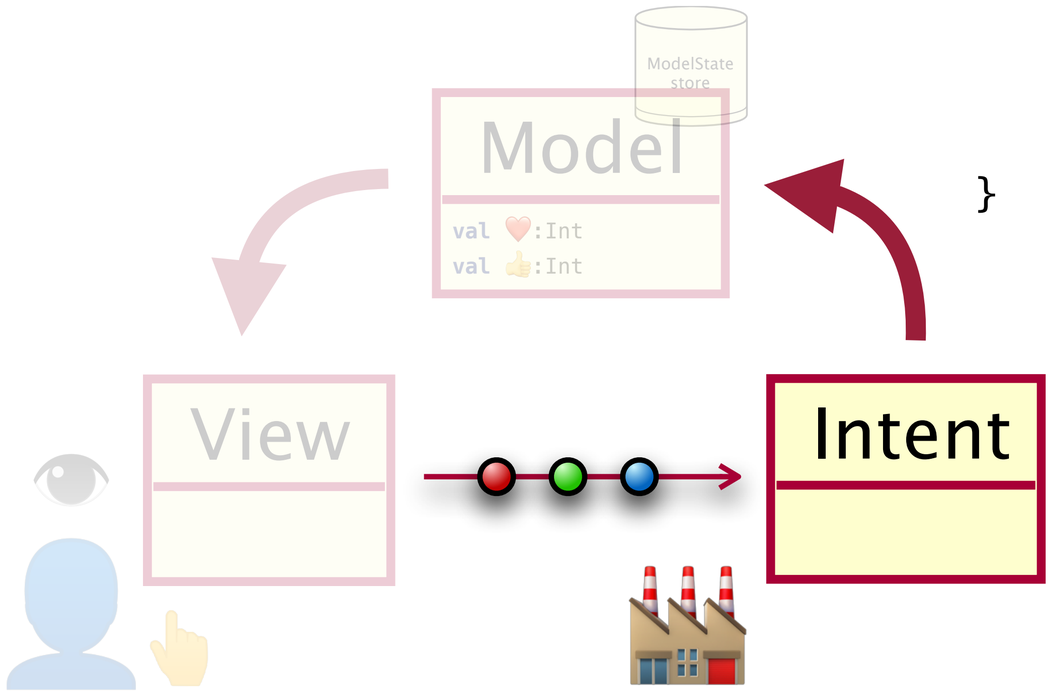


```kotlin
data class UpvoteModel(val hearts:Int, val thumbs:Int)

interface Intent<T> {
    fun reduce(oldState: T): T
}
```



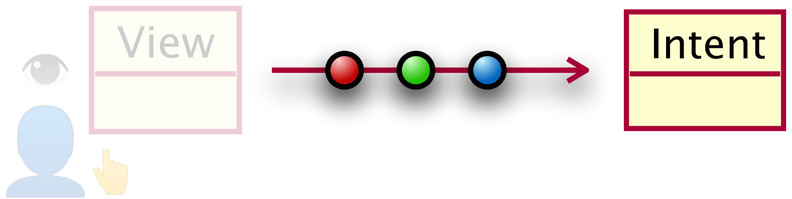


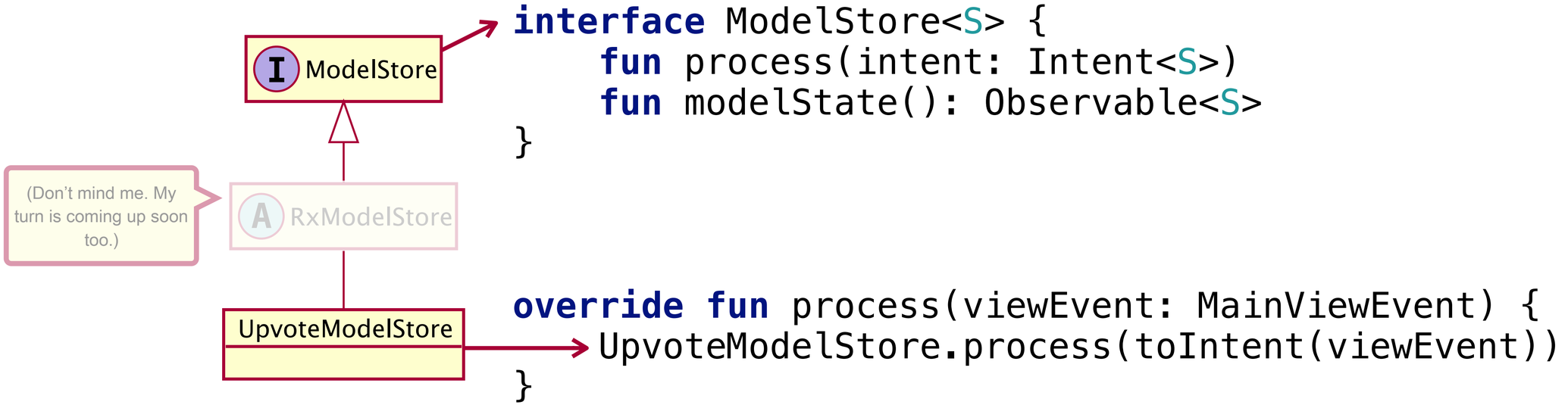

```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

interface Intent<T> {
    fun reduce(oldState: T): T
}

class AddHeart():Intent<UpvoteModel> {
    override fun reduce(oldState: UpvoteModel) =
        oldState.copy(hearts = oldState.hearts + 1)
}
```




```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

fun toIntent(viewEvent: MainViewEvent):Intent<UpvoteModel> {
    return when (viewEvent) {
        MainViewEvent.LoveItClick -> AddHeart()
        MainViewEvent.ThumbsUpClick -> AddThumb()
    }
}
```

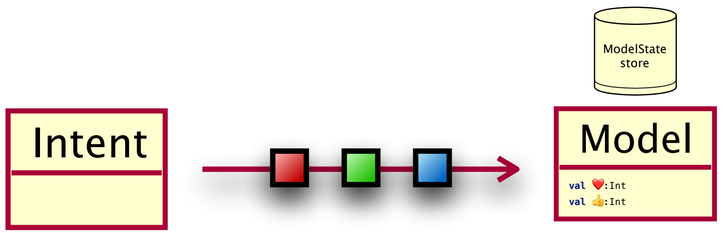
```
interface IntentFactory<E> {
    fun process(viewEvent:E)
}

object MainViewIntentFactory : IntentFactory<MainViewEvent> {

    override fun process(viewEvent: MainViewEvent) {
        UpvoteModelStore.process(toIntent(viewEvent))
    }

    private fun toIntent(viewEvent: MainViewEvent):Intent<UpvoteModel> {
        return when (viewEvent) {
            MainViewEvent.LoveItClick -> AddHeart()
            MainViewEvent.ThumbsUpClick -> AddThumb()
        }
    }

}
```

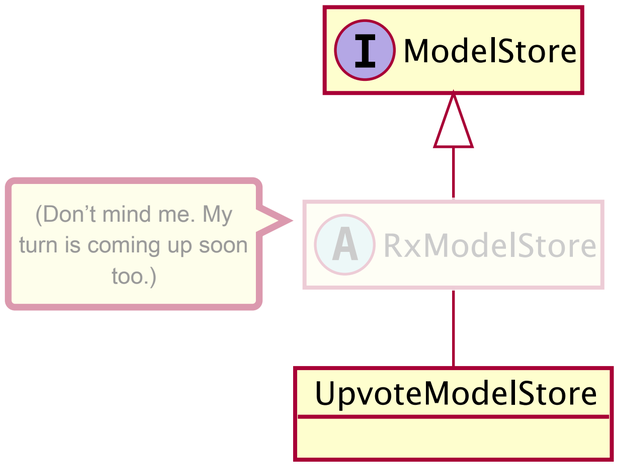




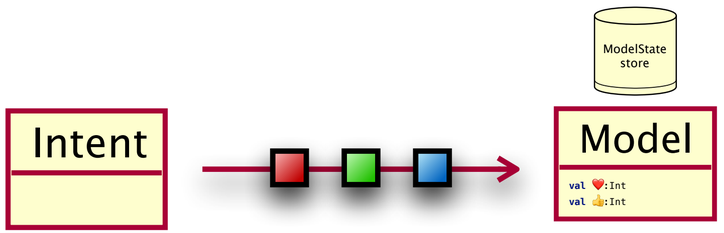

```
interface ModelStore<S> {
    fun process(intent: Intent<S>)
    fun modelState(): Observable<S>
}
```

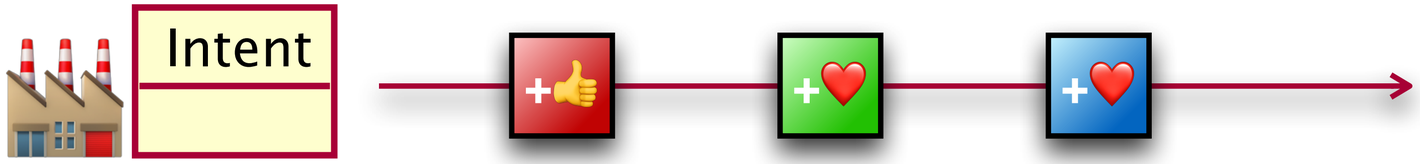
```
override fun process(viewEvent: MainViewEvent) {
    UpvoteModelStore.process(toIntent(viewEvent))
}
```

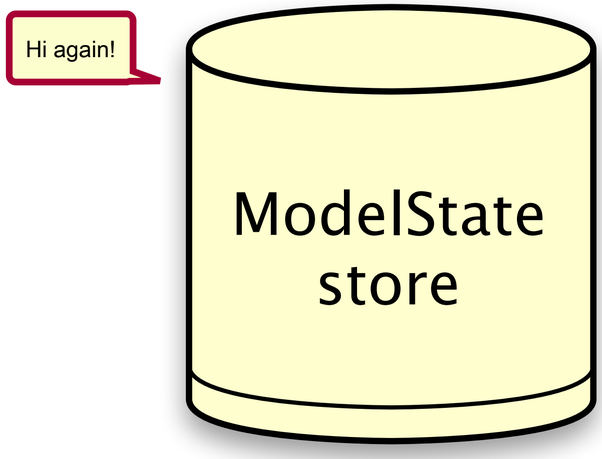




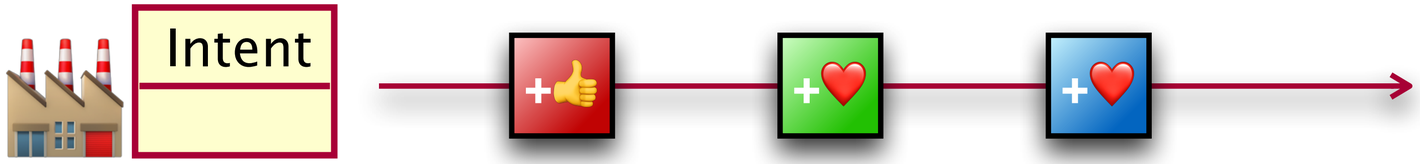

```
interface ModelStore<S> {
    fun process(intent: Intent<S>)
    fun modelState(): Observable<S>
}
```







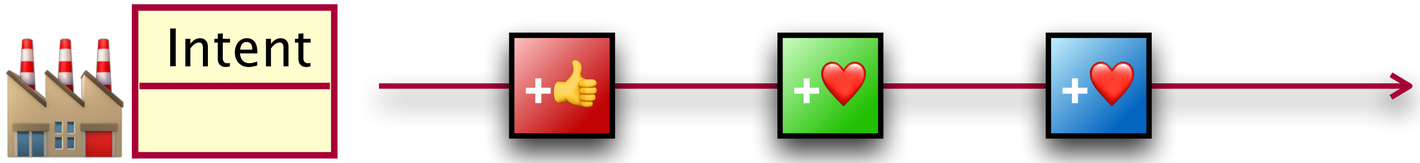

```
interface ModelStore<S> {
     fun process(intent: Intent<S>)
     fun modelState(): Observable<S>
}
```




```
import com.jakewharton.rxrelay2.PublishRelay
```

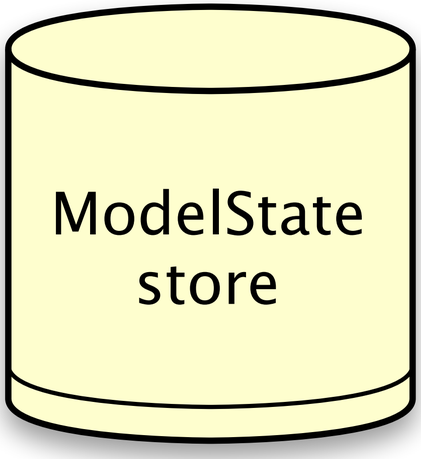
```
PublishRelay<Intent>.accept(intent)
```

