КУРСОВая Работа

Создание приложений на языке Kotlin

с использованием парадигм функционального программирования

по дисциплине

«Языки и средства функционального программирования»

Выполнил

студент гр.3530904/80007 Луценко Д.Ю

Руководитель  Лукашин А.А

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Содержание

[Цель 3](#_Toc27527484)

[Введение 4](#_Toc27527485)

[Структура 5](#_Toc27527486)

[Демонстрация работы 6](#_Toc27527487)

[Заключение 7](#_Toc27527488)

Цель

1. Разработать приложение «Калькулятор» под Android на языке Kotlin с использованием знаний, о функциональном программировании, полученных за семестр.

# Введение

У функционального программирования есть много преимуществ, но возможного максимального использования ресурсов процессора благодаря конкурентному поведению — это его главный плюс. Ниже привел основные принципы функционального программирования, в том числе, которые были использованы.

1. **Все функции — чистые**

Это правило безусловно является основным в функциональном программировании. Все функции являются чистыми, если они удовлетворяют двум условиям:

1. Функция, вызываемая от одних и тех же аргументов, всегда возвращает одинаковое значение.
2. Во время выполнения функции не возникают побочные эффекты.
3. **Все функции — первого класса и высшего порядка**

Эта концепция — не особенность ФП (она используется в Javascript, PHP и других языках) — но его обязательное требование. Для того, чтобы функция была первоклассной, у неё должна быть возможность быть объявленной в виде переменной. Это позволяет управлять функцией как обычным типом данных и в то же время исполнять её.

Функции высшего порядка же определяются как функции, принимающие другую функцию как аргумент или возвращающие функцию. Типичными примерами таких функций являются map и filter.

1. **Переменные неизменяемы**

Тут всё просто. В функциональном программировании вы не можете изменить переменную после её инициализации. Вы можете создавать новые, но не можете изменять существующие — и благодаря этому вы можете быть уверены, что никакая переменная не изменится.

1. **Относительная прозрачность функций**

Сложно дать корректное определение относительной прозрачности. Самым точным я считаю такое: если вы можете заменить вызов функции на возвращаемое значение, и состояние при этом не изменится, то функция относительно прозрачна. Это, быть может, очевидно, но я приведу пример.

