**Lab Exercise 3- Create and Customize Buttons in Jetpack Compose**

**Objective:**

* Learn how to create buttons using the Button composable.
* Customize button appearance such as colors and padding.
* Handle click events using state.

**Step 1: Set Up a New Project in Android Studio**

1. **Open Android Studio**.
2. **Create a new project** by going to File -> New -> New Project.
3. Select **Empty Compose Activity** and set up your project (or enable Compose in an empty project).
4. Set the **Minimum SDK** to 21 or higher.
5. **Click Finish** to create the project.

**Step 2: Create a Simple Button**

1. Open MainActivity.kt.
2. Replace the setContent inside onCreate to display a simple button:

package com.example.buttonlab

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.\*

import androidx.compose.material3.Button

import androidx.compose.material3.Text

import androidx.compose.runtime.Composable

import androidx.compose.ui.Modifier

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import com.example.buttonlab.ui.theme.ButtonLabTheme

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContent {

ButtonLabTheme {

Surface(modifier = Modifier.fillMaxSize()) {

DisplayButton()

}

}

}

}

}

@Composable

fun DisplayButton() {

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally

) {

Button(onClick = { /\* Do something \*/ }) {

Text(text = "Click Me")

}

}

}

@Preview(showBackground = true)

@Composable

fun DefaultPreview() {

ButtonLabTheme {

DisplayButton()

}

}

**Step 3: Run the App**

1. Click the **Run** button in Android Studio or use an emulator.
2. You should see a button in the center of the screen labeled **"Click Me"**.

**Step 4: Handle Button Clicks**

Now let’s add functionality to handle button clicks using a mutableStateOf to track the number of clicks.

1. Modify the DisplayButton composable to handle click events:

import androidx.compose.runtime.\*

@Composable

fun DisplayButton() {

var clickCount by remember { mutableStateOf(0) }

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally

) {

Button(onClick = {

clickCount++ // Increment the count on each button click

}) {

Text(text = "Click Me")

}

Spacer(modifier = Modifier.height(16.dp))

Text(text = "Button clicked $clickCount times")

}

}

1. **Explanation**:
   * mutableStateOf(0): Holds the click count, which updates when the button is clicked.
   * clickCount++: Increments the count each time the button is clicked.
   * The Text below the button displays the current number of clicks.
2. Run the app again, and clicking the button should increment the click count.

**Step 5: Customize the Button**

Now, let’s customize the button’s appearance by adding colors and modifying its size.

1. Modify the Button composable to include padding and background color:

import androidx.compose.material3.ButtonDefaults

import androidx.compose.ui.graphics.Color

@Composable

fun DisplayButton() {

var clickCount by remember { mutableStateOf(0) }

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally

) {

Button(

onClick = { clickCount++ },

modifier = Modifier

.padding(8.dp)

.size(200.dp, 60.dp), // Customize button size

colors = ButtonDefaults.buttonColors(containerColor= Color.Green) // Change button color

) {

Text(text = "Click Me")

}

Spacer(modifier = Modifier.height(16.dp))

Text(text = "Button clicked $clickCount times")

}

}

1. **Explanation**:
   * **modifier.padding(8.dp)**: Adds padding around the button.
   * **size(200.dp, 60.dp)**: Sets the width and height of the button.
   * **ButtonDefaults.buttonColors**: Changes the background color of the button to green.
2. Run the app again to see the larger, green button in action.

**Step 6: Add an Icon to the Button**

Jetpack Compose allows you to add icons alongside text inside buttons.

1. Add an icon to the button by importing the required components:

import androidx.compose.material.icons.Icons

import androidx.compose.material.icons.filled.Favorite

import androidx.compose.material3.Icon

@Composable

fun DisplayButton() {

var clickCount by remember { mutableStateOf(0) }

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally

) {

Button(

onClick = { clickCount++ },

modifier = Modifier

.padding(8.dp)

.size(200.dp, 60.dp),

colors = ButtonDefaults.buttonColors(containerColor= Color.Green)

) {

Icon(Icons.Filled.Favorite, contentDescription = null) // Add an icon

Spacer(modifier = Modifier.width(8.dp)) // Space between icon and text

Text(text = "Click Me")

}

Spacer(modifier = Modifier.height(16.dp))

Text(text = "Button clicked $clickCount times")

}

}

1. **Explanation**:
   * **Icons.Filled.Favorite**: A built-in icon from the material icons library.
   * **Icon**: A composable to display an icon inside the button.
   * **Spacer**: Adds spacing between the icon and the text.
2. Run the app to see the button.