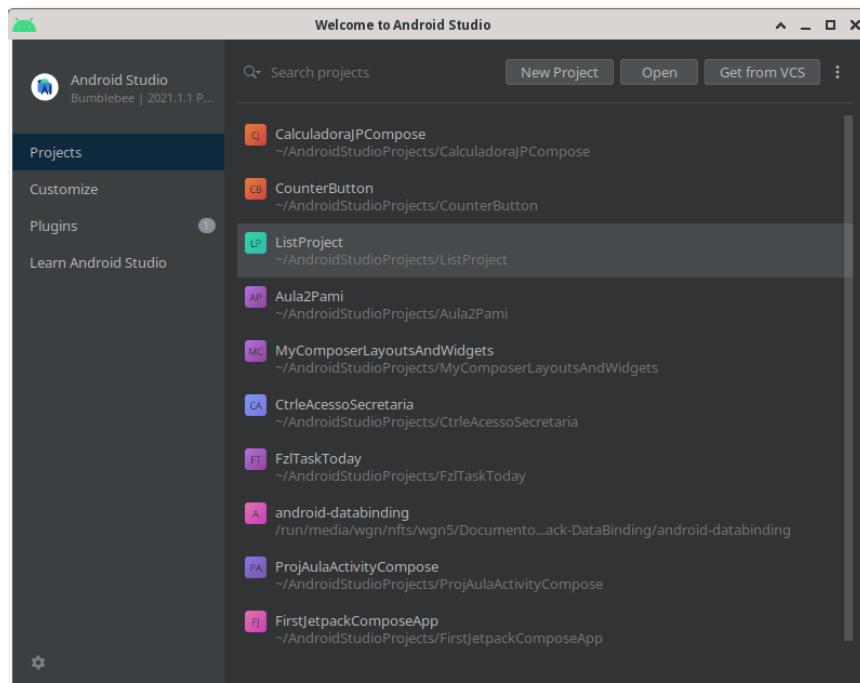


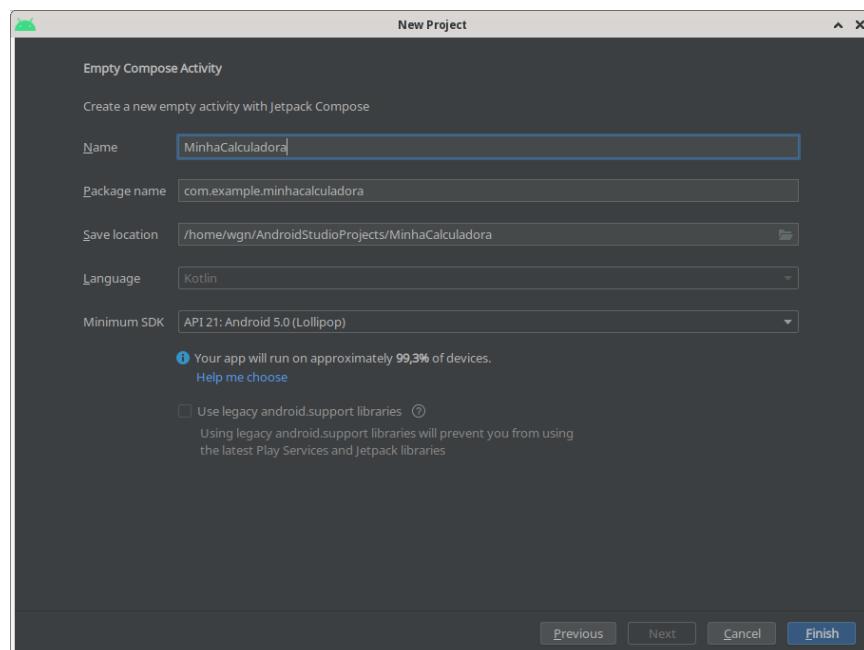
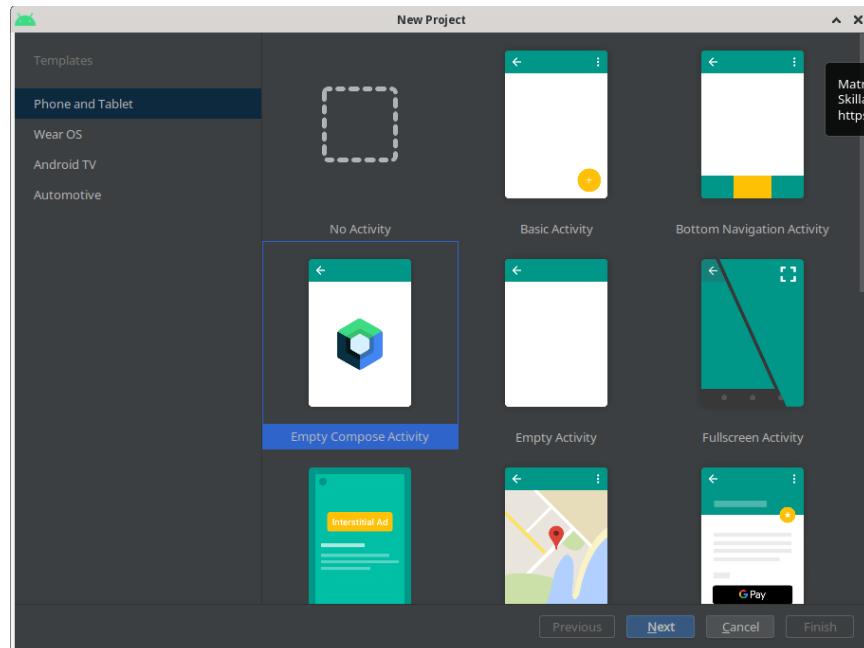
Calculadora Super Simples com Jetpack Compose

