Vidzemes University of Applied Sciences

**Faculty of Engineering**

mobile programming engineering 1

group G-03-2025

practical assignment #4

Valmiera, 2025

CONTENT

[1 Ievads 2](#_Toc210746283)

[2 “Calculator” lietotne 3](#_Toc210746284)

[Kods 7](#_Toc210746285)

[Pielikums 12](#_Toc210746286)

|  |  |  |  |
| --- | --- | --- | --- |
| Kontakti un autors | | | |
| Vārds, Uzvārds | Grupa | Pozīcija | Kontakti (e-mail) |
| Linda Brante | Grupa #3 | Koordinators | linda.brante@va.lv |
| Arturs Siliņš | Grupa #3 | Izstrādātājs | arturs.silins@va.lv |
| Jānis Petrovs | Grupa #3 | Izstrādātājs | janis.petrovs@va.lv |
|  |  |  |  |
|  |  |  |  |

# Ievads

Mērķis: Izveidot Android lietotni, kas darbojas kā kalkulators ar pamata aritmētiskajām operācijām (+, −, ×, ÷), atmiņas funkcijām (MS, MR, MC) un iespēju redzēt pilnu aprēķina izteiksmi (piem., 5 + 5 =) kopā ar rezultātu. Lietotne tika izstrādāta, izmantojot Kotlin, lai nodrošinātu mūsdienīgu, vienkārši uzturamu un efektīvu risinājumu**.**

.

# “Calculator” lietotne

Šajā praktiskajā darbā bija nepieciešams izveidot lietotni, kurā ir viena aktivitāte. Aktivitātē atrodas divu rindu displejs un pogu režģis. Augšējā displeja rinda parāda pilnu ievadīto izteiksmi (piem., 5 + 5 =), bet apakšējā – tikai aprēķina rezultātu (piem., 10).

.

Aktivitātē ir četras atmiņas pogas:

* **MS** – saglabā pašreizējo skaitli atmiņā un izmet paziņojumu “Saglabāts: <skaitlis>”.
* **MR** – atsauc saglabāto vērtību un pievieno to ievadei.
* **MC** – notīra atmiņu un izmet paziņojumu “Atmiņa notīrīta”.
* **C** – notīra visu ievadi, rezultātu un atgriež sākotnējo stāvokli.

Pogu režģī ir cipari 0–9, punkts ., operatori +, −, ×, ÷ un vienādības poga =.

* Ievadot skaitļus, tie tiek pievienoti pašreizējam operandam.
* Nospiežot operatoru, tiek saglabāts starprezultāts un sagatavots nākamais operands.
* Poga = izpilda galīgo aprēķinu, pievieno = izteiksmei un parāda rezultātu apakšējā rindā.
* Dalīšana ar nulli izdod rezultātu Error.
* Decimāldaļas tiek atbalstītas ar vienu punktu uz operandu.
* Rezultāti tiek formatēti: veseli skaitļi bez decimāldaļas, liekās nulles un punkti tiek noņemti.

Lietotne tika izstrādāta, izmantojot **Jetpack Compose** un **Kotlin**, ar stāvokļa pārvaldību caur remember { mutableStateOf() }

2.1 Lietotnes Ekrānšāviņš

A screenshot of a calculator

AI-generated content may be incorrect.

Šajā ekrānšāviņā redzams lietotnes sākuma stāvoklis. Displejā ir redzams skaitlis 0, kas norāda, ka lietotne ir gatava ievadei. Atmiņas pogas (MS, MR, MC, C)

A screenshot of a calculator

AI-generated content may be incorrect.

Šajā ekrānšāviņā redzams, ka lietotājs ir ievadījis vienadojums (5+5)

A screenshot of a calculator

AI-generated content may be incorrect.

Šajā ekrānšāviņā redzams, ka lietotājs ir nospiedis pogu MC (Memory Clear)

A screenshot of a calculator

AI-generated content may be incorrect.

**Kods  
MainActivity.ky**

package com.janis\_petrovs.calculator2

import android.os.Bundle

import android.widget.Toast

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.background

import androidx.compose.foundation.isSystemInDarkTheme

import androidx.compose.foundation.layout.\*

import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material3.\*

import androidx.compose.runtime.\*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.platform.LocalContext

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import kotlin.math.floor

private val Orange = Color(0xFFFF6D00)

private val DarkGray = Color(0xFF121212)

private val MediumGray = Color(0xFF1E1E1E)

private val LightGray = Color(0xFF2D2D2D)

private val TextWhite = Color(0xFFFFFFFF)

private val TextGray = Color(0xFFB3B3B3)

@Composable

fun CalculatorTheme(content: @Composable () -> Unit) {

MaterialTheme(

colorScheme = darkColorScheme(

primary = Orange,

secondary = Orange,

background = DarkGray,

surface = MediumGray,

onBackground = TextWhite,

onSurface = TextWhite

),

typography = Typography(),

content = content

)

}

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContent {

CalculatorTheme {

Surface(

modifier = Modifier

.fillMaxSize()

.background(DarkGray)

) {

CalculatorApp()

