Министерство образования Самарской области

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ САМАРСКОЙ ОБЛАСТИ «ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ КОЛЛЕДЖ»

Составитель: Лазарев Т.А.

МЕТОДИЧЕСКОЕ ПОСОБИЕ № 2 ДЛЯ ОПЦ.В.16 ОСНОВЫ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

«профессиональный цикл»

программы подготовки специалистов среднего звена

09.02.07 Информационные системы и программирование

Самара, 2025

Методическое пособие для второй практической работы

Содержание

Начало работы.

Необходимо сделать возможным создавать привязки на представления. Для этого нужно перейти в файл gradle и написать следующий код (рис. 1):

```
6   android {
7       namespace = "com.example.fragmentlifecycleexamples"
8       compileSdk = 36
9
10      buildFeatures {
11          viewBinding = true
12      }
13
```

Рисунок 1 – Настройка View Binding

Требуется написать buildFeatures { viewBinding=true } внутри желтых скобок

Для выполнения второй практической требуется создать вторую активити. Это можно сделать вручную, создав запись в AndroidManifest для новой активити и создания дочернего класса AppCompatActivity, но AndroidStudio предлагает более удобный автоматический способ. Для этого требуется нажать на корневую папку проекта правой кнопкой мыши.

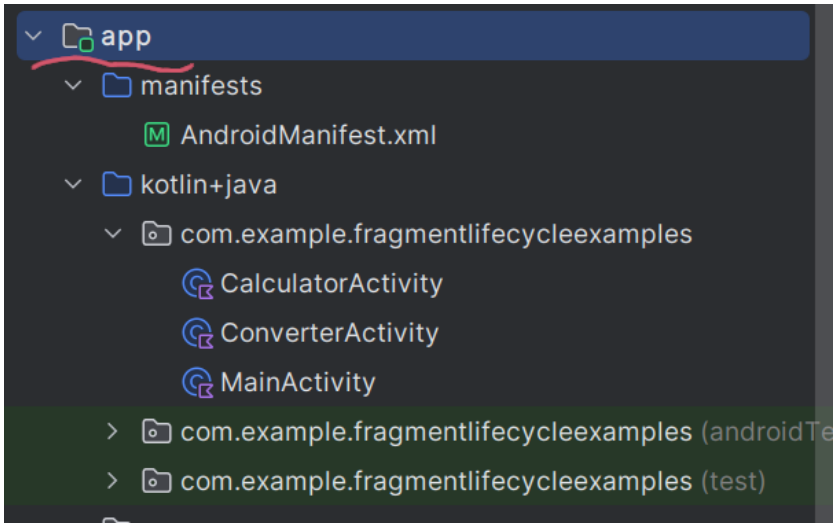

Рисунок 2 – Содержимое корневой папки app

После чего выберите **New > Activity > Empty Views Activity.** После этого, должно открыться следующее окно (рис. 3):