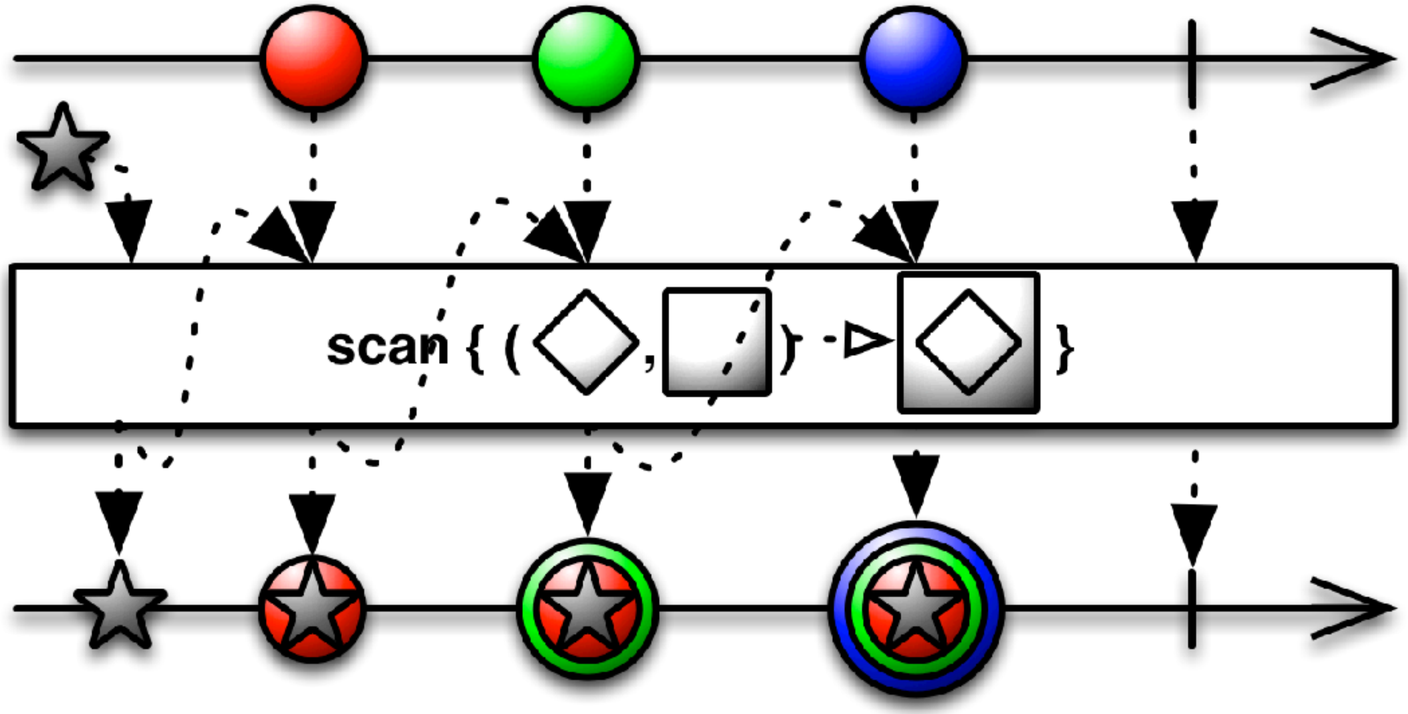

```
fun process(intent: Intent<S>) = intents.accept(intent)
```

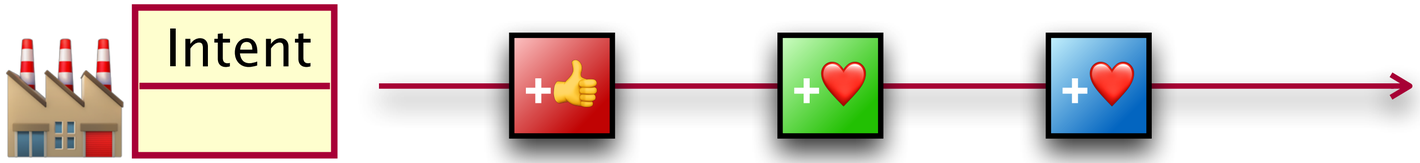



```
import com.jakewharton.rxrelay2.PublishRelay
```

```
PublishRelay<Intent>.accept(intent)
```

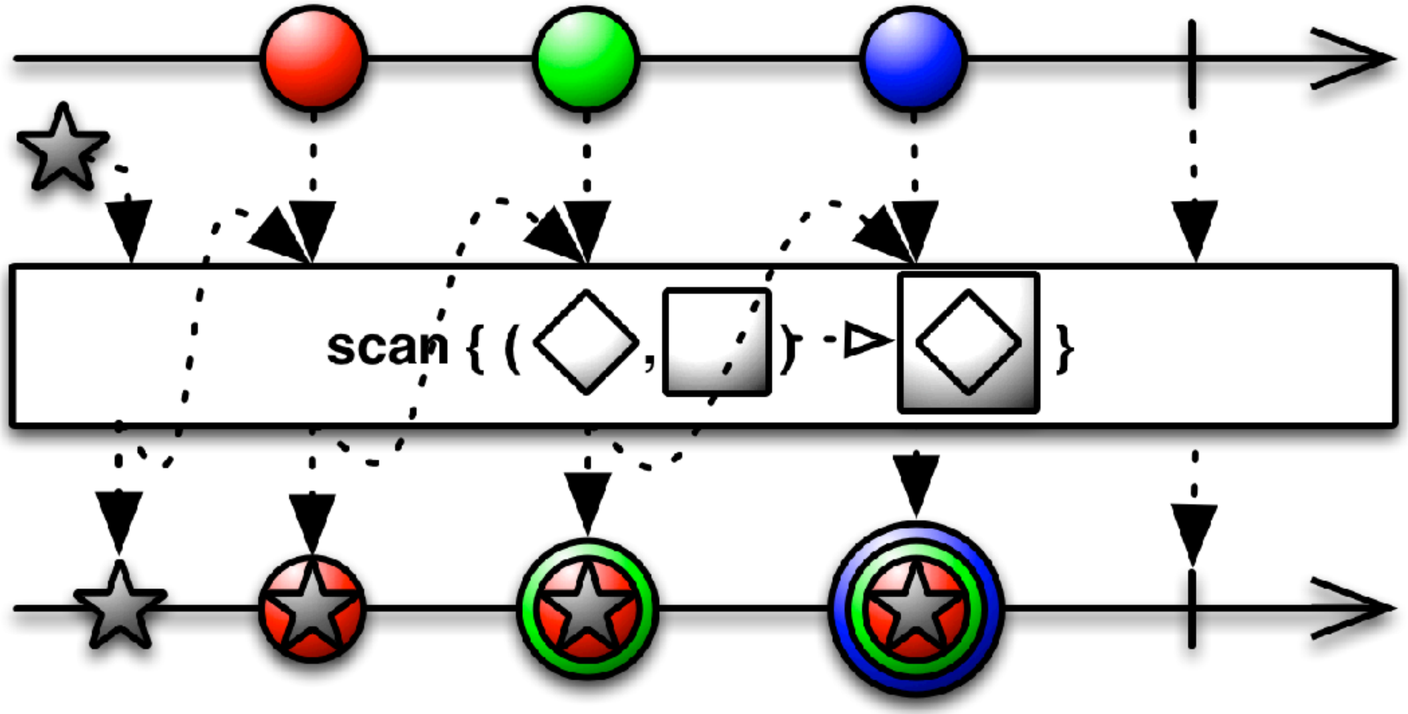


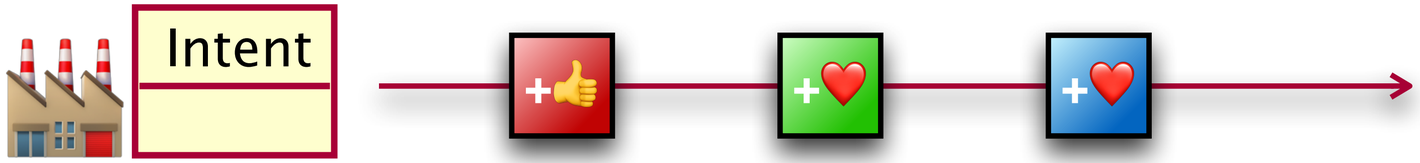





```
import com.jakewharton.rxrelay2.PublishRelay
```

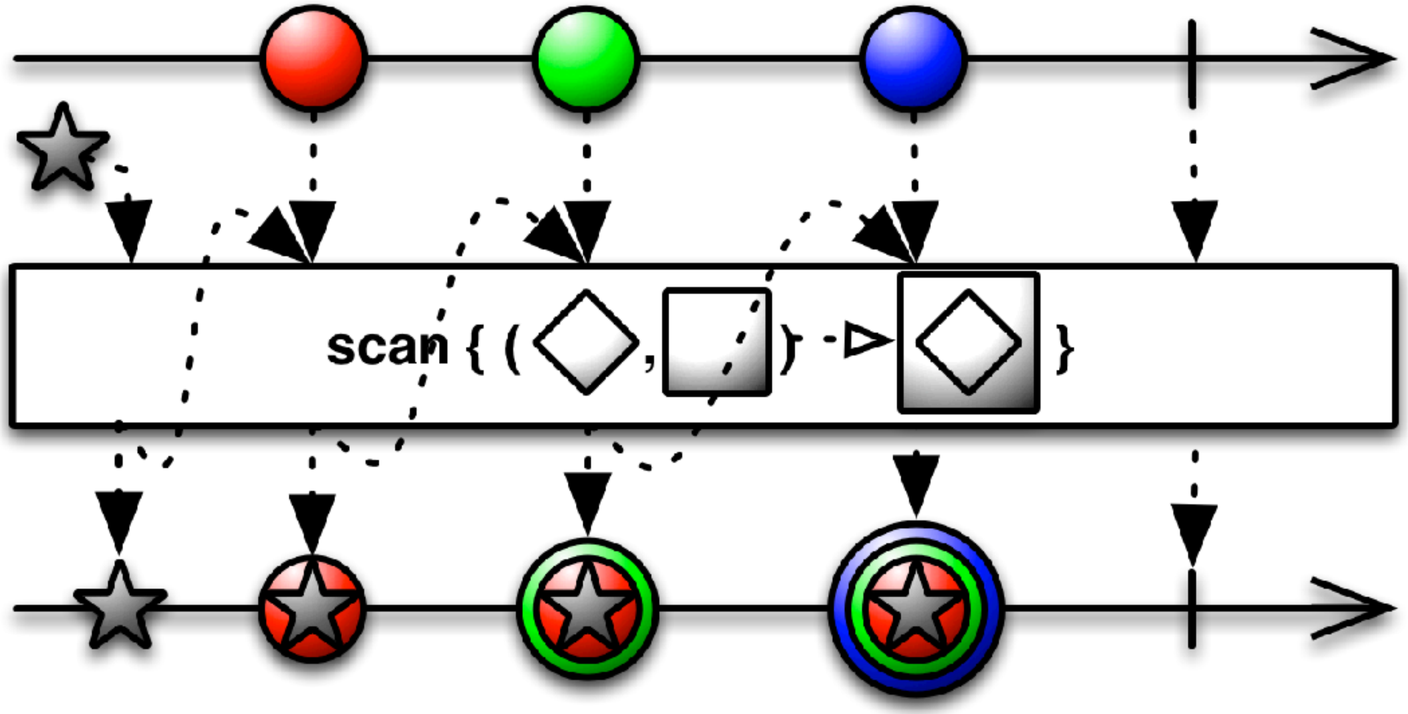
```
PublishRelay<Intent>.accept(intent)
```

