МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Балтийский федеральный университет имени Иммануила Канта»

ОНК «Институт высоких технологий»

ОТЧЁТ О ПРОХОЖДЕНИИ

УЧЕБНОЙ ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ

на базе Высшей школы компьютерных наук и искусственного интеллекта

Выполнил \_\_\_\_\_\_\_Портнов Егор Павлович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студент очной формы обучения 1 курса

направления подготовки 01.03.02 Прикладная математика и информатика

профиль обучения «Информатика и программирование»

Руководитель практики

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Тарачков М.В.

г. Калининград 2025 г.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc169703047)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_Toc169703048)

[Актуальность темы 4](#_Toc169703049)

Похожие проекты и используемые технологии…………………………5

Глава 2. Задания на практику…………………..………………………………..7

[Задача №1 «Пользовательский интерфейс» 7](#_Toc169703051)

[Задача №2 «Функционал» 7](#_Toc169703052)

[Задача №3 «Тестирование» 8](#_Toc169703053)

[Глава 3. Выполнение заданий на практику 9](#_Toc169703055)

[Решение задачи №1 9](#_Toc169703056)

[Решение задачи №2 10](#_Toc169703057)

[Решение задачи №3 10](#_Toc169703058)

[ЗАКЛЮЧЕНИЕ 12](#_Toc169703060)

[СПИСОК ЛИТЕРАТУРЫ 14](#_Toc169703061)

[ПРИЛОЖЕНИЕ 1 15](#_Toc169703062)

[ПРИЛОЖЕНИЕ 2 17](#_Toc169703063)

[ПРИЛОЖЕНИЕ 3 18](#_Toc169703063)

[ПРИЛОЖЕНИЕ 4 19](#_Toc169703063)

**ВВЕДЕНИЕ**

Вид практики – Учебная технологическая (проектно-технологическая) практика (далее Учебная практика).

Цель учебной практики:

1. Создать простое приложение калькулятор в Android Studio на языке Kotlin
2. Протестировать приложение
3. Создать apk-файл приложения

Задачи учебной практики:

1. Установить и настроить среду разработки Android Studio.
2. Разработать пользовательский интерфейс приложения с использованием XML-разметки.
3. Реализовать логику работы калькулятора (сложение, вычитание, умножение, деление).
4. Протестировать приложение на эмуляторе или реальном устройстве.

**ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Актуальность темы**

Android является лидирующей мобильной ОС с более чем 2 млрд активных устройств и 3 млн приложений в магазине Google Play [1]. Kotlin, официально поддерживаемый Google с 2017 года, стал предпочтительным языком для Android-разработки благодаря [2]:

* Совместимости с Java и возможности использования существующих библиотек.
* Упрощенному синтаксису, снижающему объем кода на 30–40% по сравнению с Java 9.
* Null safety — системе типизации, исключающей ошибки NullPointerException.
* Мультиплатформенности (Kotlin Multiplatform) для кросс-платформенной разработки.

**2. Похожие проекты и используемые технологии [3][4]**

2.1 Корпоративные приложения

* Платформа управления недвижимостью (Argo):
  + Технологии: Kotlin Multiplatform Mobile (KMM), Room для локального хранилища, Kotlin Coroutines для асинхронных операций. [3]
  + Причины выбора: KMM позволил reuse бизнес-логики для iOS/Android, сократив время разработки на 40%. Coroutines упростили работу с сетевыми запросами без блокировки UI 1315. [4]
* Медицинское приложение для реабилитации после COVID-19 (OXYFOR):
  + Технологии: Jetpack Compose (для кастомного UI), Ktor для API, WorkManager для фоновых задач.
  + Причины: Jetpack Compose ускорил создание анимированных интерфейсов, а Ktor обеспечил лёгкую интеграцию с REST API 13.