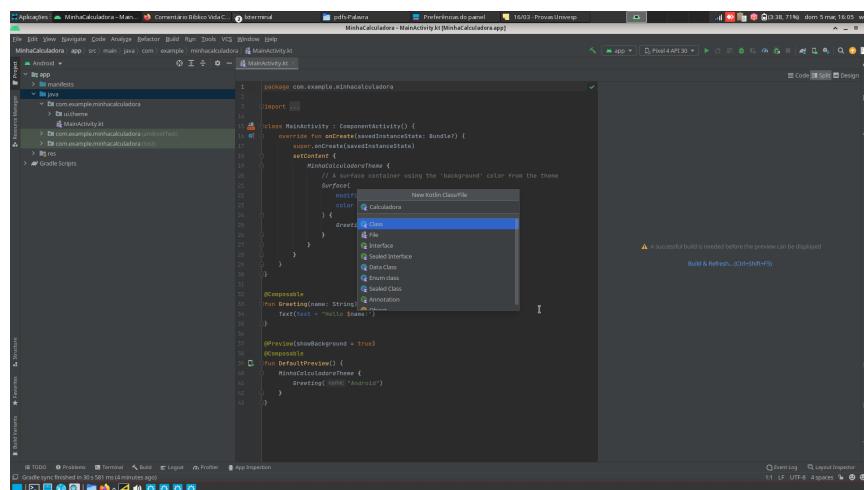
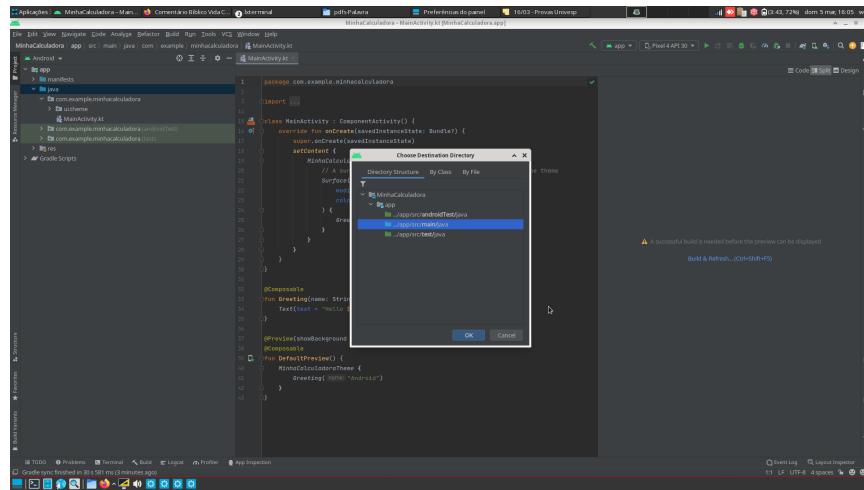
wgn

