Android - Activiteiten

Kotlin

- Verbeterde versie van Java
- JetBrains (2017)
- Werkt naadloos samen met Java

Waarom Kotlin?

- Compactere syntax
- Veiliger: geen null pointer exception

Wat is een activiteit (Activity)?

- Gebruikersinterface om één actie uit te voeren ~ venster

Principe

- UI beschrijven met functies
- Reeks functies die data omvormen tot UI-hiërarchie
- Data verandert -> update UI

Component

- Functie met @Composable
- Data(parameters)
- Immutable
- Kan andere Composables oproepen

Font grootte

- Density Independent Pixels (dp)
  - Abstracte eenheid
  - 150 dpi screen: 1 dp = 1 pixel
- Scalable pixels (sp)
  - Aangewezen voor fonts
  - sp = dp x (user-defined) scale

Functies in Kotlin

```
fun sum(a: Int, b: Int) : Int {
    return a + b;
}
```

- Default / named args

```
fun join(strings: Collection<String>, separator: String = " ", prefix: String =
"") : String
```

- Lambda uitdrukking
    - Altijd tussen "{" en "}"

Figuur

val image = painterResource(id = R.drawable.naam)

Variabelen

- val en var, geen ; nodig
- geen types nodig, compiler achterhaalt ze

Types

- Byte, Short, Int, Long
- Float, Double
- Char
- Boolean

Bronnen / Resources

- Stukken van de app die geen (Java/Kotlin) code bevatten

Registratie luisteraar

- Koppelen aan event component

```
var result by remember { mutableStateOf(iets) }
```

Remember: object bijhouden mutableStateOf: observable -> bij verandering volgt recompose

Delegated Properties

- Niet zelf bijhouden, delegereren naar andere getter en setter

State hoisting

- Toestand liften (naar boven verplaatsen)
- Maakt component statusloos

Testen

- Waarom?
    - Bij voorkeur geautomatiseerd
        - Beperkt manueel testen
        - Werkt bestaande code nog bij wijzigingen?
    - Schrijf testen telkens je nieuwe feature toevoegt
- Type testen
    - Lokale testen of Unittesten
        - Functies, klassen, eigenschappen
    - UI-test
        - Deel van de app testen
- @VisibleForTesting
- private indien geen test
- @Test voor UnitTest en UITest

Functies als object

- fun sum(a: Int, b: Int) = a + b
- val foo: (Int, Int) -> Int = sum

Klassen

- Geen new
- Eén primaire constructor
- Meerdere secundaire constructoren
- lateinit = later waarde krijgen (niet bij initialisatie)

Dataklassen

- Automatisch gegenereerde methodes
    - equals, hashCode, toString, copy

Gelijkheid

== of ===

==: inhoud vergelijken

===: referenties vergelijken

Lijsten

- List
    - Alleen lezen
- MutableList
    - Aanpasbaar

Lijsten in UI

- LazyColumn(): toont enkel zichtbare items -> betere performantie

Animatie bij openen

- Modifier.animateContentSize(animationSpec = spring(dampingRatio = Spring.DampingRatioMediumBouncy, stiffness = Spring.StiffnessLow)

Material Design

- Scaffold
    - Basislayout voor "material design"
        - Bovenste balk
        - Inhoud
        - Onderste balk
        - Drijvende actieknoppen
    - Material Design
        - Ontwerpsysteem Google voor "goede" UIs
- Kleur
    - val Rood = Color(hex-code)
    - private val DarkColorPalette = darkColors(primary = Rood)

Animatie

- high bounce
- no bounce
- high stiffness
- very low sitffness