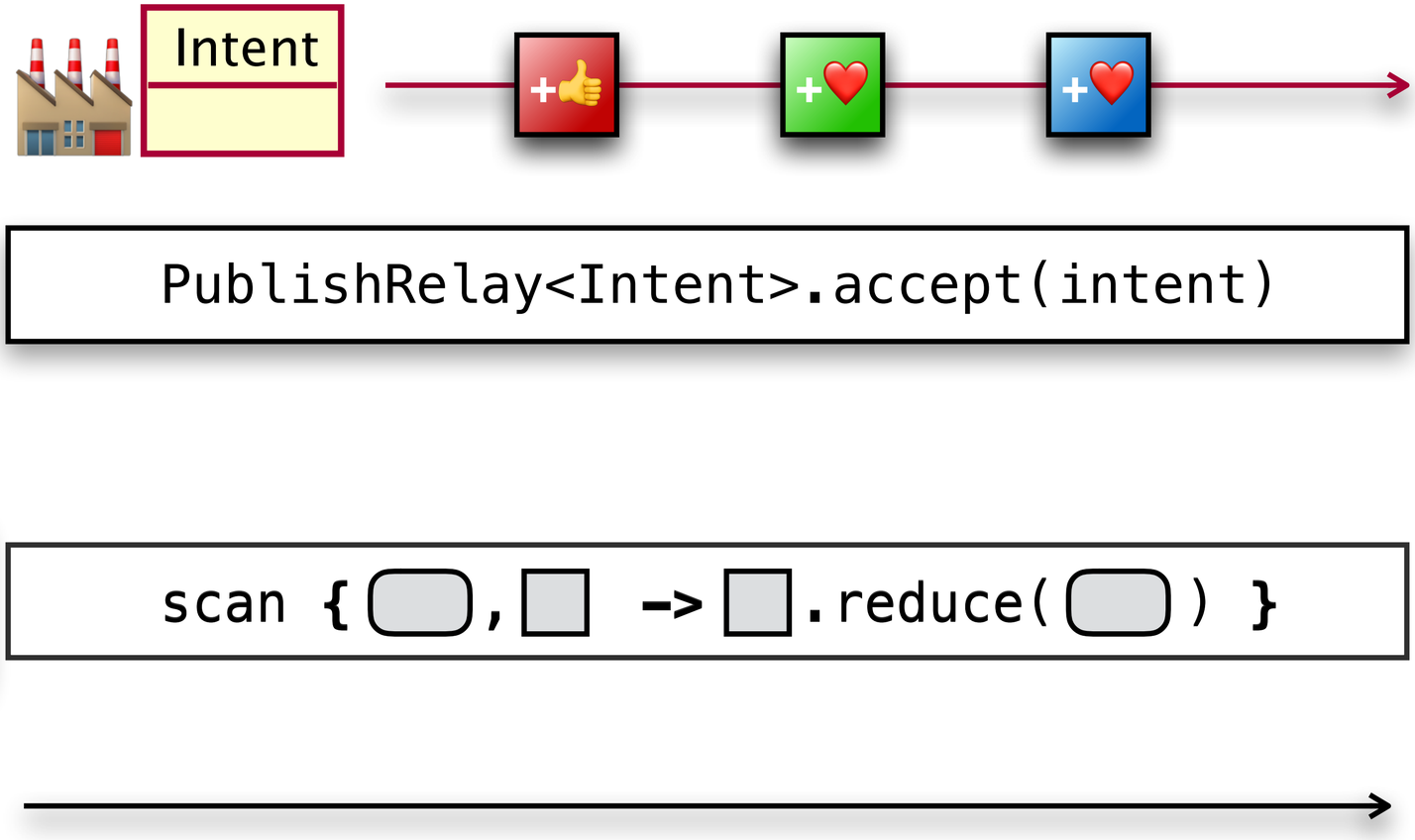


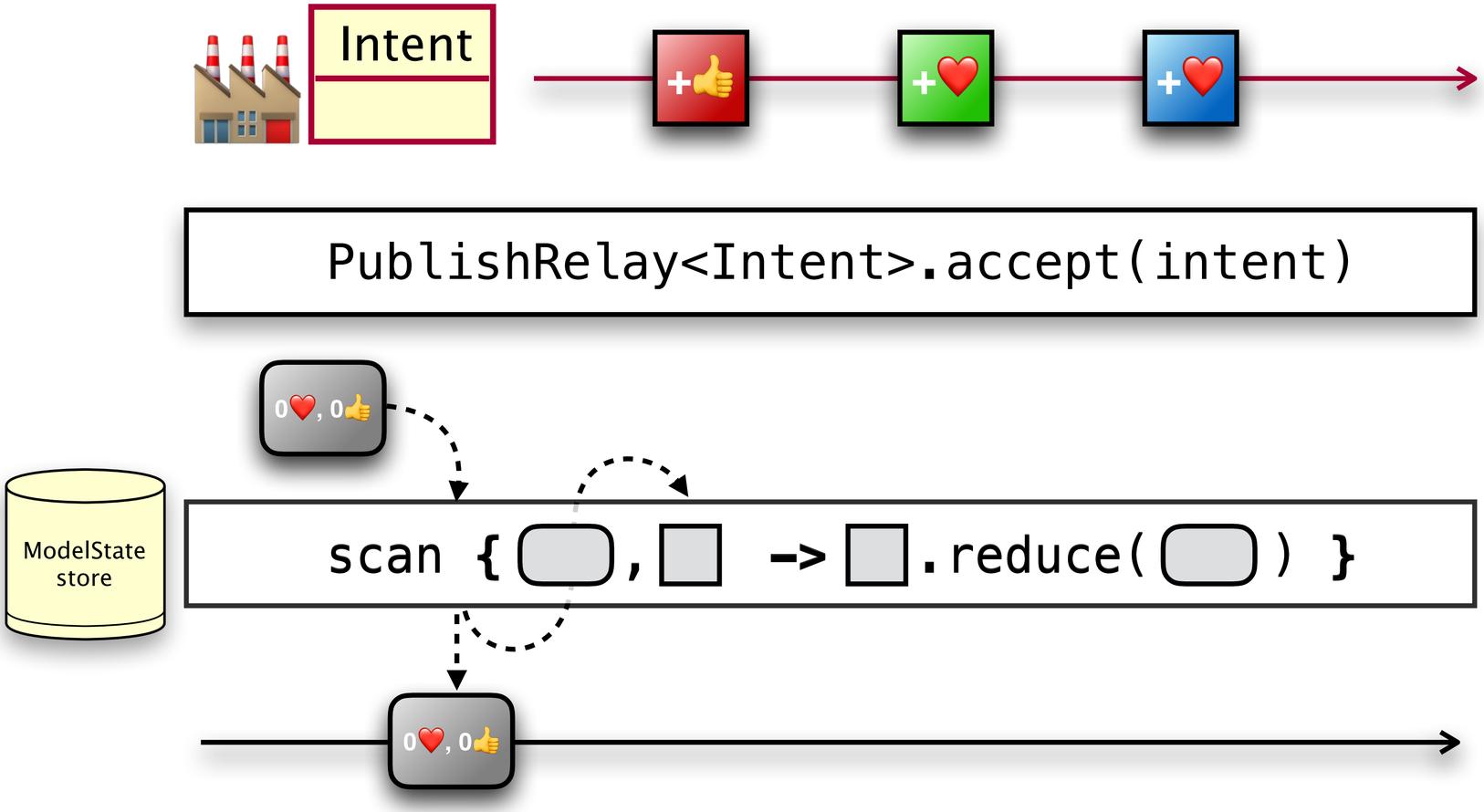


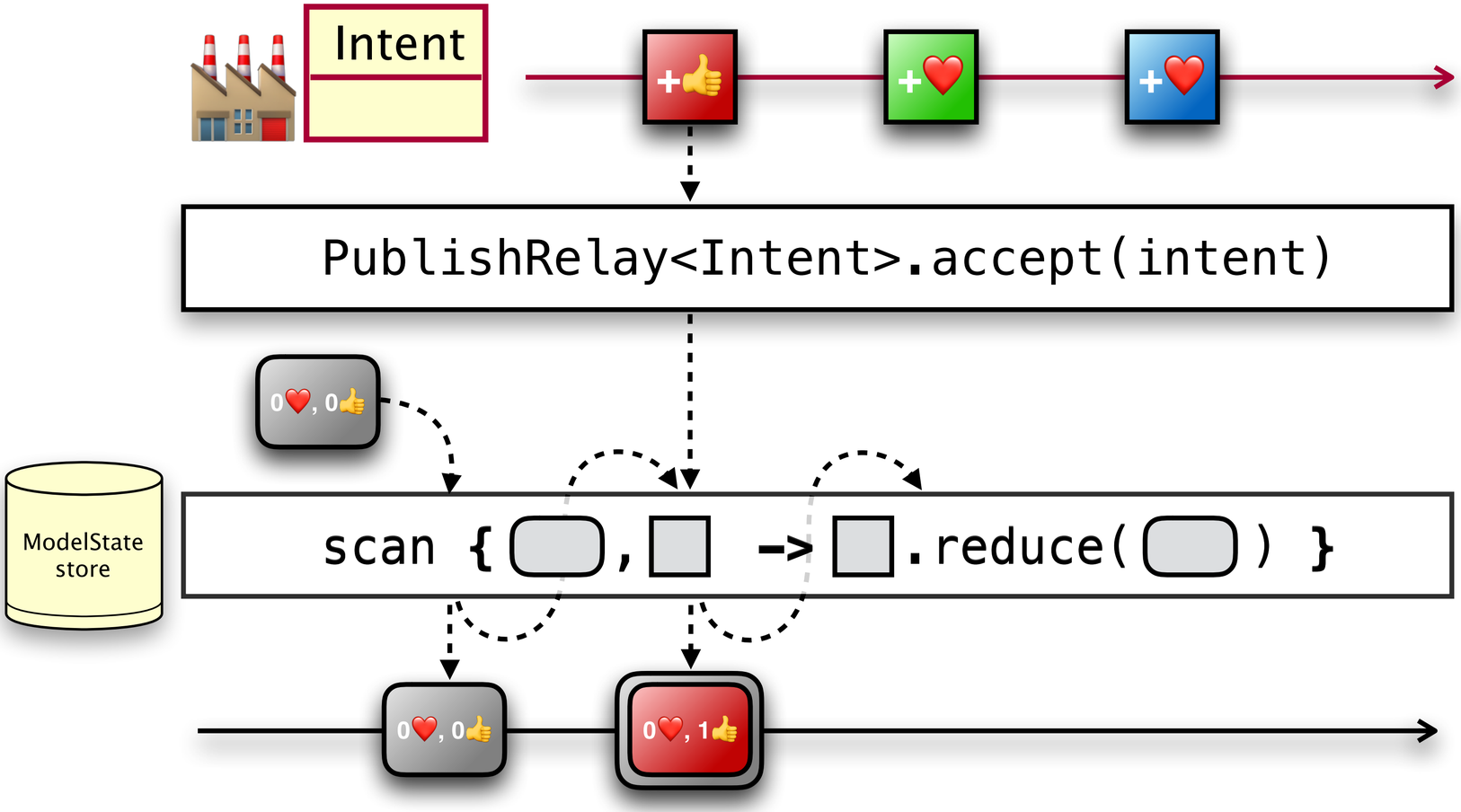

```
import com.jakewharton.rxrelay2.PublishRelay
```

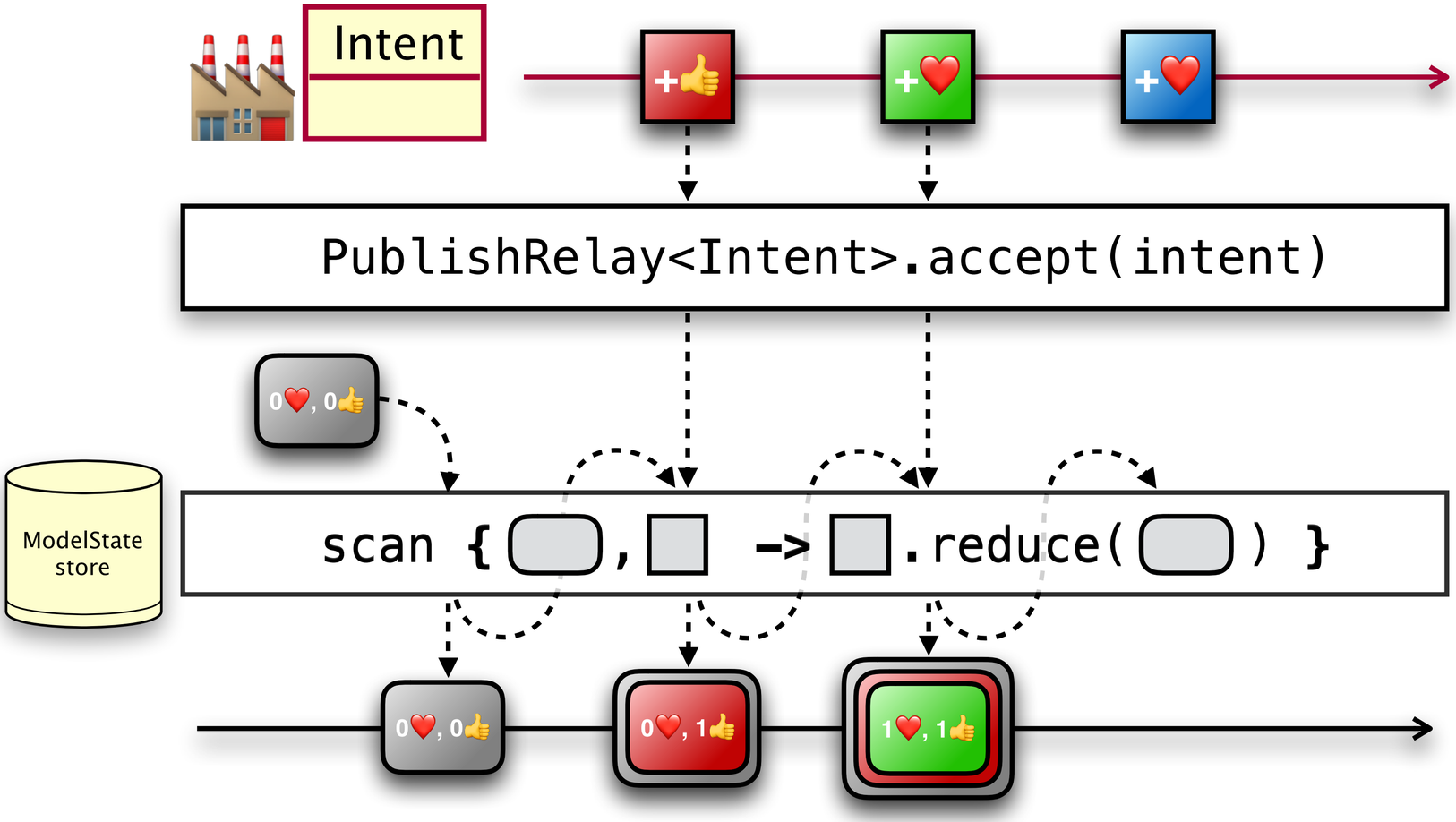
```
PublishRelay<Intent>.accept(intent)
```