March 6, 2023

Contents







The screenshot shows the Android Studio interface with the code editor open. The project navigation bar at the top includes 'File', 'Edit', 'View', 'Navigate', 'Code', 'Analyze', 'Behaviors', 'Build', 'Run', 'Tools', 'VCS', 'Windows', and 'Help'. The title bar says 'Aplicativos - MinhaCalculadora - Calculadora - Internal - KodenAndroid - MinhaCalculadora - Main - Calculadora.kt [MinhaCalculadora.kt]'. The code editor displays the following Java code:

```
class Calculadora {  
    fun sum(a: Double, b: Double): Double = a + b  
    fun subtract(a: Double, b: Double): Double = a - b  
    fun multiply(a: Double, b: Double): Double = a * b  
    fun divide(a: Double, b: Double): Double {  
        if (b == 0.0) throw IllegalArgumentException("Divisao por zero nao pode....")  
        return a / b  
    }  
}
```

```
class Calculadora {  
  
    fun sum(a: Double, b: Double): Double = a + b  
  
    fun subtract(a: Double, b: Double): Double = a - b  
  
    fun multiply(a: Double, b: Double): Double = a * b  
  
    fun divide(a: Double, b: Double): Double {  
        if (b == 0.0) throw IllegalArgumentException("Divisao por zero nao pode....")  
        return a / b  
    }  
}
```

The screenshot shows the Android Studio interface with the code editor open. The project navigation bar at the top includes 'File', 'Edit', 'View', 'Navigate', 'Code', 'Analyze', 'Behaviors', 'Build', 'Run', 'Tools', 'VCS', 'Windows', and 'Help'. The title bar says 'Aplicativos - MinhaCalculadora - Main - CalculadoraScreen.kt [MinhaCalculadora.kt]'. The code editor displays the following Kotlin code:

```
@Composable  
fun CalculadoraScreen() {  
    var value1 by remember { mutableStateOf("") }  
    var value2 by remember { mutableStateOf("") }  
    var operator by remember { mutableStateOf("") }  
    var result by remember { mutableStateOf("") }  
  
    Column(Modifier.padding(16.dp)) {  
        TextField(  
            value = value1,  
            onValueChange = { value1 = it },  
            label = { Text("Value 1") },  
            keyboardOptions = KeyboardOptions.Default.copy(  
                keyboardType = KeyboardType.Number  
            ),  
            modifier = Modifier.fillMaxWidth()  
        )  
        TextField(  
            value = value2,  
            onValueChange = { value2 = it },  
            label = { Text("Value 2") },  
            keyboardOptions = KeyboardOptions.Default.copy(  
                keyboardType = KeyboardType.Number  
            ),  
            modifier = Modifier.fillMaxWidth()  
        )  
        //a gente vai terminar de fazer a tela da calculadora mais pra frente nesse tutorial...  
    }  
}
```

The screenshot shows two windows of the Android Studio IDE. The left window displays the Kotlin code for `MainActivity.kt`, which contains a `CalculatorView` component. The right window shows a preview of the UI, which consists of a single text input field with placeholder text and a floating label.

```

    package com.example.minhacalculadora
    import android.os.Bundle
    import androidx.activity.ComponentActivity
    import androidx.activity.compose.setContent
    import androidx.compose.foundation.layout.fillMaxSize
    import androidx.compose.material.MaterialTheme
    import androidx.compose.material.Surface
    import androidx.compose.material.Text
    import androidx.compose.runtime.Composable
    import androidx.compose.ui.tooling.preview.Preview
    import com.example.minhacalculadora.theme.MinhaCalculadoraTheme
    import com.example.minhacalculadora.ui.theme.CalculatorView

    class MainActivity : ComponentActivity() {
        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContent {
                CalculatorView()
            }
        }
    }

    @Composable
    fun CalculatorView() {
        var value1 by remember { mutableStateOf("") }
        var value2 by remember { mutableStateOf("") }
        var result by remember { mutableStateOf("") }
        var modifier = Modifier.fillMaxWidth()

        Column(modifier.padding(16.dp)) {
            TextField(
                value = value1,
                onValueChange = { value1 = it },
                label = { Text(text = "Value 1") },
                keyboardOptions = KeyboardOptions.Default.copy(
                    keyboardType = TextInputType.Number
                ),
                modifier = Modifier.fillMaxWidth()
            )
            TextField(
                value = value2,
                onValueChange = { value2 = it },
                label = { Text(text = "Value 2") },
                keyboardOptions = KeyboardOptions.Default.copy(
                    keyboardType = TextInputType.Number
                ),
                modifier = Modifier.fillMaxWidth()
            )
            // A note to return the focus to the calculator's main screen...
        }
        // Focu to the calculator screen
    }

    @Preview(showBackground = true)
    @Composable
    fun DefaultPreview() {
        CalculatorView()
    }
}