}

}

}

}

}

@Composable

fun CalculatorApp() {

var input by remember { mutableStateOf("") }

var result by remember { mutableStateOf("") }

var memory by remember { mutableStateOf(0.0) }

var currentNumber by remember { mutableStateOf(0.0) }

var currentOperator by remember { mutableStateOf("") }

var isNewInput by remember { mutableStateOf(true) }

val context = LocalContext.current

fun getCurrentOperand(): Double =

input.split(" ").lastOrNull()?.toDoubleOrNull() ?: 0.0

fun performCalculation(a: Double, b: Double, op: String): Double =

when (op) {

"+" -> a + b

"-" -> a - b

"\*" -> a \* b

"/" -> if (b != 0.0) a / b else Double.NaN

else -> b

}

fun formatNumber(num: Double): String {

if (num.isNaN() || num.isInfinite()) return "Error"

return if (num == floor(num)) num.toLong().toString()

else num.toString().trimEnd('0').trimEnd('.')

}

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

horizontalAlignment = Alignment.CenterHorizontally

) {

Column(

modifier = Modifier

.fillMaxWidth()

.background(LightGray, RoundedCornerShape(16.dp))

.padding(16.dp)

) {

Text(

text = input.ifEmpty { "0" },

modifier = Modifier.fillMaxWidth(),

textAlign = TextAlign.End,

fontSize = 28.sp,

color = TextGray

)

Text(

text = result,

modifier = Modifier

.fillMaxWidth()

.padding(top = 4.dp),

textAlign = TextAlign.End,

fontSize = 36.sp,

color = TextWhite

)

}

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

Row(modifier = Modifier.fillMaxWidth()) {

listOf("MS", "MR", "MC", "C").forEach { txt ->

CalculatorButton(

text = txt,

isOperator = false,

isEquals = false,

modifier = Modifier.weight(1f)

) {

when (txt) {

"MS" -> {

memory = getCurrentOperand()

Toast.makeText(context, "Saved $memory", Toast.LENGTH\_SHORT).show()

}

"MR" -> {

val mem = formatNumber(memory)

input = if (isNewInput) mem else input + mem

result = ""

}

"MC" -> {

memory = 0.0

Toast.makeText(context, "Memory cleared", Toast.LENGTH\_SHORT).show()

}

"C" -> {

input = ""

result = ""

currentNumber = 0.0

currentOperator = ""

isNewInput = true

}

}

}

}

}

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

CalculatorGrid { button ->

when (button) {

in "0".."9" -> {

if (isNewInput) {

input = button

isNewInput = false

} else {

input += button

}

result = ""

}

"." -> {

val lastPart = input.split(" ").lastOrNull() ?: ""

if (!lastPart.contains(".")) {

if (isNewInput) {

input = "0."

isNewInput = false

} else {

input += "."

}

result = ""

}

}

in listOf("+", "-", "\*", "/") -> {

if (input.isNotEmpty() && !isNewInput) {

val operand = getCurrentOperand()

if (currentOperator.isNotEmpty()) {

val res = performCalculation(currentNumber, operand, currentOperator)

input = "${formatNumber(res)} $button "

currentNumber = if (res.isNaN()) 0.0 else res

} else {

currentNumber = operand

input += " $button "

}

currentOperator = button

isNewInput = true

result = ""

}

}

"=" -> {

if (currentOperator.isNotEmpty() && input.isNotEmpty()) {

val operand = getCurrentOperand()

val res = performCalculation(currentNumber, operand, currentOperator)

result = formatNumber(res)

input += " ="

currentOperator = ""

isNewInput = true

}

}

}

}

}

}

@Composable

fun CalculatorButton(

text: String,

isOperator: Boolean = text in listOf("+", "-", "\*", "/"),

isEquals: Boolean = text == "=",

modifier: Modifier = Modifier,

onClick: () -> Unit

) {

Button(

onClick = onClick,

modifier = modifier

.padding(4.dp)

.height(64.dp),

shape = RoundedCornerShape(16.dp),

colors = ButtonDefaults.buttonColors(

containerColor = when {

isEquals -> Orange

isOperator -> Orange.copy(alpha = 0.9f)

else -> LightGray

},

contentColor = if (isOperator || isEquals) Color.Black else TextWhite

)

) {

Text(

text = text,

fontSize = 22.sp,

fontWeight = androidx.compose.ui.text.font.FontWeight.Medium

)

}

}

@Composable

fun CalculatorGrid(onButtonClick: (String) -> Unit) {

val rows = listOf(

listOf("7", "8", "9", "/"),

listOf("4", "5", "6", "\*"),

listOf("1", "2", "3", "-"),

listOf("0", ".", "=", "+")

)

Column(modifier = Modifier.fillMaxWidth()) {

rows.forEach { row ->

Row(modifier = Modifier.fillMaxWidth()) {

row.forEach { btn ->

CalculatorButton(

text = btn,

modifier = Modifier.weight(1f)

) { onButtonClick(btn) }

}

}

}

}

}

# Pielikums

Github liks uz “repository” https://github.com/MrAlphaF/ViA\_mobile4