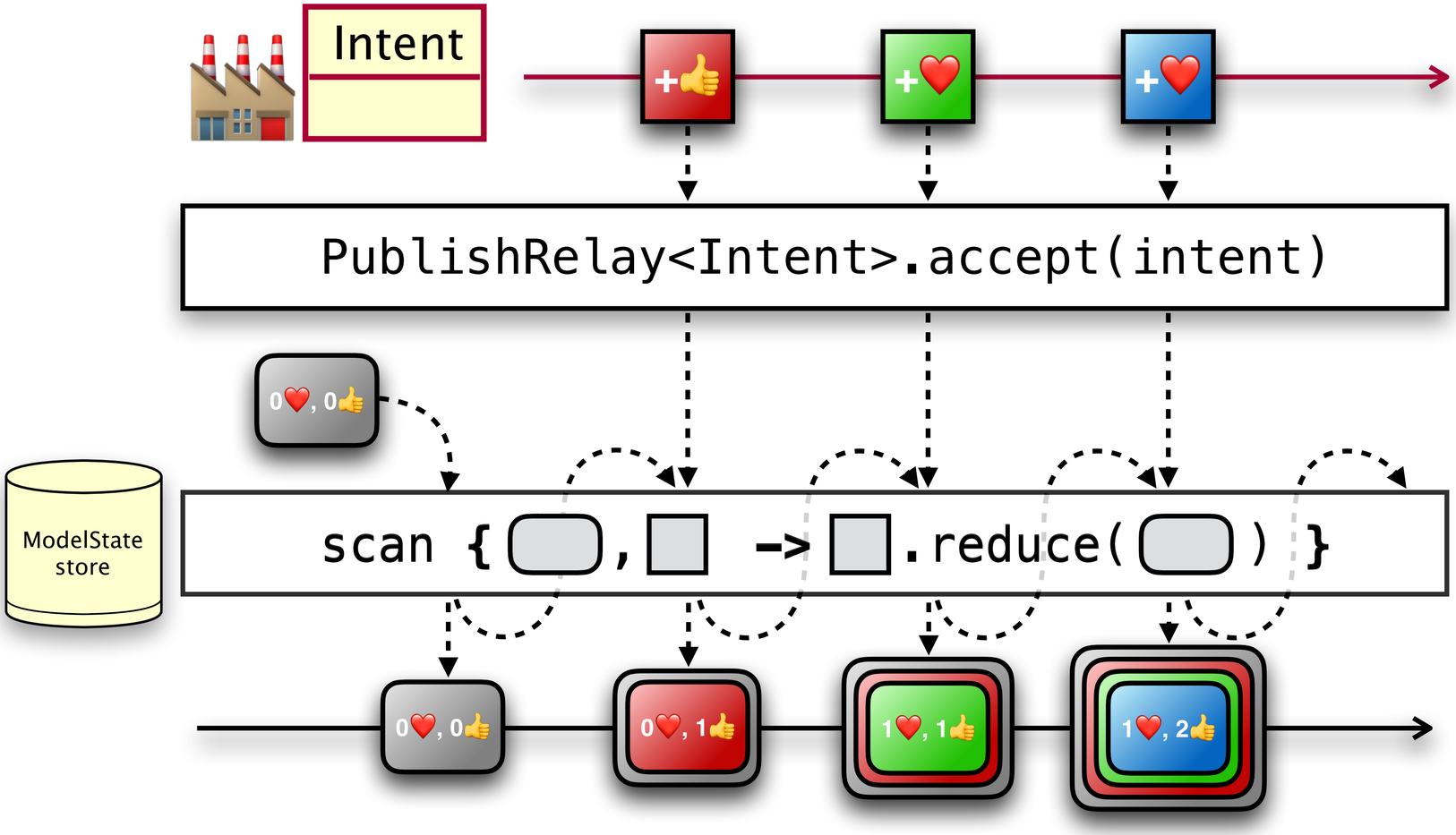


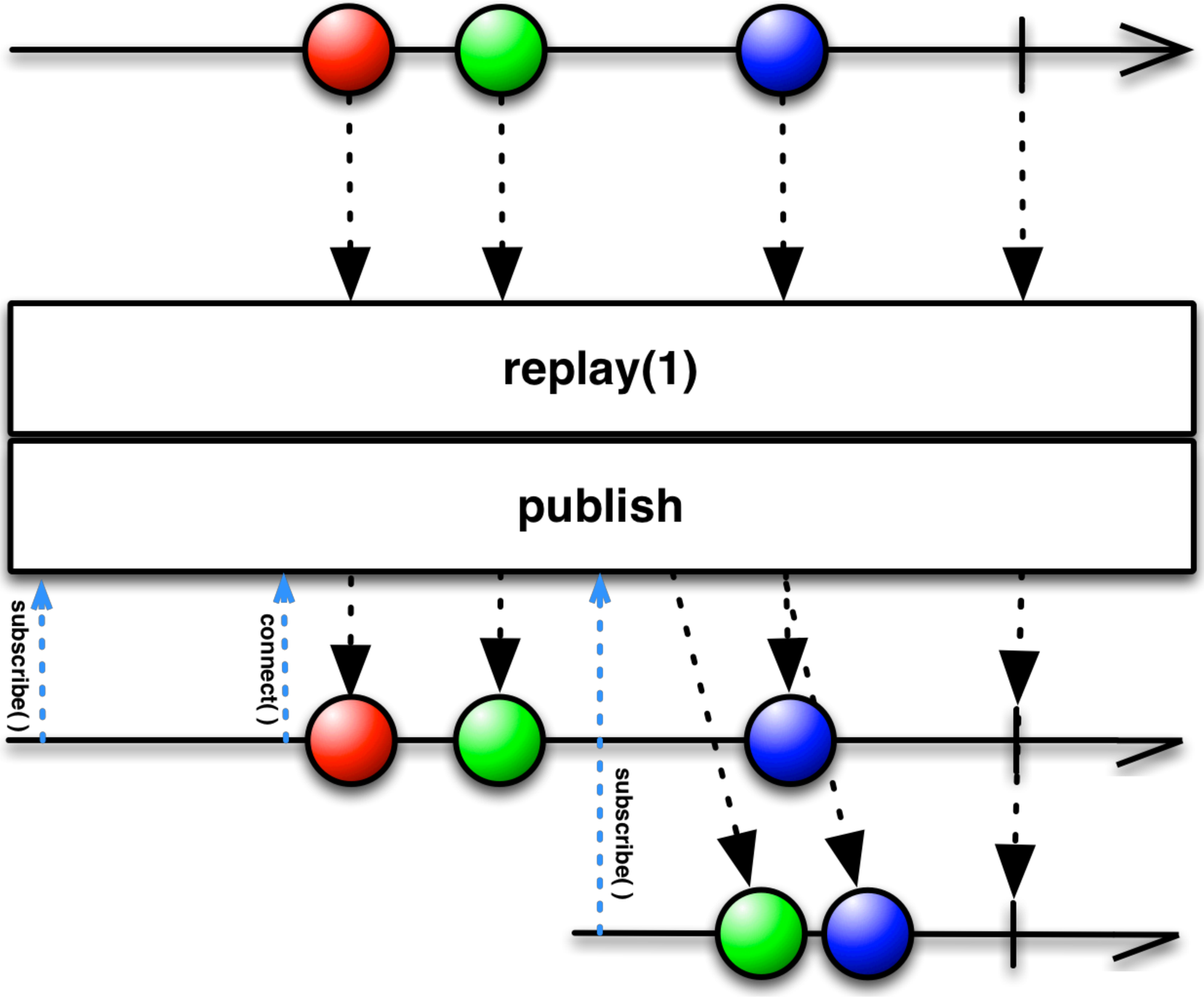

Intent
PublishRelay<Intent>.accept(intent)
ModelState
store
scan { , -> .reduce( ) }

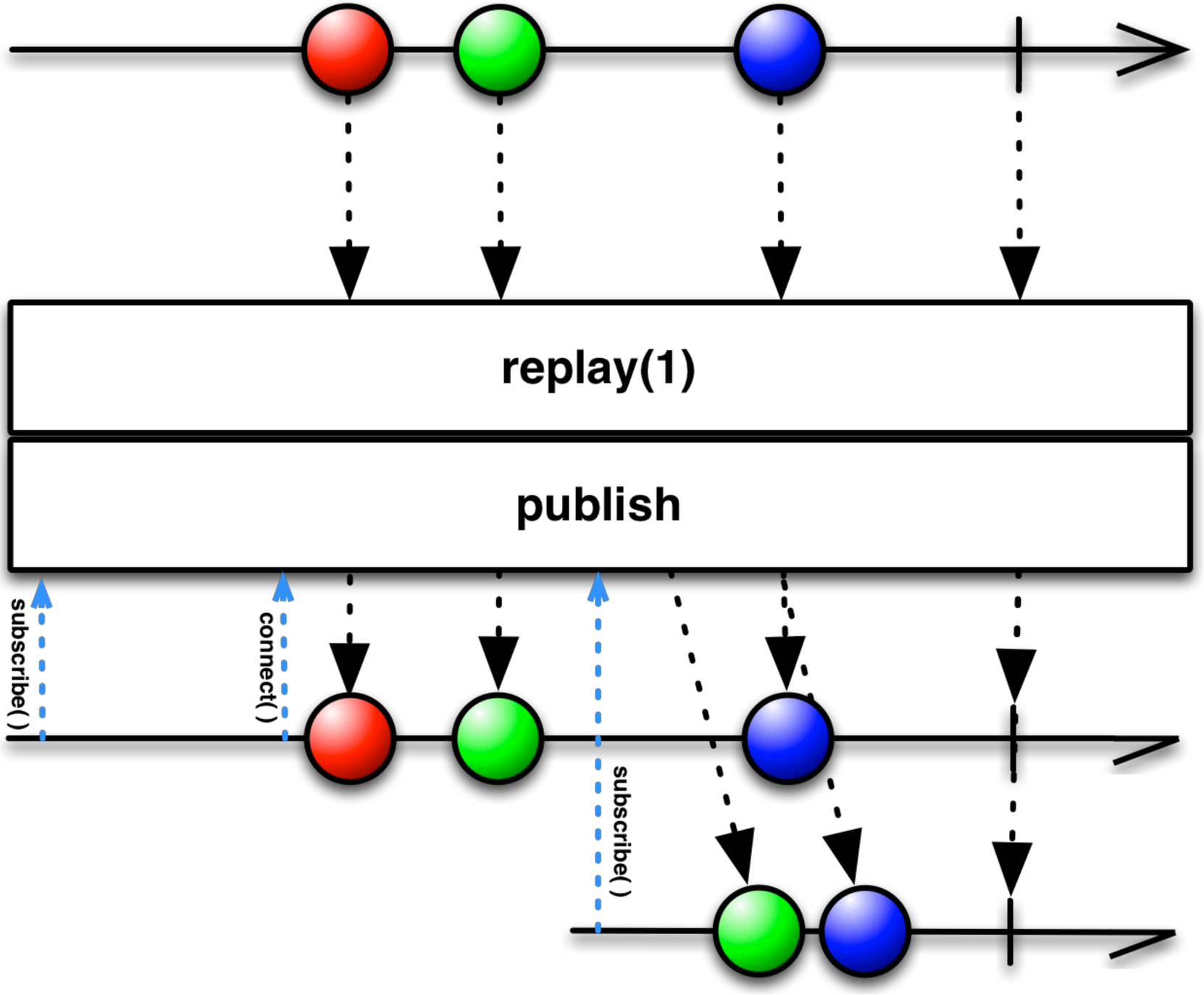

Intent
PublishRelay<Intent>.accept(intent)
0❤️, 0👍
ModelState store
scan { , -> .reduce( ) }
0❤️, 0👍

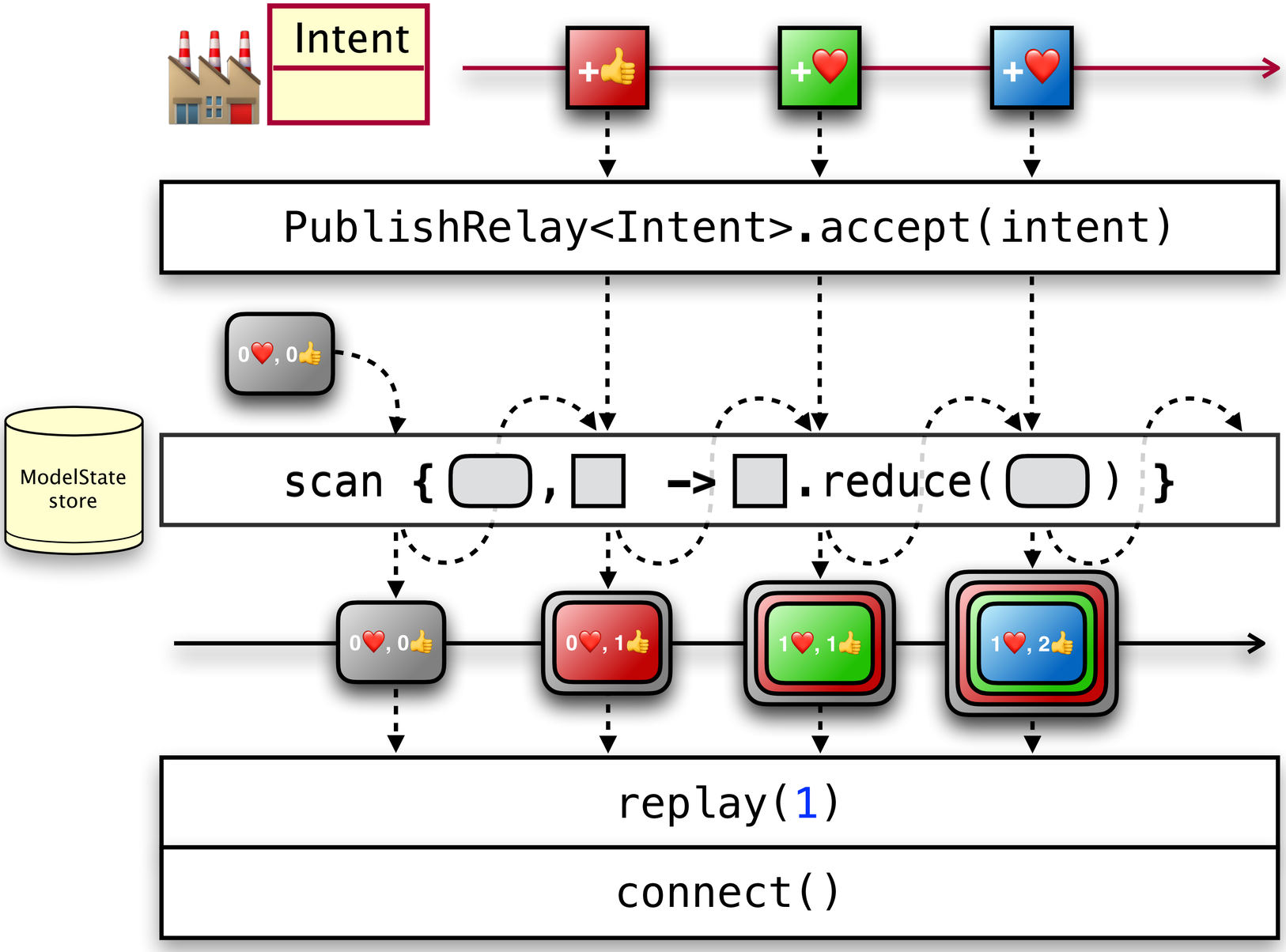

Intent
+👍
+❤️
+❤️
PublishRelay<Intent>.accept(intent)
0❤️, 0👍
ModelState store
scan { , -> .reduce( ) }
0❤️, 0👍
0❤️, 1👍