```

The right window shows the preview of the `CalculatorView` component. It features a large text input field with the placeholder "Value 1" and a floating label "Value 1". Below it is another text input field with the placeholder "Value 2" and a floating label "Value 2".

```
import androidx.compose.runtime.getValue  
import androidx.compose.runtime.setValue
```

```
private void calculateResult() {
    String result = calculate();
    if(result != null) {
        resultText.setText(result);
        resultText.setSelection(result.length());
    }
}

private String calculate() {
    String result = null;
    if(resultText.getText().length() == 0) {
        return null;
    }
    String[] tokens = resultText.getText().toString().split(" ");
    if(tokens.length > 1) {
        String operator = tokens[1];
        String value1 = tokens[0];
        String value2 = tokens[2];
        if(operator.equals("+")) {
            result = Double.parseDouble(value1) + Double.parseDouble(value2) + "";
        } else if(operator.equals("-")) {
            result = Double.parseDouble(value1) - Double.parseDouble(value2) + "";
        } else if(operator.equals("*")) {
            result = Double.parseDouble(value1) * Double.parseDouble(value2) + "";
        } else if(operator.equals("/")) {
            result = Double.parseDouble(value1) / Double.parseDouble(value2) + "";
        }
    }
    return result;
}

private void clearScreen() {
    resultText.setText("");
}
```

```
    }
}

// Fecha class MainActivty : ComponentActivity() {
//
//    override fun onCreate(savedInstanceState: Bundle?) {
//        super.onCreate(savedInstanceState)
//        setContentView(R.layout.activity_main)
//        val textInput = findViewById<EditText>(R.id.text_input)
//        val numericKeypad = findViewById<ViewGroup>(R.id.numeric_keypad)
//        val keyboard = findViewById<KeyboardView>(R.id.keyboard)
//        val lock = KeyguardManagerCompat.getLock()
//        val previewManager = getSystemService(PREVIEW_SERVICE) as PreviewManager
//        lock.setKeyguardLockEnabled(true)
//        previewManager.createViewFinderPreviewSession(textInput)
//    }
//}
```

// nota val terminal de fechar a tela da calculadora para fechar nesse tutorial...

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** Shows the project tree with packages like `minhaCalculadora`, `CalculatorActivity`, and `Calculated`.
- Code Editor:** Displays Java code for `CalculatorActivity`. The code includes imports for `java.util.ArrayList`, `java.util.List`, and `java.util.Stack`. It defines a class `CalculatorActivity` with methods for initializing the screen, setting up the keyboard, and performing calculations.
- Toolbars:** Standard Android Studio toolbars for file operations, navigation, and search.
- Bottom Bar:** Shows tabs for `TODO`, `Problems`, `Terminal`, `Build`, `Logcat`, and `App Inspection`.

The screenshot shows the Android Studio interface with the code editor open. The code is part of a MainActivity.kt file. A tooltip from the IDE's code completion feature is displayed over the `Column` modifier. The tooltip lists several options under the heading "ColumnModifier.padding(...)"

- input
- onTextChange
- onValueChange
- onKeyboardOptions
- onKeyboardType
- onFocus
- onDefaultPreview

The main code block contains a `Column` modifier followed by a `TextField` component. The `Column` modifier has a parameter `padding(16.dp)`.

```
// fecha class MainActivity : ComponentActivity() {  
//  
//    override fun onCreate(savedInstanceState: Bundle?) {  
//        super.onCreate(savedInstanceState)  
//        setContent {  
//            Column(modifier = Modifier.padding(16.dp)) {  
//                TextField(  
//                    value = value1,  
//                    onValueChange = { value1 = it },  
//                    label = { Text("Value 1") },  
//                    keyboardOptions = KeyboardOptions.Default.copy(  
//                        keyboardType = KeyboardType.Number  
//                    ),  
//                    modifier = Modifier.fillMaxWidth()  
//                )  
//  
//                TextField(  
//                    value = value2,  
//                    onValueChange = { value2 = it },  
//                    label = { Text("Value 2") },  
//                    keyboardOptions = KeyboardOptions.Default.copy(  
//                        keyboardType = KeyboardType.Number  
//                    ),  
//                    modifier = Modifier.fillMaxWidth()  
//                )  
//            }  
//        }  
//    }  
//  
//    @Composable  
//    fun calculateScreen() {  
//        var value1 by remember { mutableStateOf("") }  
//        var value2 by remember { mutableStateOf("") }  
//        var result by remember { mutableStateOf("") }  
//        var error by remember { mutableStateOf("") }  
//  
//        Column(modifier = Modifier.padding(16.dp)) {  
//            //...  
//        }  
//    }  
//}
```