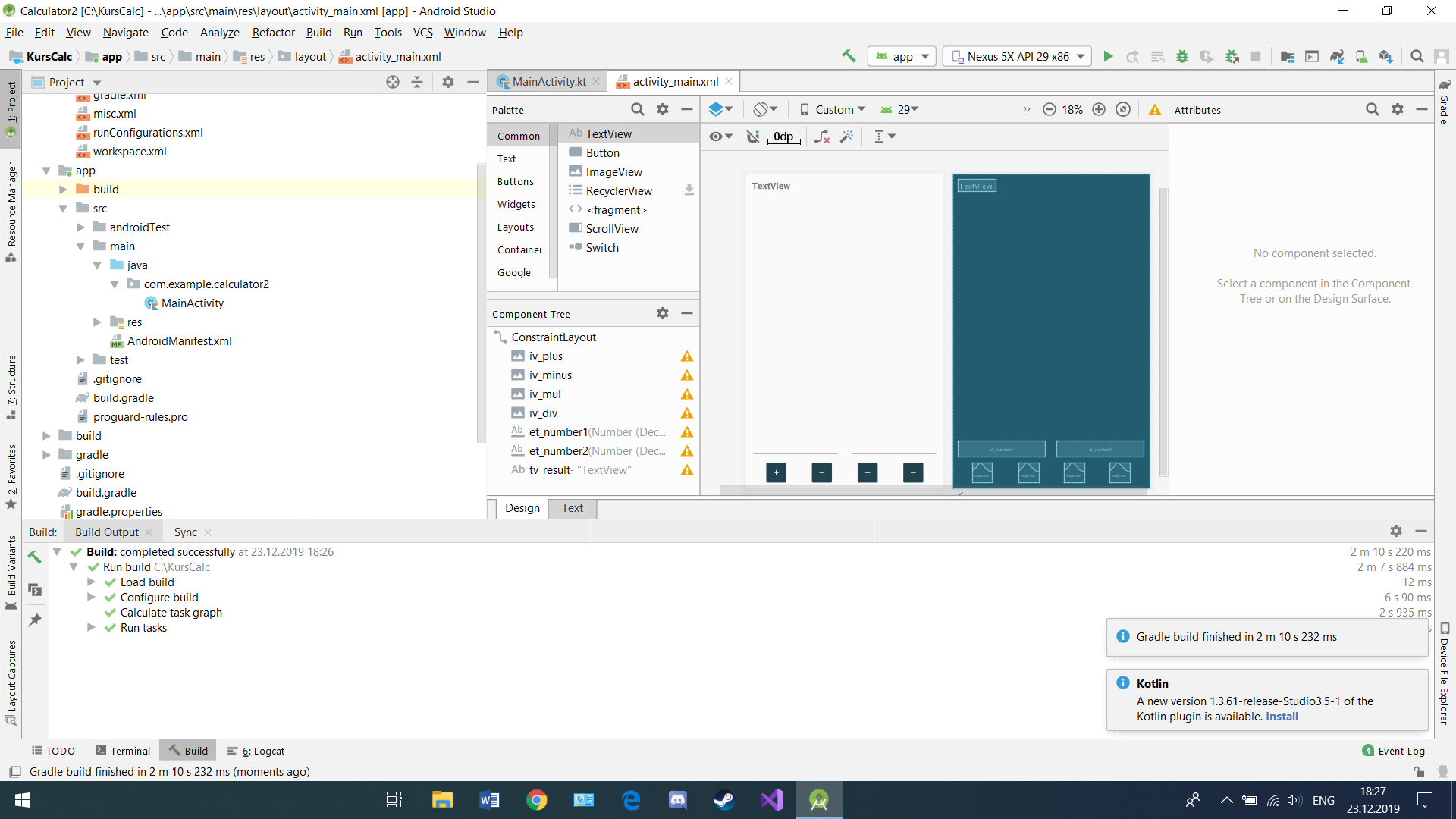
1. **Функциональное программирование основано на лямбда-исчислении**

Функциональное программирование сильно опирается на математическую систему, называющуюся лямбда-исчислением.

1. В лямбда-исчислении все функции могут быть анонимными, поскольку единственная значимая часть заголовка функции — это список аргументов.
2. При вызове все функции проходят процесс каррирования. Он заключается в следующем: если вызывается функция с несколькими аргументами, то сперва она будет выполнена лишь с первым аргументом и вернёт новую функцию, содержащую на 1 аргумент меньше, которая будет немедленно вызвана

# Структура

Внешний вид калькулятора основан на положение телефона и написан на расширяемом языке разметки XML.



Основная логика приложения написана в классе MainActivity.kt

OnCreate – создаем приложение с проверкой ориентации телефона

Имеется 4 кнопки, соответствующие каждой из 4-х арифметической операции: +, -, \*, / и текстовое поле с результатом*.*

*Пример фрагмента кода для “+”*

ivPlus.setOnClickListener {

if (handleNoInputNumbers())

return@setOnClickListener

val number1 = etNumber1.text.toString().toBigDecimal()