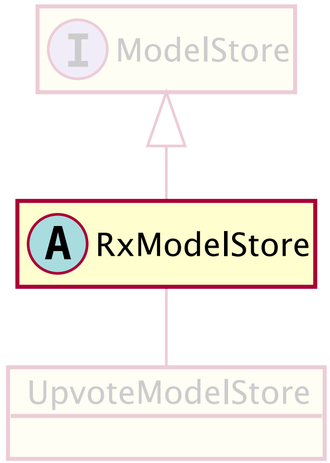

Intent
+👍
+❤️
+❤️
PublishRelay<Intent>.accept(intent)
0❤️, 0👍
ModelState store
scan { , -> .reduce( ) }
0❤️, 0👍
0❤️, 1👍
1❤️, 1👍


Intent
PublishRelay<Intent>.accept(intent)
0❤️, 0👍
ModelState store
scan { , -> .reduce( ) }
0❤️, 0👍
0❤️, 1👍
1❤️, 1👍
1❤️, 2👍

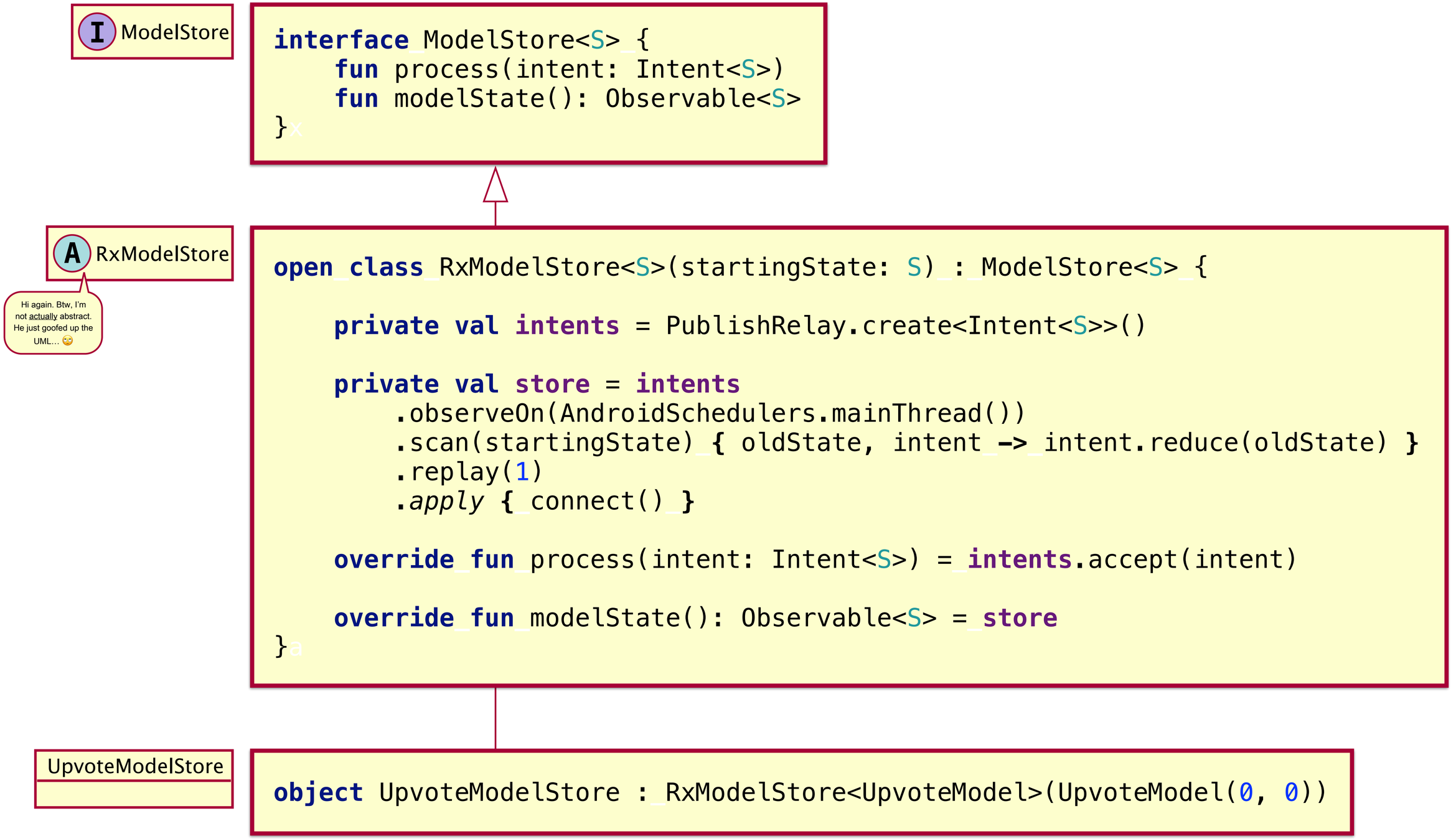

replay(1)
publish
subscribe( )
connect( )
subscribe( )

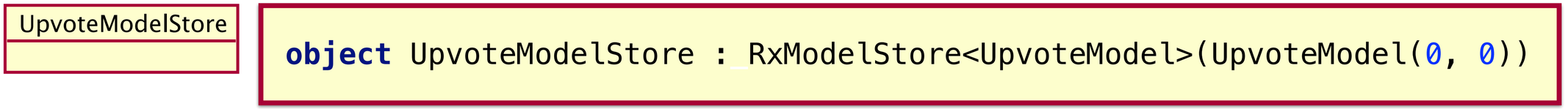

replay(1)
publish
subscribe( )
connect( )
subscribe( )


Intent
+👍
+❤️
+❤️
PublishRelay<Intent>.accept(intent)
0❤️, 0👍
ModelState store
scan { , -> .reduce( ) }
0❤️, 0👍
0❤️, 1👍
1❤️, 1👍
1❤️, 2👍
replay(1)
connect()

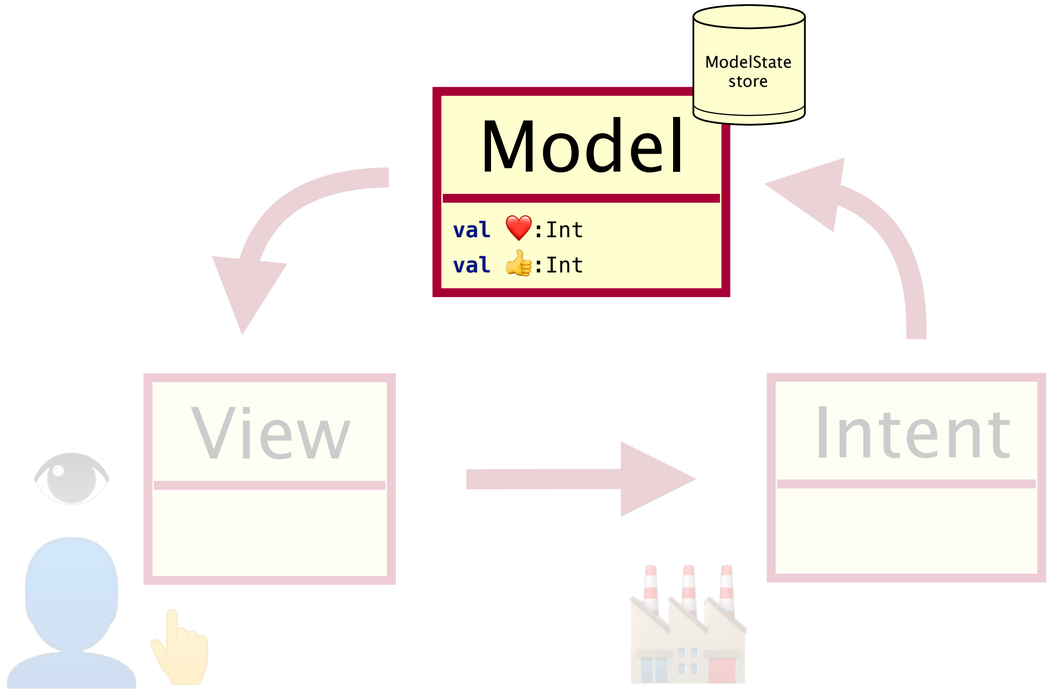



```
open class RxModelStore<S>(startingState: S) : ModelStore<S> {

    private val intents = PublishRelay.create<Intent<S>>()

    private val store = intents
        .observeOn(AndroidSchedulers.mainThread())
        .scan(startingState) { oldState, intent -> intent.reduce(oldState) }
        .replay(1)
        .apply { connect() }

    override fun process(intent: Intent<S>) = intents.accept(intent)

    override fun modelState(): Observable<S> = store
}
```

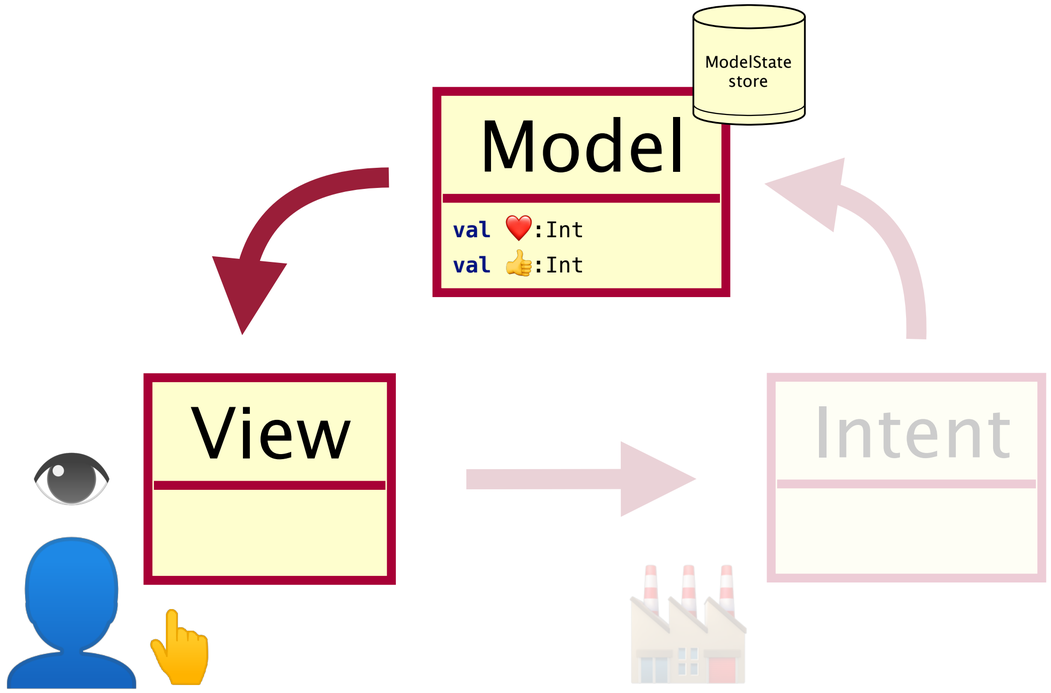

I ModelStore
interface ModelStore<S> {
fun process(intent: Intent<S>)
fun modelState(): Observable<S>
}x
A RxModelStore
Hi again. Btw, I'm not actually abstract. He just goofed up the UML… 😁
open class RxModelStore<S>(startingState: S) : ModelStore<S> {
private val intents = PublishRelay.create<Intent<S>>()
private val store = intents
.observeOn(AndroidSchedulers.mainThread())
.scan(startingState) { oldState, intent -> intent.reduce(oldState) }
.replay(1)
.apply { connect() }
override fun process(intent: Intent<S>) = intents.accept(intent)
override fun modelState(): Observable<S> = store
}a
UpvoteModelStore
object UpvoteModelStore : RxModelStore<UpvoteModel>(UpvoteModel(0, 0))