This screenshot is similar to the one above, but the tooltip message "The preview is out of date" is visible at the top of the preview window. The code and the list of options for the `Column` modifier are identical to the first screenshot.

```
// fecha class MainActivity : ComponentActivity() {  
//  
//    override fun onCreate(savedInstanceState: Bundle?) {  
//        super.onCreate(savedInstanceState)  
//        setContent {  
//            Column(modifier = Modifier.padding(16.dp)) {  
//                TextField(  
//                    value = value1,  
//                    onValueChange = { value1 = it },  
//                    label = { Text("Value 1") },  
//                    keyboardOptions = KeyboardOptions.Default.copy(  
//                        keyboardType = KeyboardType.Number  
//                    ),  
//                    modifier = Modifier.fillMaxWidth()  
//                )  
//  
//                TextField(  
//                    value = value2,  
//                    onValueChange = { value2 = it },  
//                    label = { Text("Value 2") },  
//                    keyboardOptions = KeyboardOptions.Default.copy(  
//                        keyboardType = KeyboardType.Number  
//                    ),  
//                    modifier = Modifier.fillMaxWidth()  
//                )  
//            }  
//        }  
//    }  
//  
//    @Composable  
//    fun calculateScreen() {  
//        var value1 by remember { mutableStateOf("") }  
//        var value2 by remember { mutableStateOf("") }  
//        var result by remember { mutableStateOf("") }  
//        var error by remember { mutableStateOf("") }  
//  
//        Column(modifier = Modifier.padding(16.dp)) {  
//            //...  
//        }  
//    }  
//}
```

The screenshot shows the Android Studio IDE with the Java code for `MainActivity.kt`. A code completion dropdown is open at the bottom of the screen, listing various options for the variable `modifier`. The dropdown includes suggestions like `Modifier.IME_VISIBLE`, `Modifier.IME_VISIBLE_IF_NEEDED`, `Modifier.IME_VISIBLE_IF_NOT_FOCUS`, and `Modifier.IME_VISIBLE_IF_NOT_TOUCHED`. The code completion interface has tabs for "DefaultPreview" and "Design". The status bar at the bottom indicates "4244 LF UTF-8 4 spaces".

