Android Compose

UI Structure

Human-Computer Interaction Group

Prof. Dr. Michael Rohs
michael.rohs@hci.uni-hannover.de

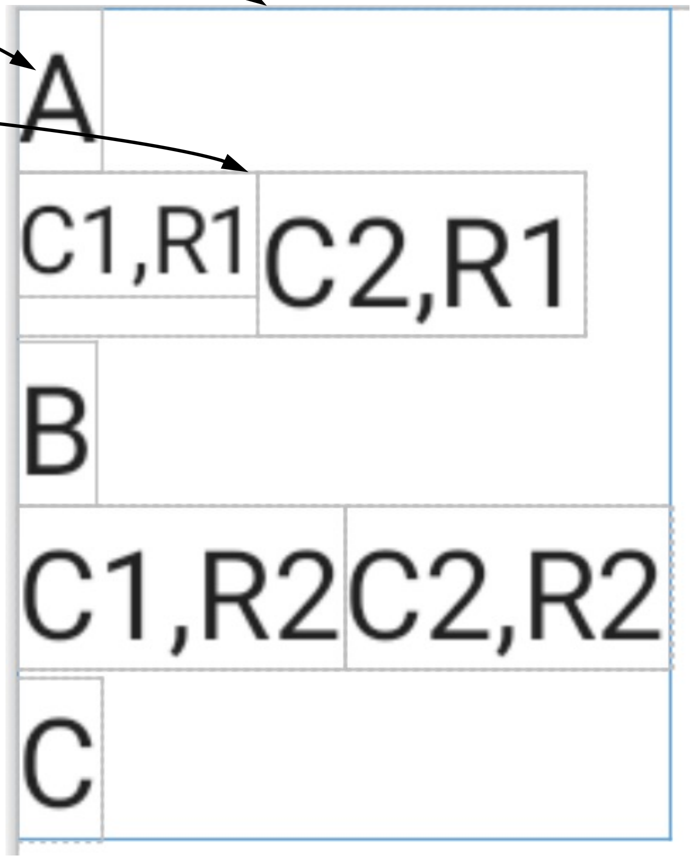
Android: UI Structure Basics

- Android Basics in Kotlin with XML
  - UI structure defined in XML
  - UI behavior defined in Kotlin
  - “old”

- Android Basics in Kotlin with Compose
  - UI structure defined in Kotlin
  - UI behavior defined in Kotlin
  - “Compose” system
  - “new”

Layout

```
Column {
    Text("A")
    Row {
        Text("C1,R1", fontSize = 12.sp)
        Text("C2,R1")
    }
    Text("B")
    Row {
        Text("C1,R2")
        Text("C2,R2")
    }
    Text("C")
}
```



Kotlin

function parameter g

```kotlin
fun f(s: String, g: () -> Unit) {
    println(s)
    g()
    println(s)
}

fun main() {
    val g = { println("abc") }
    f("---", g)
}
```

```
---
abc
---
```

Kotlin

```kotlin
fun f(s: String, g: () -> Unit) {
    println(s)
    g()
    println(s)
}

fun main() {
    f("---", { println("abc") })
}
```

```
---
abc
---
```

Kotlin

```kotlin
fun f(s: String, g: () -> Unit) {
    println(s)
    g()
    println(s)
}

fun main() {
    f("---", { println("abc") })
    f("---") { println("abc") }
}
```

```
---
abc
---
---
abc
---
```

Kotlin

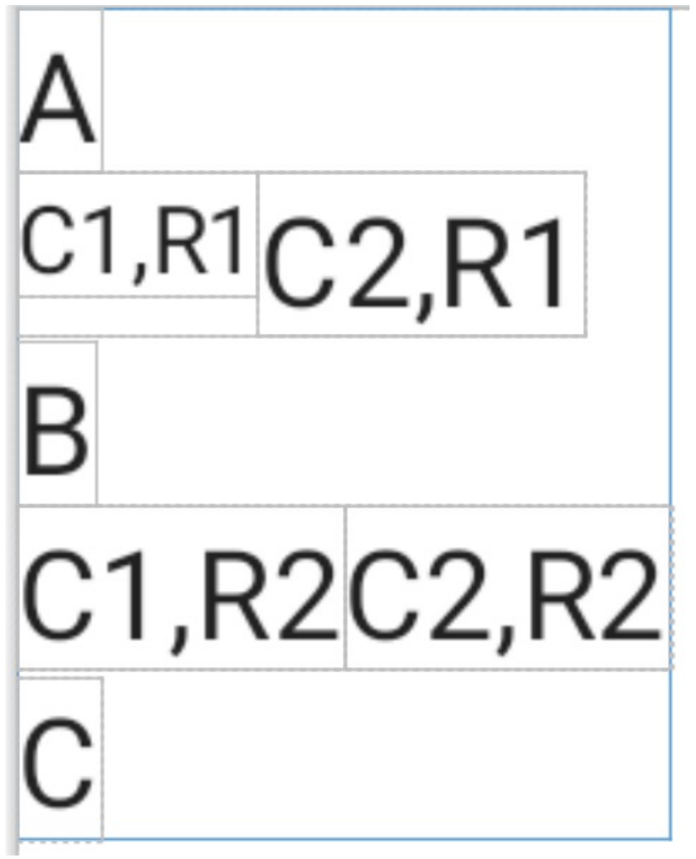
```kotlin
fun f(s: String = "---", g: () -> Unit) {
    println(s)
    g()
    println(s)
}

fun main() {
    f(g = { println("abc") })
    f () { println("abc") }
    f { println("abc") }
}
```

```
---
abc
---
---
abc
---
---
abc
---
```

