ADVANCED UI COMPONENTS

Advanced UI Components

Definition:
Efficiently handles long lists by loading items as needed.

Syntax:

```
LazyColumn {
items(listOfItems) { item →
// Item UI
}
}
```

Examples:

```
LazyColumn {
items(listOf("Apple", "Banana", "Cherry")) { item →
Text(text = item)
}
}
LazyColumn {
items(listOf(1, 2, 3, 4, 5)) { number →
Text(text = "Number $number")
Divider()
}
}
LazyColumn {
items(listOf("Data1", "Data2", "Data3")) { data →
CustomCard(data = data)
}
}
```

RoundedCornerShape

Definition:
Adds rounded corners to UI elements.

Syntax:

```
RoundedCornerShape(size)
```

Examples:

```
Box(modifier =
Modifier.background(Color.Blue, shape
= Rou
ndedCornerShape(10.dp))) {
Text("Hello, world!", color =
Color.White)
}
```

RoundedCornerShape

Definition:
Transforms collections based on a function.

Syntax:

```
val newCollection = oldCollection.map
{ element → /* tran
sformation */ }
```

Examples:

```
val numbers = listOf(1, 2, 3, 4, 5)
val doubled = numbers.map { it * 2 }
println(doubled) // [2, 4, 6, 8, 10]
```

Arrangement.SpaceBetween and Arrangement.SpaceEvenly

Definition:
Controls spacing between elements in a layout.

Examples:

```
Row(horizontalArrangement = Arrangement.SpaceBetween) {
// Items
}
Row(horizontalArrangement = Arrangement.SpaceEvenly) {
// Items
}
```

AlertDialog

Definition:
Pop-up for user notifications or inputs.

Syntax:

```
AlertDialog(
onDismissRequest = { /* close dialog */ },
title = { Text("Title") },
text = { Text("Message") },
buttons = {
Button(onClick = { /* action */ }) {
Text("Button Text")
}
}
)
```

Examples:

```
val showDialog = remember { mutableStateOf(true) }
if (showDialog.value) {
AlertDialog(
onDismissRequest = { showDialog.value = false },
title = { Text("Alert") },
text = { Text("This is an alert dialog.") },
buttons = {
Button(onClick = { showDialog.value = false })
{
Text("OK")
}
}
)
}
```

IconButton and Icons

Definition:
Icons for visual cues and IconButtons for clickable actions.

Syntax:

```
Icon(imageVector = Icons.Default.Home,
contentDescription
= "Icon")
IconButton(onClick = { /* action */ })
{
Icon(imageVector = Icons.Default.Home,
contentDescript
ion = "Icon Button")
}
```

Examples:

```
IconButton(onClick = { println("Icon
clicked!") }) {
Icon(imageVector = Icons.Default.Home,
contentDescript
ion = "Home Icon")
}
```

Background Modifier

Definition:
Sets background color for composables.

Syntax:

```
Modifier.background(color)
```

Examples:

```
Box(modifier =
Modifier.background(Color.Red)) {
Text("This is a red box", color =
Color.White)
}
```

Lambdas

Definition:
Concise way to define functions.

Syntax:

```
{ parameters → body }
```

Examples:

```
val sayHello = { println("Hello,
world!") }
sayHello()
val add = { a: Int, b: Int → a + b }
println(add(2, 3))
val numbers = listOf(1, 2, 3, 4, 5)
numbers.forEach { println(it) }
```

Copy Method

Definition:
Creates modified copies of data class instances.

Syntax:

```
val newObject = oldObject.copy(attribute = newValue)
```

Examples:

```
data class Person(val name: String,
val age: Int)
val person = Person("Alice", 30)
val newPerson = person.copy(age = 31)
```

Let Function

Definition:
Executes code only if the object is non-null.

Syntax:

```
nullableString?.let {
// Code block
}
```

Examples:

```
var name: String? = "Alice"
name?.let { println("Name is $it") }
```

Find Function

Definition:
Searches collections based on a condition.

Syntax:

```
val element = collection.find { it → /* condition */ }
```

Examples:

```
val numbers = listOf(1, 2, 3, 4, 5)
val found = numbers.find { it > 3 }
println(found) // 4
```

tutorials.EU

Modifier.wrapContentSize

Definition:
Adjusts size to fit content.

Syntax:

```
Modifier.wrapContentSize()
```

Example:

```
Text("Hello, world!", modifier = Modifier.wrapContentSize
())
```