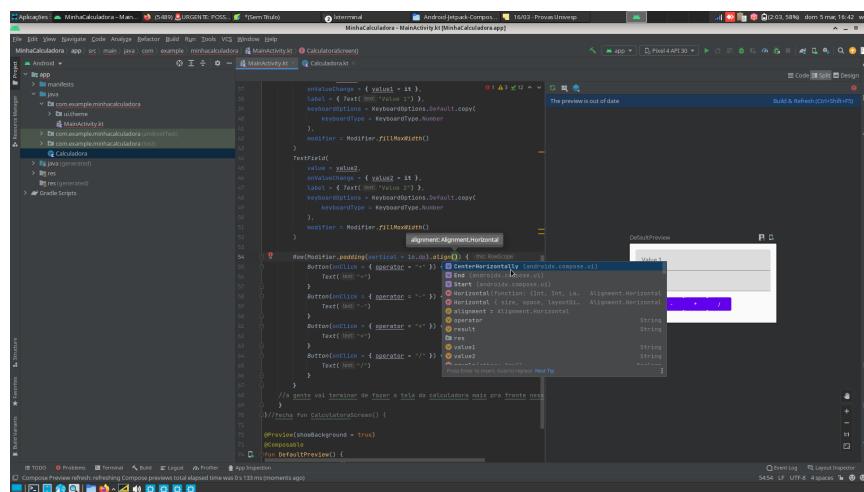
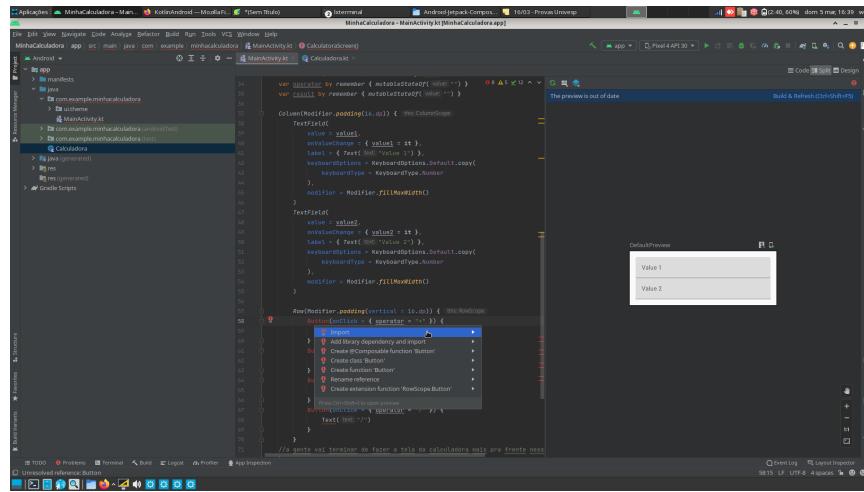
```
// fecha class MainActivity : ComponentActivity() {
    ...
    var x1 by remember { mutableStateOf<String>("") }
    var x2 by remember { mutableStateOf<String>("") }
    var y1 by remember { mutableStateOf<String>("") }
    var y2 by remember { mutableStateOf<String>("") }
    ...
    Column(modifier = Modifier.padding(16.dp)) {
        TextField(
            value = x1,
            onValueChange = { value1 = it },
            label = { Text("Value 1") },
            keyboardOptions = KeyboardOptions.Default.copy(
                keyboardType = KeyboardType.Number
            ),
            modifier = Modifier.fillMaxWidth()
        )
        ...
    }
    //sóto val terminar de fazer a tela da calculadora mais pra frente nesse tutorial...
}
//fecha fun CalculatorsScreen() {
}
@Preview(modulesBackground = true)
@Composable
fun DefaultPreview() {
}
```

This screenshot is identical to the one above, showing the same Java code for `MainActivity.kt` and the same code completion dropdown for the `modifier` variable. The dropdown again lists various `Modifier.IME_VISIBLE*` options. The Android Studio interface and status bar are also identical.

```
// fecha class MainActivity : ComponentActivity() {
    ...
    var x1 by remember { mutableStateOf<String>("") }
    var x2 by remember { mutableStateOf<String>("") }
    var y1 by remember { mutableStateOf<String>("") }
    var y2 by remember { mutableStateOf<String>("") }
    ...
    Column(modifier = Modifier.padding(16.dp)) {
        TextField(
            value = x1,
            onValueChange = { value1 = it },
            label = { Text("Value 1") },
            keyboardOptions = KeyboardOptions.Default.copy(
                keyboardType = KeyboardType.Number
            ),
            modifier = Modifier.fillMaxWidth()
        )
        ...
    }
    //sóto val terminar de fazer a tela da calculadora mais pra frente nesse tutorial...
}
//fecha fun CalculatorsScreen() {
}
@Preview(modulesBackground = true)
@Composable
fun DefaultPreview() {
}
```

```
class CalculatorScreen : Component() {
    fun onCreate(savedInstanceState: Bundle?) {
        ColumnModifier.Pending {
            TextEditor {
                value = "Value1"
                onValueChange = { value = it }
                label = { Text("Value1") }
                keyboardOptions = KeyboardOptions.Default.copy(
                    keyboardType = KeyboardType.Number
                )
                modifier = Modifier.fillMaxWidth()
            }
        }
    }
}
```

```
class CalculatorScreen : Component() {
    fun onCreate(savedInstanceState: Bundle?) {
        ColumnModifier.Pending {
            TextEditor {
                value = "Value1"
                onValueChange = { value = it }
                label = { Text("Value1") }
                keyboardOptions = KeyboardOptions.Default.copy(
                    keyboardType = KeyboardType.Number
                )
                modifier = Modifier.fillMaxWidth()
            }
        }
    }
}
```