UpvoteModelStore
object UpvoteModelStore : RxModelStore<UpvoteModel>(UpvoteModel(0, 0))

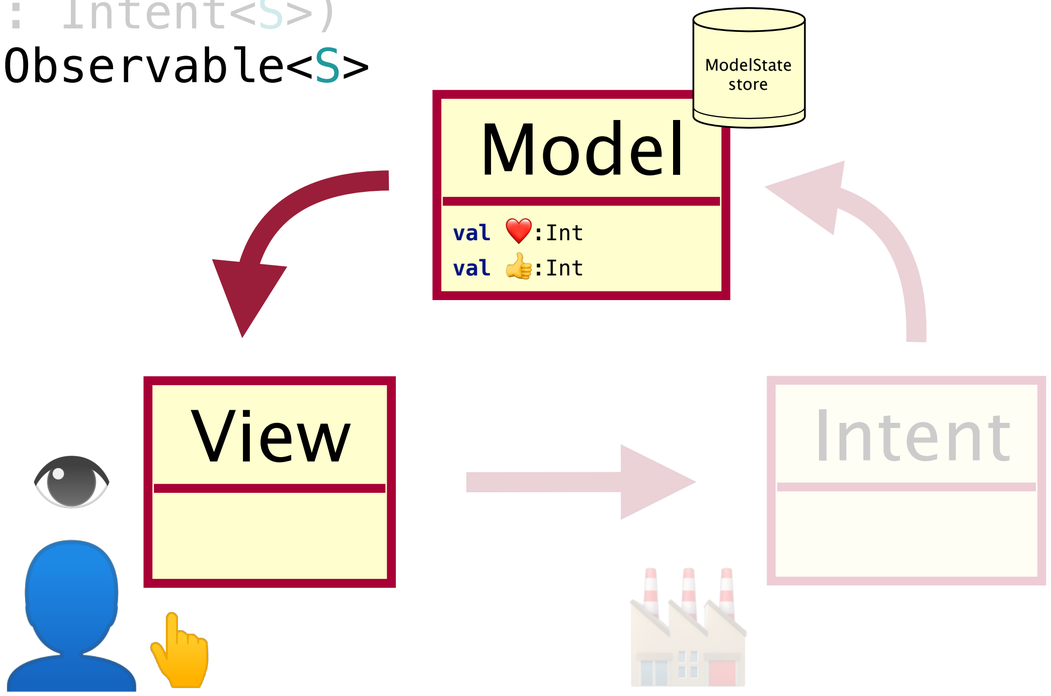
```
object UpvoteModelStore
```

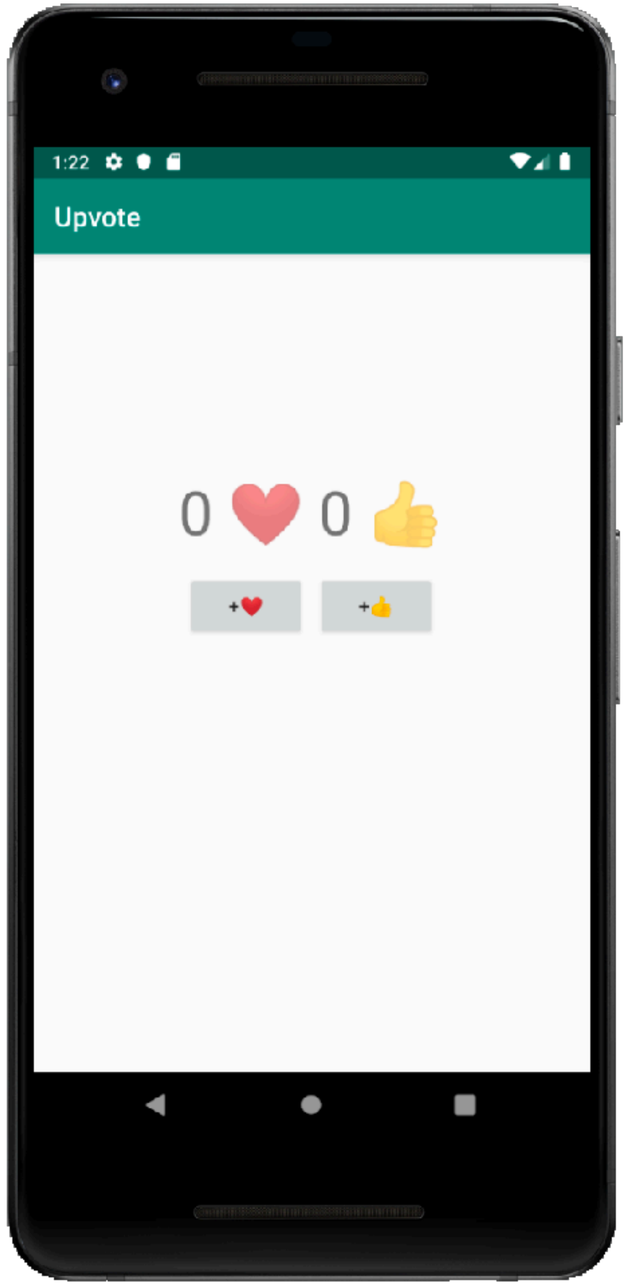


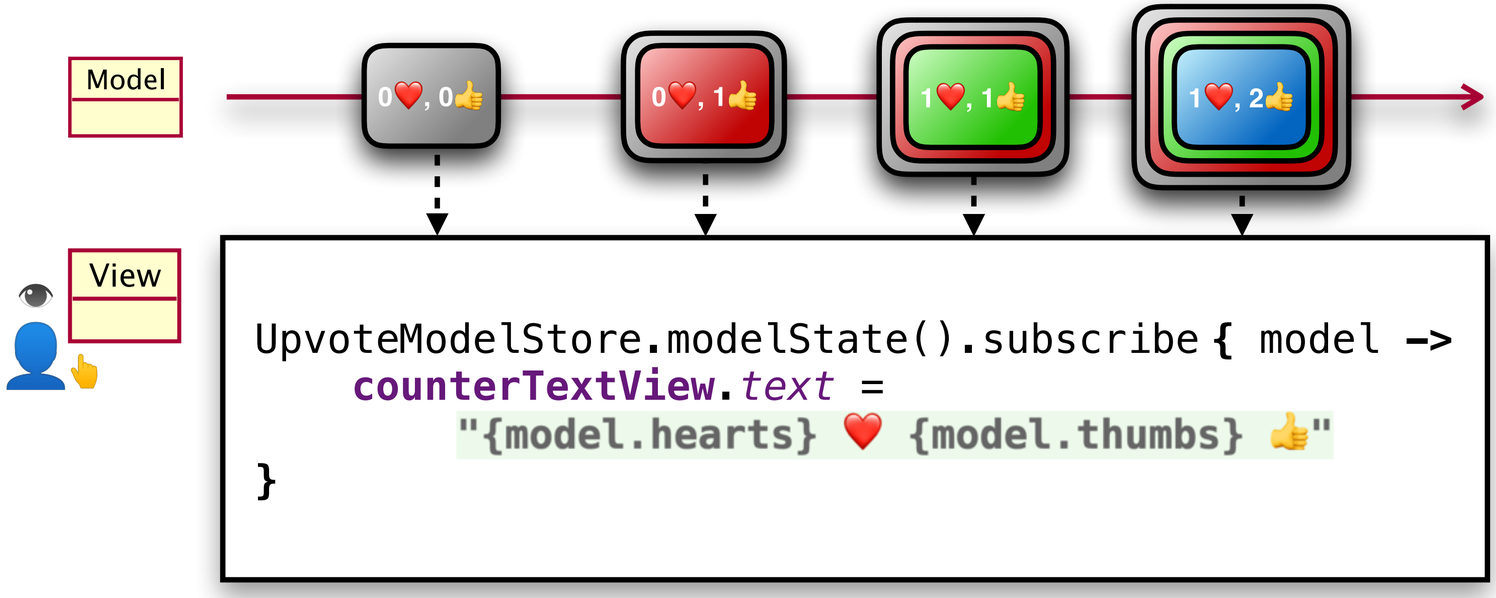
```
object UpvoteModelStore
```

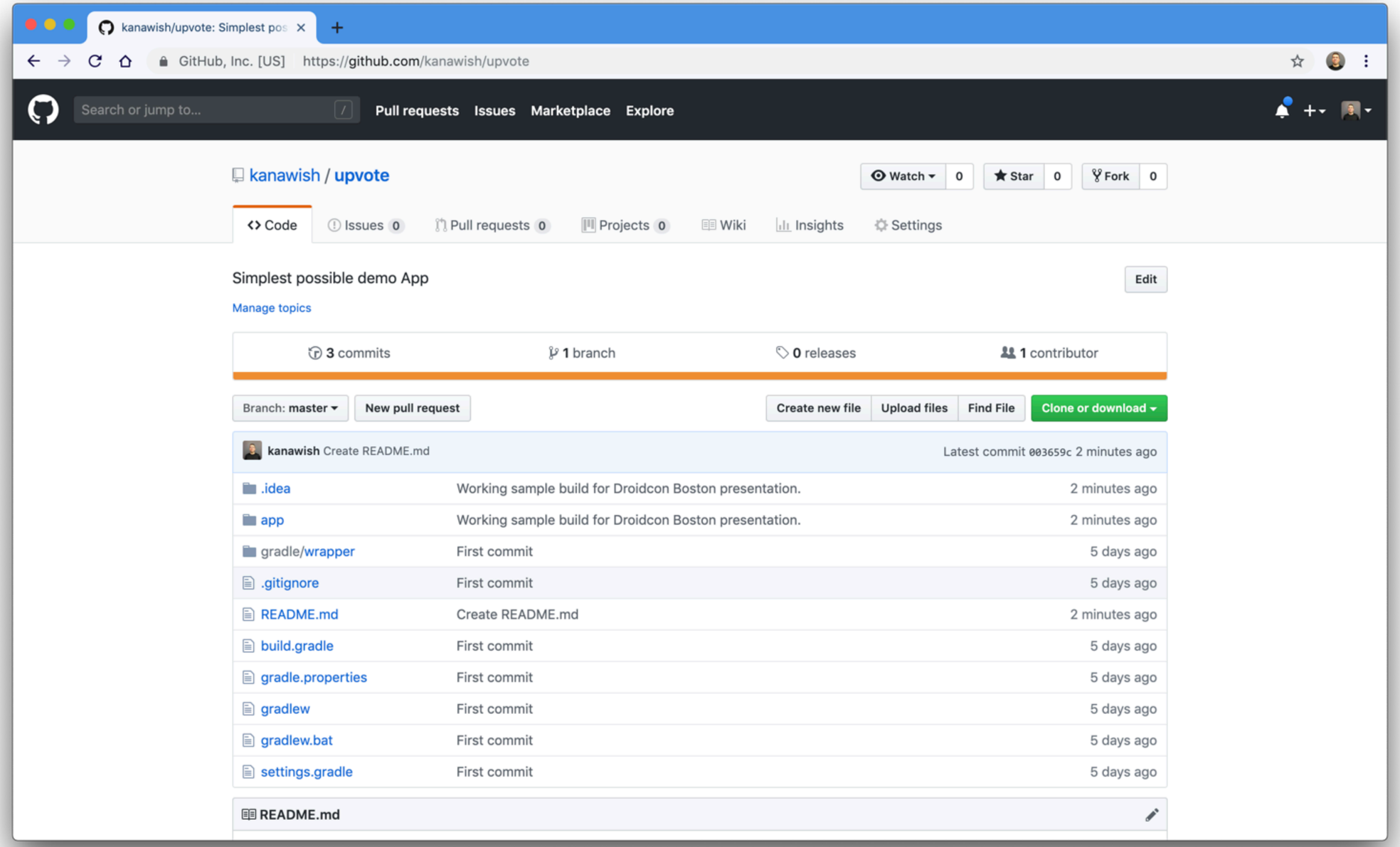


```
object UpvoteModelStore

interface ModelStore<S> {
    fun process(intent: Intent<S>)
    fun modelState(): Observable<S>
}
```




