What you can build at the end of this semester?

A lot as long as you don't build everything yourself.

- e.g. Compose Cookbook

What skills do you need to build upon others work and modify it to fit your own ideas?

Ability to trace code and understand why it is implemented this way.

Challenge of what you have learnt in Unit.3

Outline

1. Activity
2. Fragment
3. ViewModel
4. LiveData

1. Activity

1.1. Why should we care about *activity lifecycle*?

We might need to be careful about `onCreate` (as for state initialization) and `onDestroy` (as the states will be lost), but what about others like `onStart`, `onResume`, etc?

2. Fragment

2.1. Why do we need Fragment?

Instead of using fragments for different pages, can't we just use many activities?

2.2. Fragment Lifecycle vs. ViewBiding

Q1. Why do we initiize the `binding` in `onCreateView`, but set its attribute in `onViewCreated`? Can't we just have them all in 1 method?

```
class StartFragment : Fragment() {
  ...
  override fun onCreateView(...): View? {
    val fragmentBinding = FragmentStartBinding.inflate(inflater, container, false)
    binding = fragmentBinding
    return fragmentBinding.root
  }

  override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    binding?.startFragment = this
  }
}
```

Q2. Why do we need to have `binding = null` in `onDestroyView`? Shouldn't it be automatically removed once the Fragment dies?

```
class StartFragment : Fragment() {
  private var binding: FragmentStartBinding? = null

  override fun onDestroyView() {
    super.onDestroyView()
    binding = null
  }
}
```

3. ViewModel

3.1. Why do we need ViewModel?

Instead of inheriting from `ViewModel`,

```
class OrderViewModel : ViewModel() {
  ...
}
```

can't we just have a custom class to store the view state?

```
class OrderViewState() {
  ...
}
```

3.2. What does property delegation(`by`) do? Why do we need to use it when declaring a `ViewModel` in a `Fragment`?

```
private val viewModel: OrderViewModel by OrderViewModel()
```

Can't we just use `=` instead?

```
private val viewModel: OrderViewModel = OrderViewModel()
```

4. LiveData

4.1. Why do we need LiveData?

With LiveData, we have to attach an observer in different places to notify the change:

```
// OrderViewModel
val quantity = LiveData<Int>(0)

// StartFragment
viewModel.quantity.observe(lifeCycleOwner) { value ->
  observer1 = value
  observer2 = value
}
```

Can't we just use a custom setter?

```
// OrderViewModel
public var quantity: Int = 0
  set(value) {
    observer1 = value
    observer2 = value
    field = value
  }
```

4.2. Why do we write LiveData in ViewModel like this

```
private val _quantity = MutableLiveData<Int>()
val quantity: LiveData<Int> = _quantity
```