The screenshot shows the Java code for the `CalculatorActivity` class. The code handles button click events for operators (+, -, *, /) and the equals button (=). It uses a `RowModifier` to manage the layout of the calculator buttons. The code also includes logic for clearing the screen and handling keyboard input.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calculator);
        ButterKnife.bind(this);
        calculate();
    }

    private void calculate() {
        String value1 = findViewById(R.id.value1).getEditText().getText().toString();
        String value2 = findViewById(R.id.value2).getEditText().getText().toString();
        String operator = findViewById(R.id.operator).getEditText().getText().toString();
        double result = 0;
        if (!operator.equals("") & !value1.equals("") & !value2.equals("")) {
            if (operator.equals("+")) {
                result = Double.parseDouble(value1) + Double.parseDouble(value2);
            } else if (operator.equals("-")) {
                result = Double.parseDouble(value1) - Double.parseDouble(value2);
            } else if (operator.equals("*")) {
                result = Double.parseDouble(value1) * Double.parseDouble(value2);
            } else if (operator.equals("/")) {
                result = Double.parseDouble(value1) / Double.parseDouble(value2);
            }
            findViewById(R.id.result).getEditText().setText(String.valueOf(result));
        }
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.clear:
                calculate();
                break;
            case R.id.value1:
                calculate();
                break;
            case R.id.value2:
                calculate();
                break;
            case R.id.operator:
                calculate();
                break;
            case R.id.equals:
                calculate();
                break;
            case R.id.backspace:
                calculate();
                break;
            case R.id.percent:
                calculate();
                break;
            case R.id.divide:
                calculate();
                break;
            case R.id.multiply:
                calculate();
                break;
            case R.id.subtract:
                calculate();
                break;
            case R.id.add:
                calculate();
                break;
            case R.id.point:
                calculate();
                break;
        }
    }

    @Override
    public void onEditorAction(TextView v, int actionId, KeyEvent event) {
        if (actionId == EditorInfo.IME_ACTION_EQUIVALENT) {
            calculate();
        }
    }

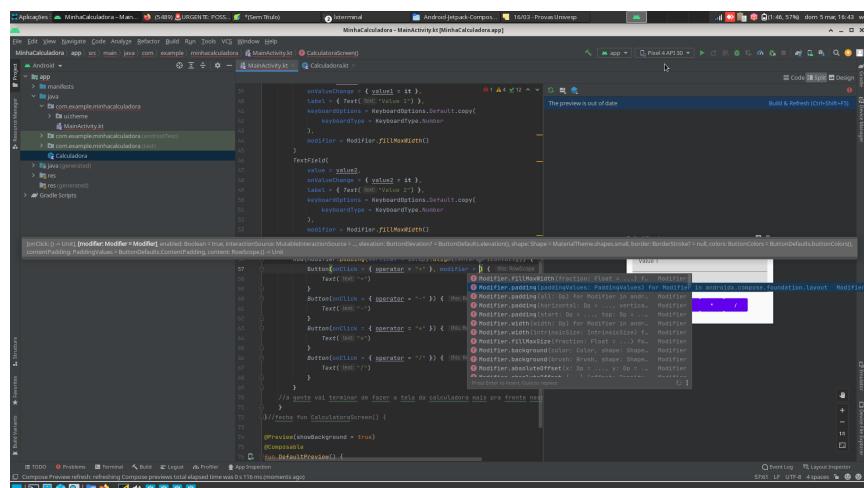
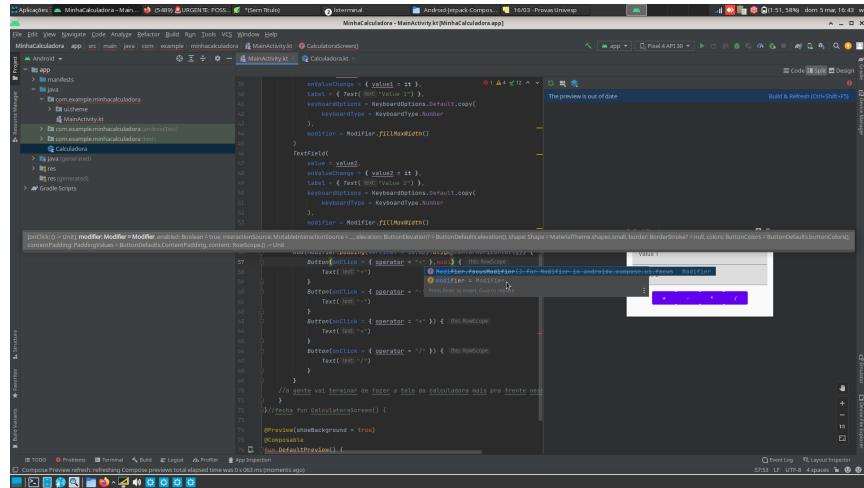
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        calculate();
    }