1:22
Upvote
0 ❤️ 0 👍
+❤️
+👍

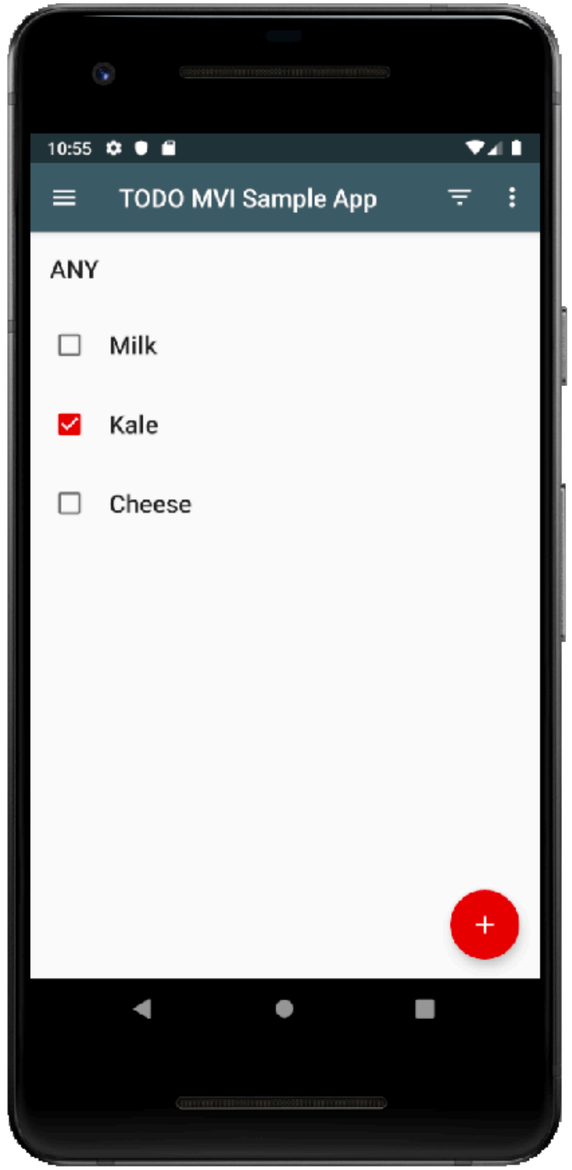

Model
0❤️, 0👍
0❤️, 1👍
1❤️, 1👍
1❤️, 2👍
View
UpvoteModelStore.modelState().subscribe { model ->
counterTextView.text =
"{model.hearts} ❤️ {model.thumbs} 👍"
}



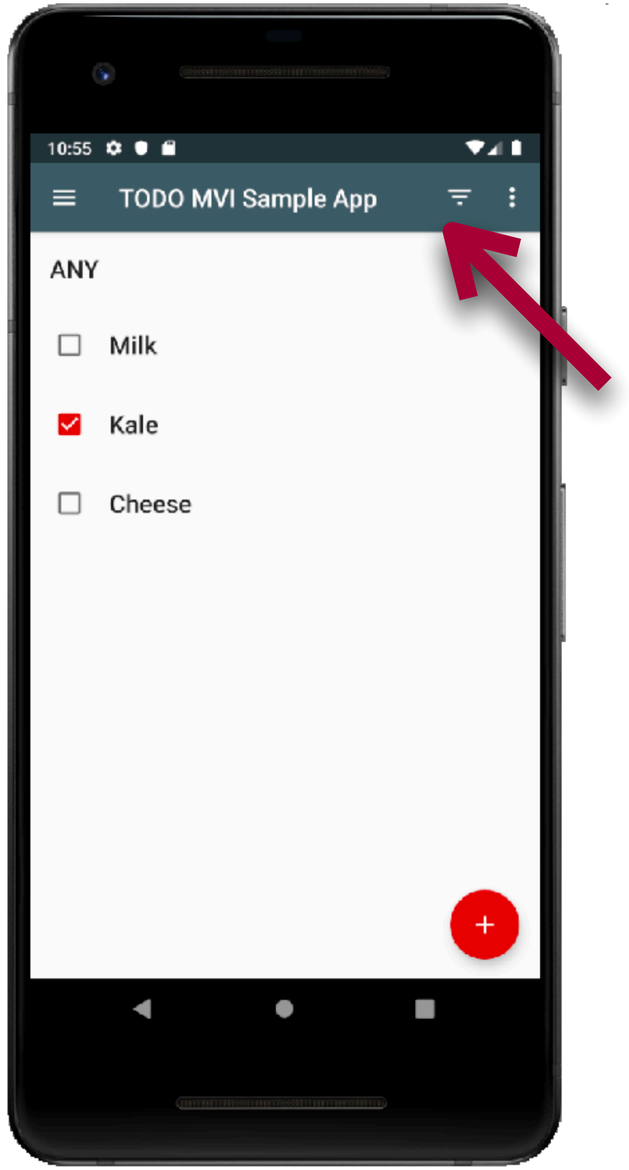
https://github.com/kanawish/upvote

Live demo

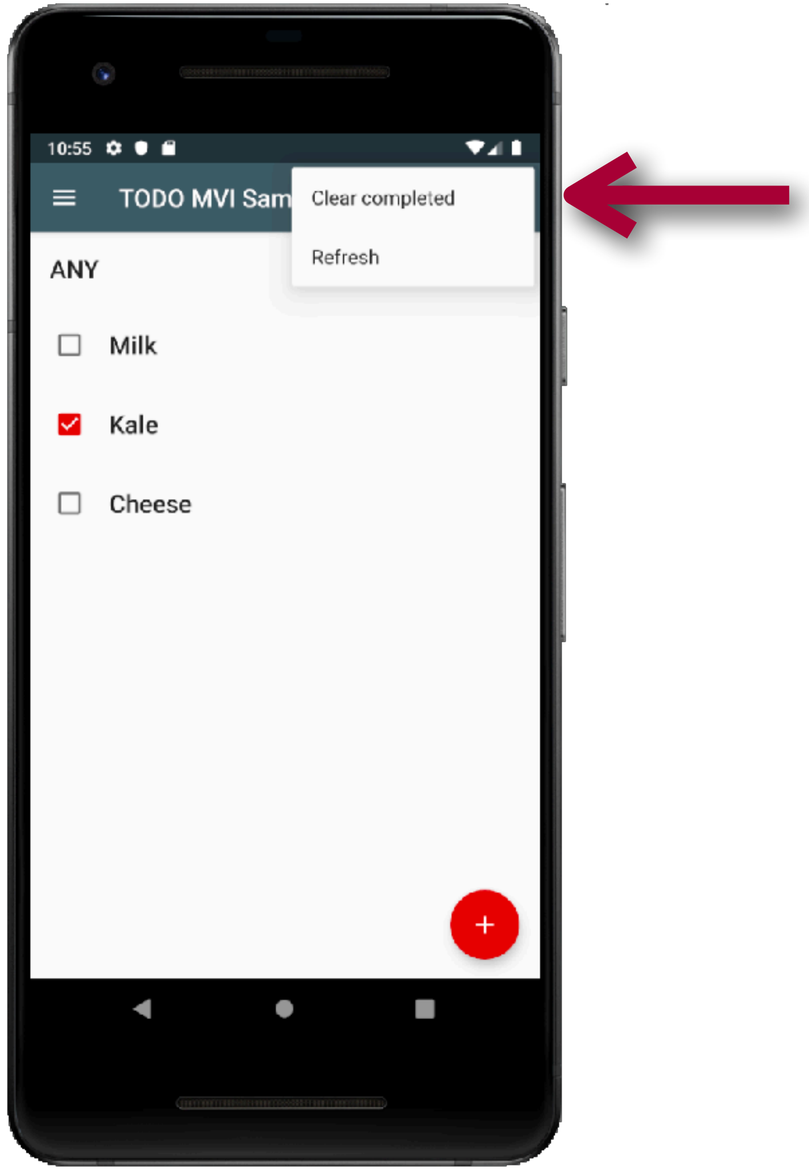
Live demo



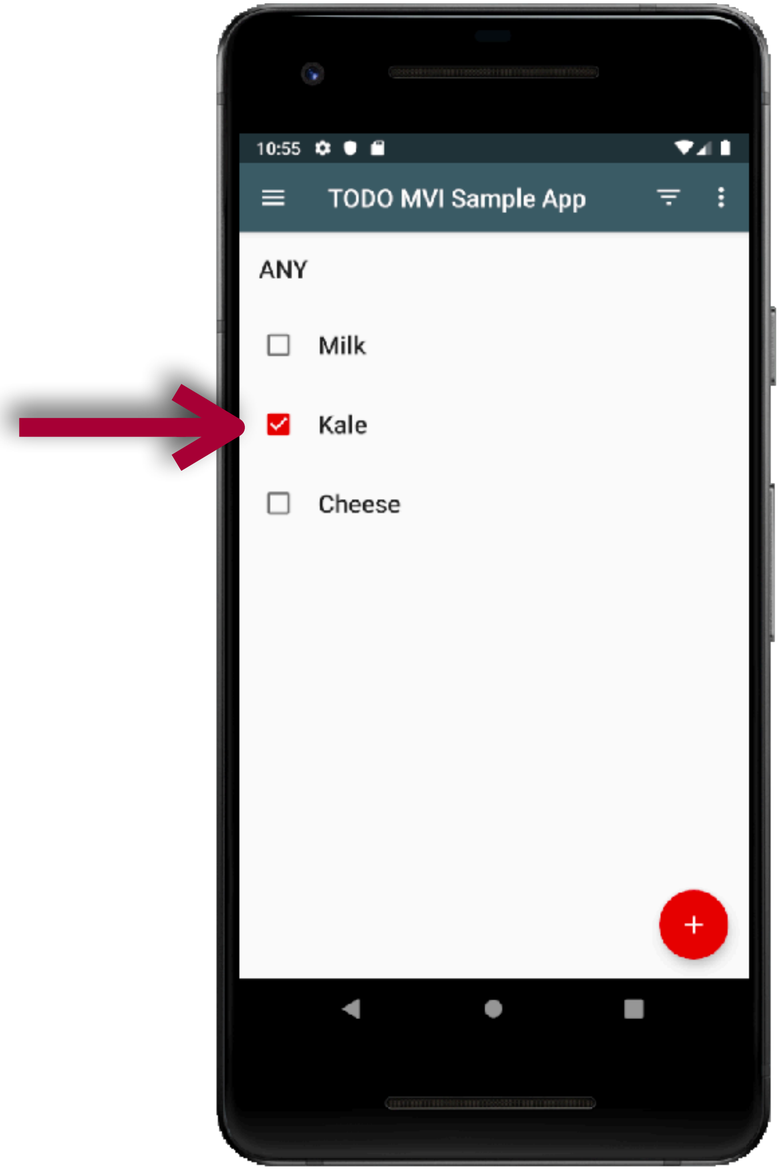
https://github.com/kanawish/android-mvi-sample



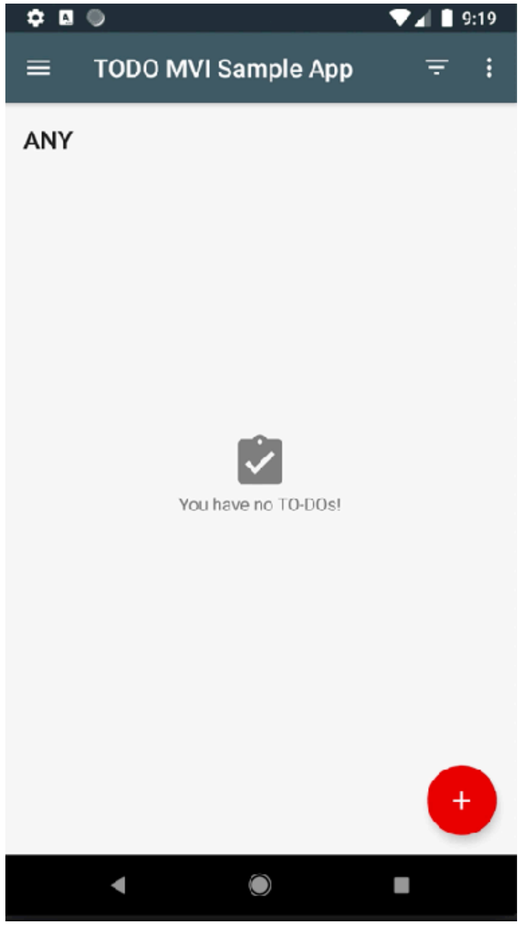
https://github.com/kanawish/android-mvi-sample



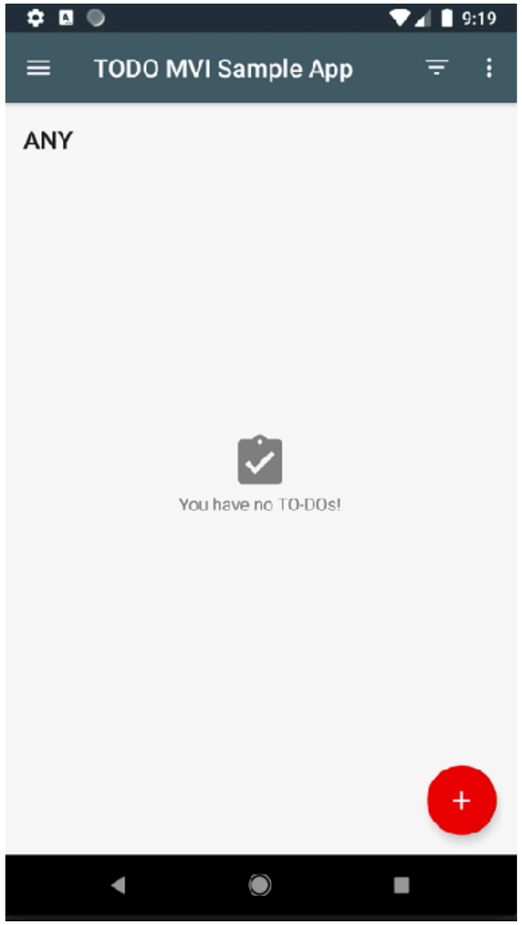
https://github.com/kanawish/android-mvi-sample