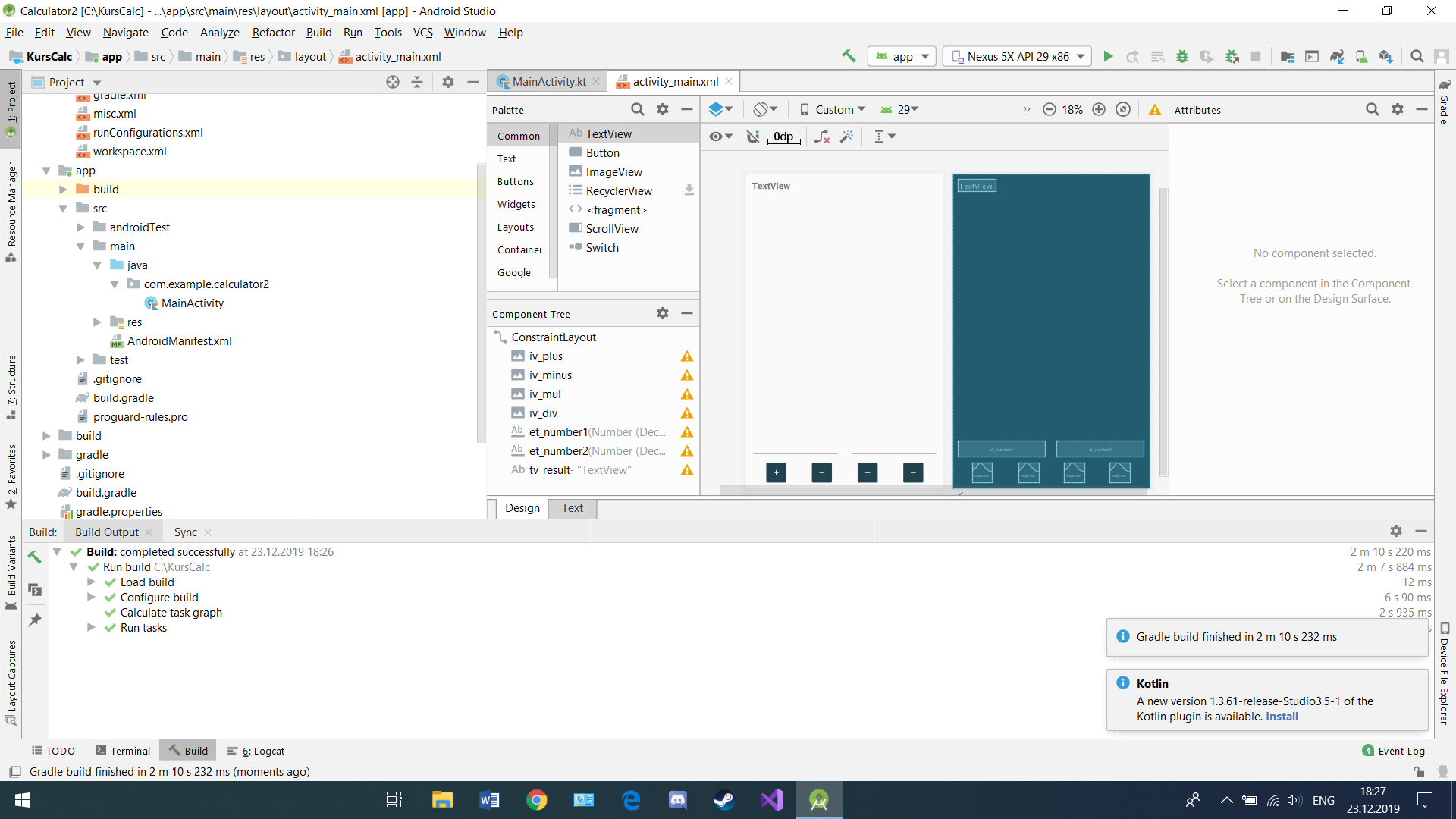
val number2 = etNumber2.text.toString().toBigDecimal()

val sum = number1 + number2

tvResult.text = sum.toString()

}

Демонстрация работы



Заключение

В данной курсовой работе реализована программа «Калькулятор» на языке Kotlin на Android.

В результате данной работы были получены знания и навыки в написании программ, использующих функциональный подход к программированию, стандартных библиотек для разработки программ на Android, а также лучше усвоены возможности языка Kotlin.