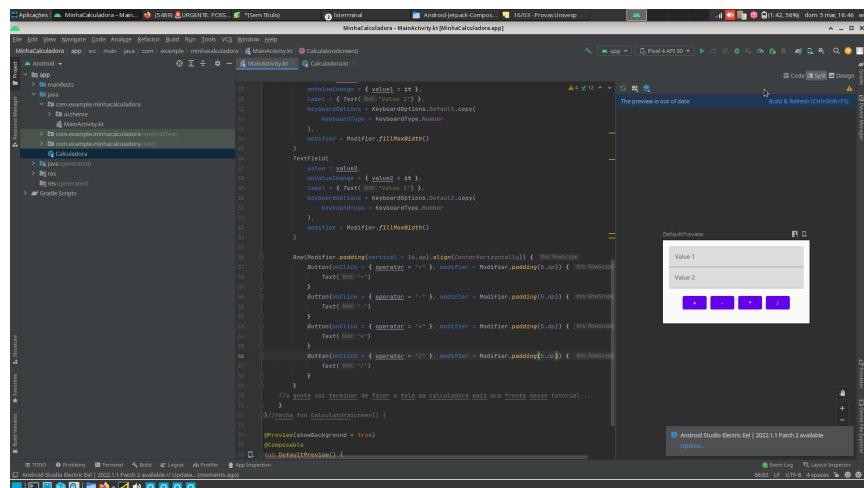
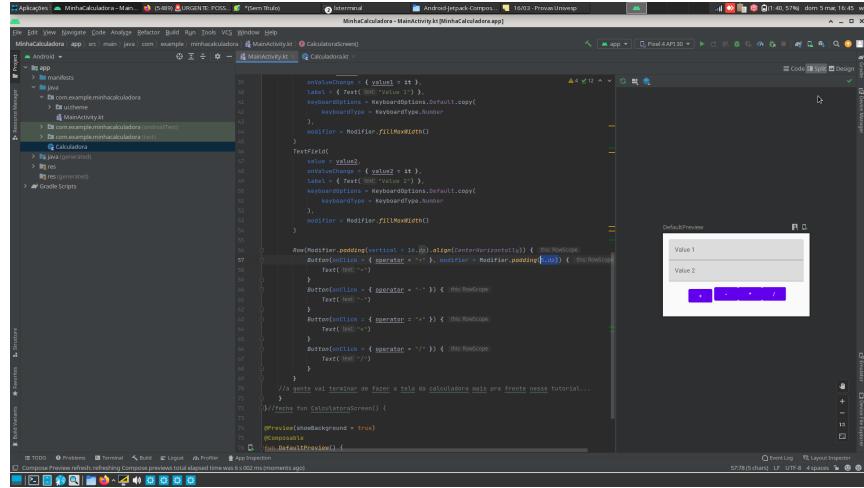
    @Override
    public void onWindowFocusChanged(boolean hasFocus) {
        super.onWindowFocusChanged(hasFocus);
        calculate();
    }

    @Override
    public void onPointerCaptureChanged(boolean hasCapture) {
        super.onPointerCaptureChanged(hasCapture);
        calculate();
    }
}
```

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "MinhaCalculadora" and contains an "app" module.
- Code Editor:** The file "Calculadora.java" is open, showing Java code for a calculator. The code includes methods for performing arithmetic operations like addition, subtraction, multiplication, division, and percentage calculations. It also handles button click events for operators and the decimal point.
- Run Tab:** The "Run" tab is selected, showing the package name "br.com.example.minhacalculadora" and the target "Pixel API 21".
- Preview Window:** A floating window titled "DefaultPreview" displays the calculator's user interface with two input fields labeled "Value 1" and "Value 2", and a result field below them.





The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "MinhaCalculadora" and contains an "app" module.
- Code Editor:** The file "MainActivity.java" is open, showing Java code for a calculator. The code includes imports for Context, View, KeyEvent, Modifier, and KeyboardLayout. It defines a class "CalculatorView" with methods for handling key events and modifying text fields. The code uses annotations like @Override, @StringRes, and @LayoutRes.
- Preview Window:** A floating "Default Preview" window is displayed, showing a UI mockup with two text input fields labeled "Value 1" and "Value 2", and a row of five purple square buttons.
- Bottom Bar:** The "Launch succeeded" message is visible at the bottom left, along with icons for Run, Stop, and Build.
- Bottom Status Bar:** The status bar shows "Android Studio Electric Eel | 2022.1.1 Patch 2 available" and "Update..."