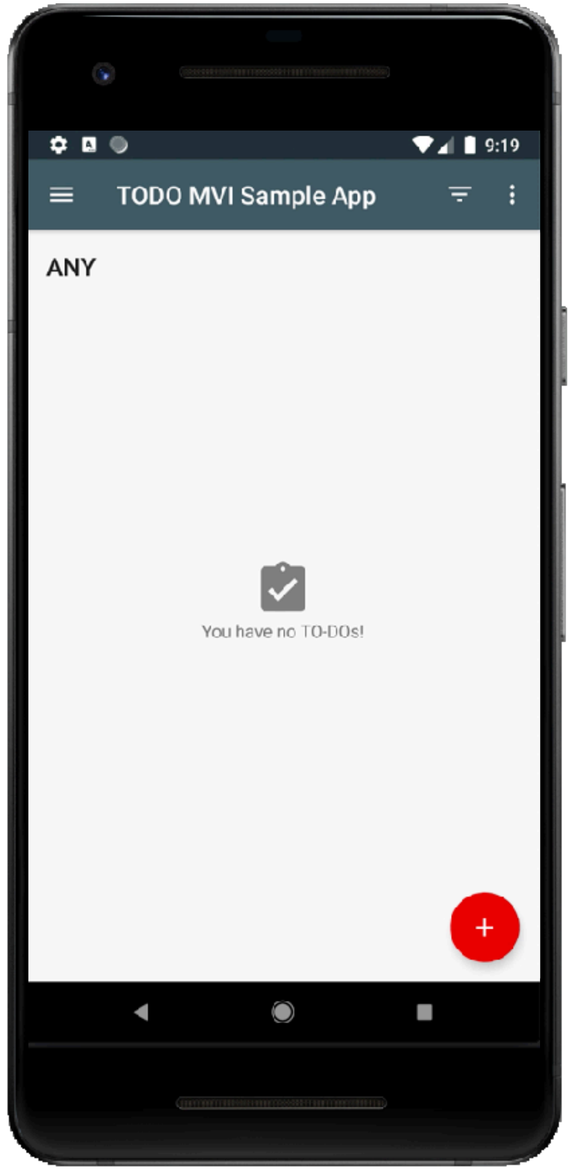
https://github.com/kanawish/android-mvi-sample



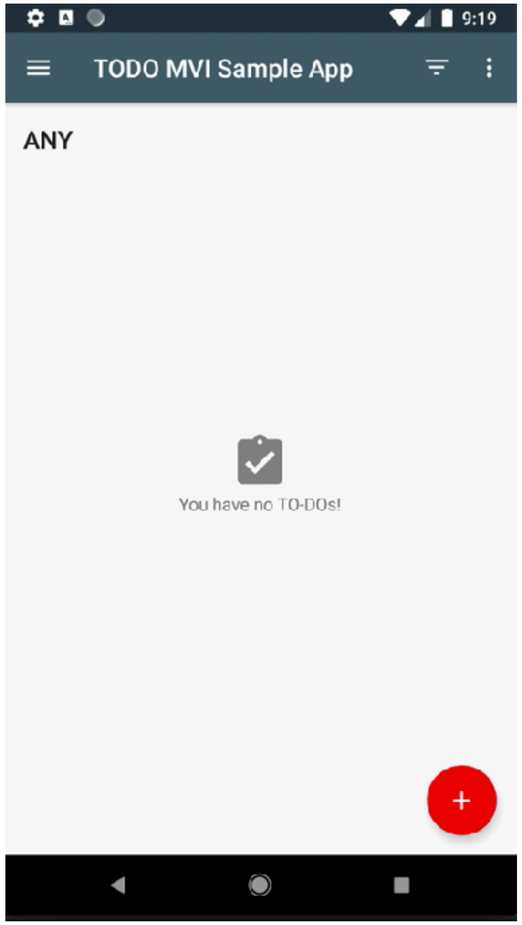
https://github.com/kanawish/android-mvi-sample



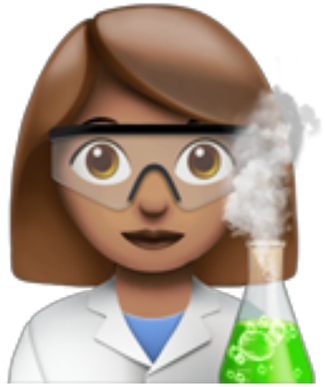
https://github.com/kanawish/android-mvi-sample

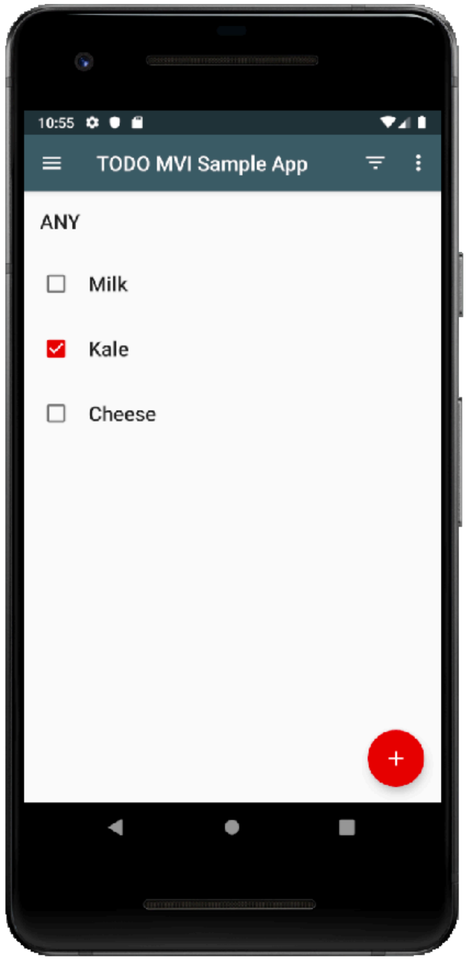


https://github.com/kanawish/android-mvi-sample

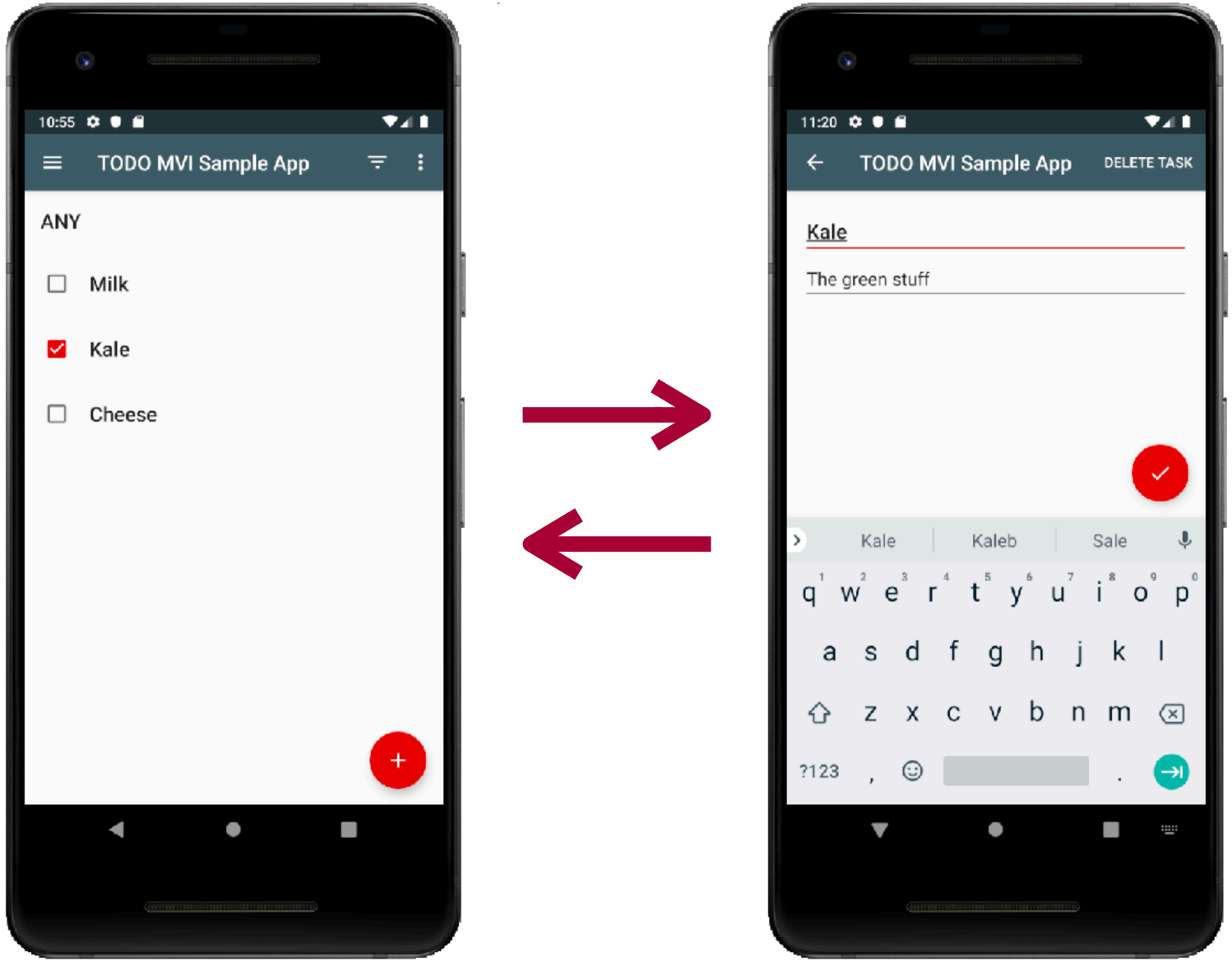


https://github.com/kanawish/android-mvi-sample

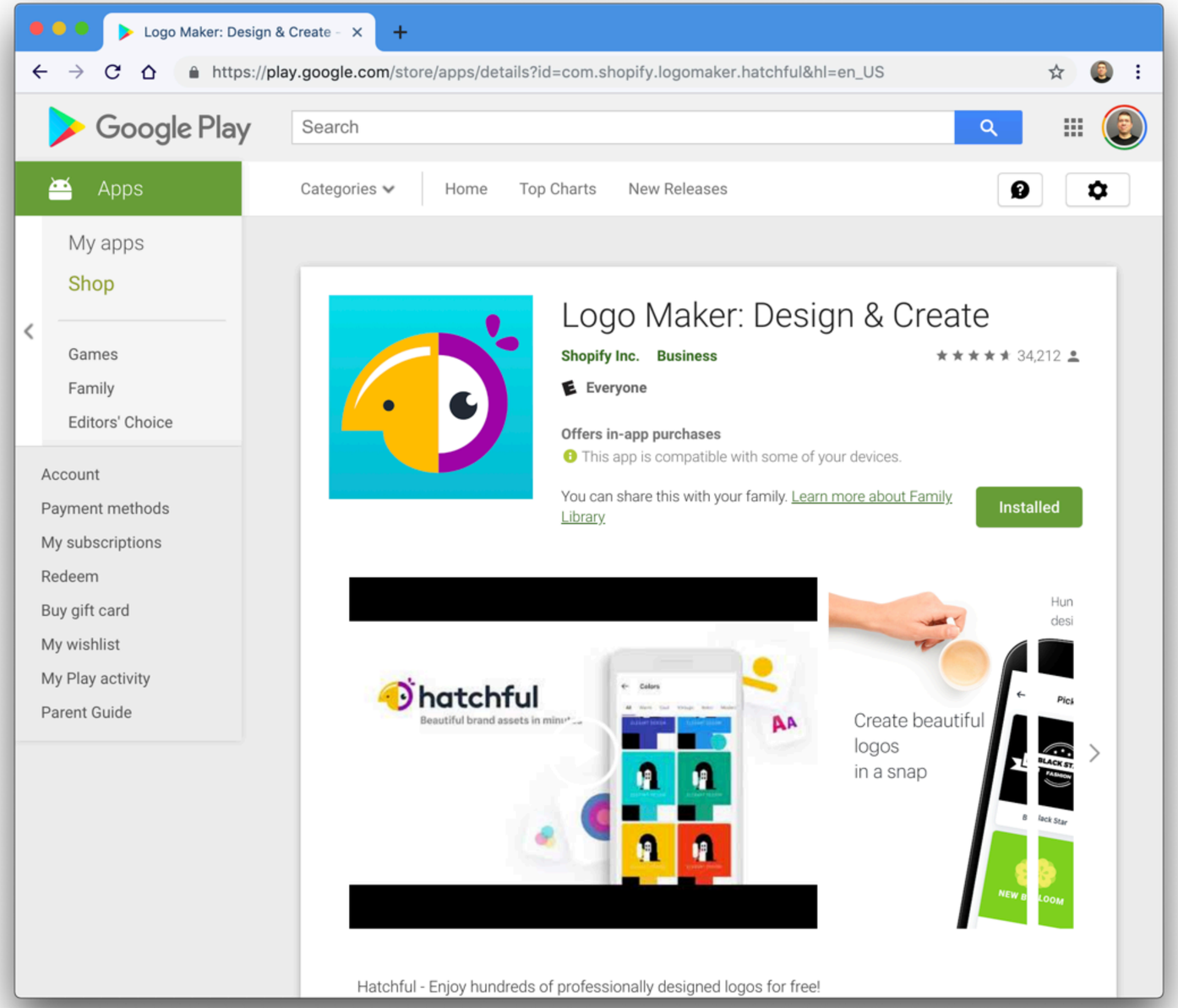




https://github.com/kanawish/android-mvi-sample



https://github.com/kanawish/android-mvi-sample



https://play.google.com/store/apps/details?id=com.shopify.logomaker.hatchful

Thank you!

https://github.com/kanawish/upvote
https://github.com/kanawish/android-mvi-sample