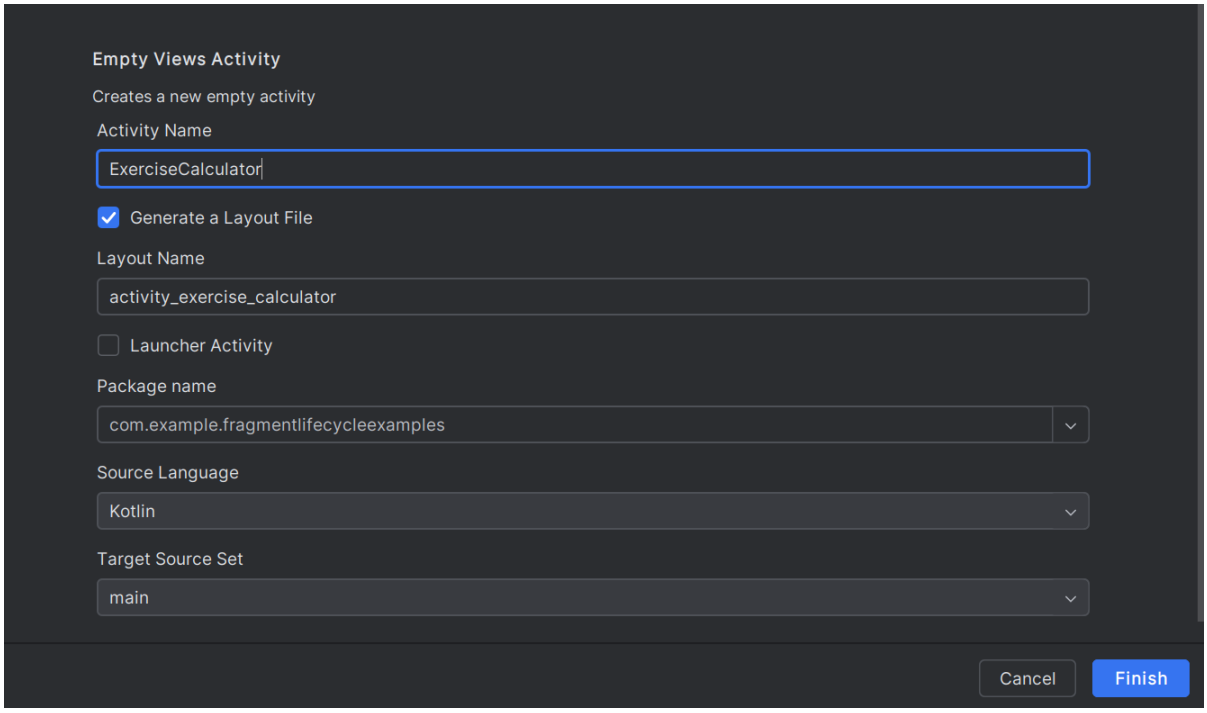

Рисунок 3 – Окно создания активити

Название файла макета соответствует названию активити в snake_case с префиксом activity. После чего, у вас должен появится файл с вашей активити со следующим содержимым (рис. 4):

Рисунок 4 – Второй экран, файл новой активити

Все строчки кода подчеркнутые зеленым цветом можно удалить. Они не нужны для выполнения задания.

Далее, требуется перейти на первый экран и создать событие на кнопку, которая при щелчке должна переместить пользователя на второй экран. На скриншоте (рис 5) всё, что начинается и заканчивается желтой фигурной скобкой является телом класса. Всё, что начинается и заканчивается зеленой фигурной скобкой является телом метода onCreate().

Переменная binding содержит в себе привязку к представлениям, создается она из класса, сгенерированного из названия макета. Из неё можно получить ссылку на кнопку не используя функцию findViewById.

Требуется совершить подписку на событие щелчка кнопки, это можно сделать через вызов метода setOnClickListener, передав в функцию лямбда-выражение.

Синие скобки отличаются от всех прочих фигурных скобок, они являются лямбда-выражением или анонимной функцией, функцией без имени. Они не являются телом метода или телом цикла.

Внутри лямбда выражения вызывается активити через явное намеренье (Intent). Такое действие должно переносить пользователя на калькулятор.

Все подписки на события должны совершаться из тела метода onCreate (рис. 5):

```
2 Usages
class MainActivity : AppCompatActivity() {

    3 Usages
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        binding.nextButton
            .setOnClickListener {
                startActivity(Intent( packageContext = this,  cls = ExerciseCalculator::class.java))
            }

    }
}
```

Рисунок 5 – Первый экран, подписка на событие клика

Далее на втором экране требуется сохранять данные при перевороте экрана (рис. 6). Здесь также показан пример обработки события кнопки и объявления и изменения переменной.

```
2 Usages
class ExerciseCalculator : AppCompatActivity() {
    4 Usages
    private lateinit var binding: ActivityExerciseCalculatorBinding
    4 Usages
    private var resultText = ""

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityExerciseCalculatorBinding.inflate(layoutInflater)
        setContentView(binding.root)
        savedInstanceState?.let {
            resultText = it.getString( key = RESULT,  defaultValue = "")
        }
        binding.num1.setOnClickListener {
            resultText += "1"
            binding.resultBox.text = resultText
        }
    }
    override fun onSaveInstanceState(outState: Bundle, outPersistentState: PersistableBundle) =
        with( receiver = outState) {
            putString(RESULT, resultText)
        }

    2 Usages
    companion object {
        2 Usages
        const val RESULT = "RESULT"
    }
}
```

Рисунок 6 – Восстановление состояния

Блок схема.

