SENG 440

Week 3: Declarative UI with Compose and Material

Ben Adams, benjamin.adams@canterbury.ac.nz

Problem with Views

- Views and ViewGroups are built with a **mix of code and XML resources**
- Widgets defined in our XML, but state change / ownership and event-handling happens at different places in code
- Recall **design patterns** in other SENG courses
- Android widgets do not do a good **separation of concerns**

Spinner example from TwoTimer

```
val listener = object: AdapterView.OnItemSelectedListener {
  override fun onNothingSelected(p0: AdapterView<*>?) {
    syncTimes()
  }

  override fun onItemSelected(p0: AdapterView<*>?, p1: View?, p2: Int, p3: Long) {
    syncTimes()
  }
}

picker1.onItemSelectedListener = listener
picker2.onItemSelectedListener = listener
```

Declarative **component**-based frameworks

- Very popular in web programming, e.g. React, Vue
- SwiftUI for iOS
- React Native, Flutter, etc. for cross-platform dev

- **Jetpack Compose** for Android
  - Complete rewrite of the Android UI toolkit
  - Unbundled from the underlying Android OS (user space library)
  - Less boilerplate than Views
  - State ownership and event handling are more clearly delineated

Compose framework

- Function components are the building blocks
- @Composable annotation

```
@Composable
fun MessageCard(name: String) {
    Text(text = "Hello $name!")
}
```

https://developer.android.com/jetpack/compose/tutorial

Compose framework

- @Composable annotation is **syntactic sugar** for the compiler that adds in Compose framework data structures for dynamic rendering
- Trailing lambda functions in Kotlin allow us to **nest Composable** functions in a **tree structure**
- Composables are Kotlin code, so can have **any logic** you want based on program state
  - E.g. `if`, `when` statements
- The Compose runtime will **re-render the app view** dynamically based on state changes

Under the hood (somewhat older video, some info e.g. about state is *out of date*): https://school.geekwall.in/p/SM6ZtY5f

Theming - Material Design

- **Design language** for App UIs on Google devices
  - Layout components
  - Colors (Light / dark themes)
  - Typography (headers, fonts, spacing)
  - Shapes (common UI elements: cards, drop-downs, buttons, etc)

- Compatible with Views and Jetpack Compose

Material Design for Android

Design systems in Compose | Jetpack Compose | Android Developers

Todo before next week

- Read through Thinking in Compose.
- Complete the Week 3 tutorial in the class notes repo.
- Give some more thought to what app you want to build for assignment 1. If you have any questions about your idea, send me a message on Slack or Email. (General questions that others might benefit from please post to Slack)
- We will cover more about handling state and binding views in Compose to the models in upcoming weeks.