Layout

Column is a function whose last parameter is a function, here called with a lambda expression {...}

```
Column {
    Text("A")
    Row {
        Text("C1,R1", fontSize = 12.sp)
        Text("C2,R1")
    }
    Text("B")
    Row {
        Text("C1,R2")
        Text("C2,R2")
    }
    Text("C")
}
```



Column

```
// A layout composable that places its children in a vertical sequence.

@Composable
inline fun Column(
    modifier: Modifier = Modifier,
    verticalArrangement: Arrangement.Vertical = Arrangement.Top,
    horizontalAlignment: Alignment.Horizontal = Alignment.Start,
    content: @Composable ColumnScope.() -> Unit // call content with ColumnScope instance
) {
    val measurePolicy = columnMeasurePolicy(verticalArrangement, horizontalAlignment)
    Layout(
        content = { ColumnScopeInstance.content() }, // ColumnScope instance available as "this" in content
        measurePolicy = measurePolicy, // layout as column
        modifier = modifier
    )
}
```

Calling a Function with an Object Instance

```
// definition of class A
class A(val s: String)
// definition of function f
fun f(obj: A, content: A.()->Unit) {
    obj.content()
}
// instance of class A
val a = A("hello")
// calling f
f(a) { println(this.s) } // output: hello
```

Layout

```
// Layout is the main core component for layout. It can be used
// to measure and position zero or more layout children.

@Composable inline fun Layout(
    content: @Composable () -> Unit,
    modifier: Modifier = Modifier,
    measurePolicy: MeasurePolicy
) {
    ...
    ReusableComposeNode<ComposeUiNode, Applier<Any>>(
        factory = ComposeUiNode.Constructor,
        update = {...},
        skippableUpdate = materializerOf(modifier),
        content = content
    )
}
```

Text

```
// High level element that displays text and provides semantics / accessibility information.
@Composable fun Text(
    text: String,
    modifier: Modifier = Modifier,
    color: Color = Color.Unspecified,
    fontSize: TextUnit = TextUnit.Unspecified,
    fontStyle: FontStyle? = null,
    fontWeight: FontWeight? = null,
    fontFamily: FontFamily? = null,
    letterSpacing: TextUnit = TextUnit.Unspecified,
    textDecoration: TextDecoration? = null,
    textAlign: TextAlign? = null,
    lineHeight: TextUnit = TextUnit.Unspecified,
    overflow: TextOverflow = TextOverflow.Clip,
    softWrap: Boolean = true,
    maxLines: Int = Int.MAX_VALUE,
    onTextLayout: (TextLayoutResult) -> Unit = {},
    style: TextStyle = LocalTextStyle.current
) {...}
```

Modifier

```
// An ordered, immutable collection of modifier elements that decorate or
// add behavior to Compose UI elements. For example, backgrounds, padding
// and click event listeners decorate or add behavior to rows, text or buttons.

interface Modifier {
    ...
}
```

Resources

- Android Basics in Kotlin with Compose
  - https://developer.android.com/courses/android-basics-compose/course
- Basic Layouts in Compose
  - https://youtu.be/kyH01Lg4G1E
- Compose Basics
  - https://www.youtube.com/playlist?list=PLWz5rJ2EKKc-CG9riunK996aI6cRhXFDC
- Compose Layouts and Modifiers
  - https://www.youtube.com/playlist?list=PLWz5rJ2EKKc94tpHND8pW8Qt8ZfT1a4cq