Блок схема для алгоритма калькулятора выглядит следующим образом (рис. 7):

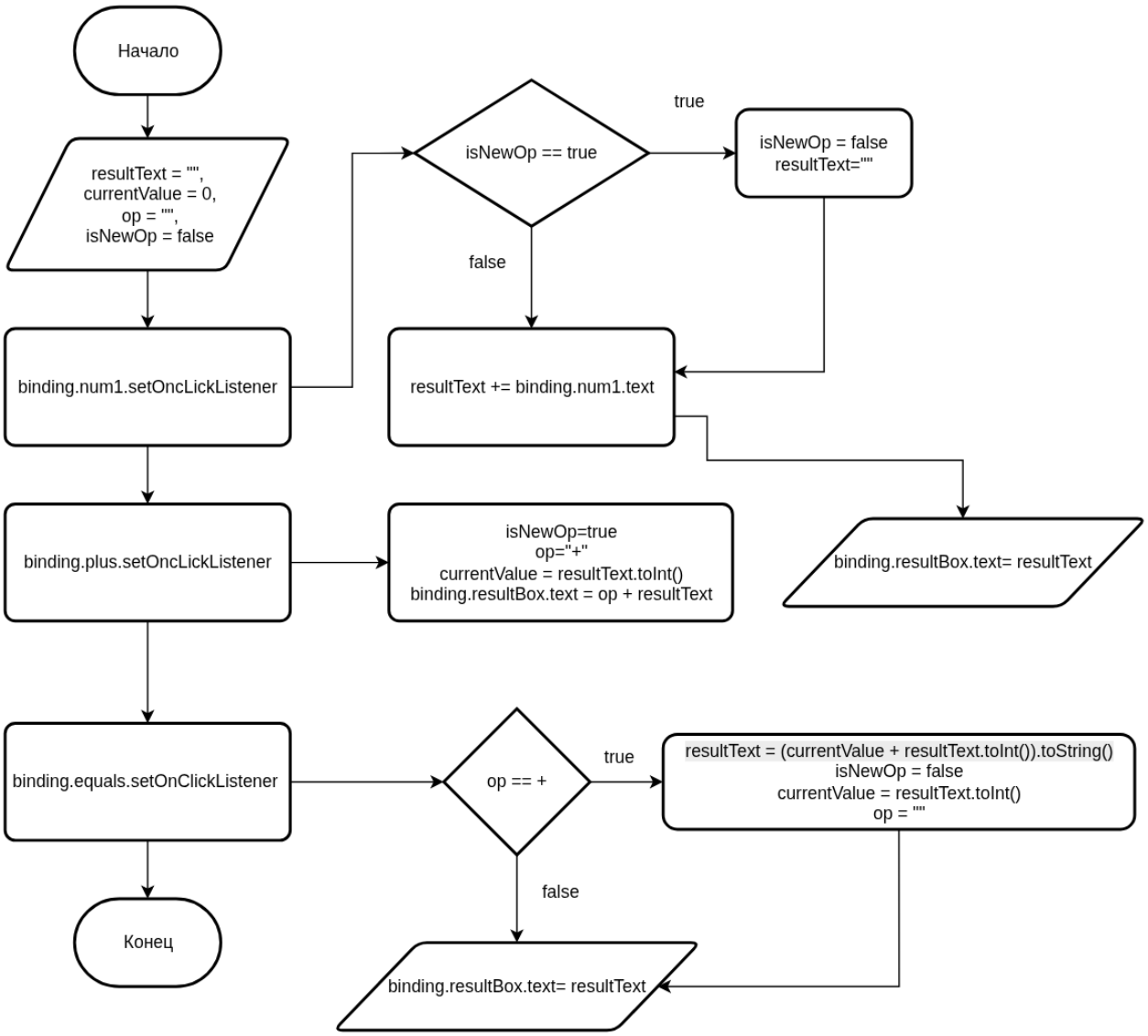

Рисунок 7 – Блок схема калькулятора