2.2 Кроссплатформенные решения

* Приложение Pinterest:
  + Технологии: Kotlin с частичным использованием KMM для синхронизации данных между iOS и Android.
  + Причины: Миграция с Java на Kotlin сократила код на 30%, а KMM позволил использовать общую логику для фотоальбомов и платежей 15.
* Uber (внутренние инструменты):
  + Технологии: Kotlin/Native для Gradle-плагинов, Kotlin Coroutines.
  + Причины: Совместимость с Java-библиотеками и безопасность типов предотвратили ошибки в критических модулях 15.

Таблица 1: Сравнение технологий для Android-разработки

|  |  |  |
| --- | --- | --- |
| Технология | Преимущества | **Ограничения** |
| Kotlin | Null safety, совместимость с Java, лаконичность | Долгая компиляция в больших проектах |
| Android Studio | Официальная среда, встроенный эмулятор | Требовательна к ресурсам ПК |
| ConstraintLayout | Гибкое позиционирование элементов | Сложность освоения |

Почему Kotlin доминирует в Android? [5]

* Безопасность null-значений: Контроль NullPointerException через оператор «?» снижает количество падений приложений.
* Совместимость с Java: Возможность интеграции с legacy-кодом и библиотеками (например, Retrofit, Gson).
* Лаконичность синтаксиса: Код на Kotlin на 40% короче Java-аналогов (сравнение функций в [5])

**ГЛАВА 2. ЗАДАНИЕ НА ПРАКТИКУ**

**Задача №1 «Пользовательский интерфейс»**

Для реализации интерфейса калькулятора необходимо:

1. Разработать XML-разметку с использованием следующих компонентов:

* Основной контейнер LinearLayout с вертикальной ориентацией
* Два TextView для отображения вводимых операций и результатов
* Набор TextView для кнопок калькулятора, организованных в 6 горизонтальных рядов

1. Применить стили оформления:

* Для цифровых кнопок: серый фон (#403e3d), белый текст
* Для основных операций: темный фон (#171616), оранжевый текст (#fc4f05)
* Для научных функций: темный фон (#171616), серый текст (#999594)
* Для кнопки "=": оранжевый фон (#f7550f), белый текст

1. Реализовать адаптивное расположение элементов:

* Использование weight для пропорционального распределения пространства
* Отступы 0.5dp между кнопками
* Разделение экрана на область вывода и клавиатуру

**Задача №2 «Функционал»**

Приложение должно обеспечивать:

1. Базовые арифметические операции:

* Сложение, вычитание, умножение, деление
* Работу с процентами
* Использование скобок

1. Научные функции:

* Тригонометрические вычисления (sin, cos)
* Логарифмы (ln, log2)
* Квадратный корень
* Возведение в степень
* Работу с константами (π, e)

1. Служебные функции:

* Полную очистку ввода (AC)
* Удаление последнего символа (back)
* Вычисление и вывод результата (=)

**Задача №3 «Тестирование»**

Методы тестирования:

1. Базовое функциональное тестирование:
   * Проверка всех математических операций
   * Тестирование обработки ошибок
   * Проверка корректности отображения интерфейса
2. Тестирование на разных платформах:
   * Эмулятор Android Studio
   * Физические устройства (разные производители и версии ОС)

**ГЛАВА 3. Выполнение заданий на практику**

**Решение задачи №1**

Для создания интерфейса калькулятора была разработана иерархическая структура компонентов:

1. **Корневой макет**:

* Реализован через LinearLayout с вертикальной ориентацией
* Установлен черный фон (#FF000000)
* Разделен на две области с использованием layout\_weight

1. **Область вывода**:

* Включает два TextView:
  + operation (50sp) - для отображения вводимой последовательности
  + result (30sp) - для вывода результатов вычислений
* Настроено выравнивание по правому краю (gravity="end")
* Добавлены свойства singleLine и ellipsize для обработки длинных выражений

1. **Клавиатура калькулятора**:

* Организована в виде 6 вложенных LinearLayout с горизонтальной ориентацией. В каждом LinearLayout находится еще 5 TextView.
* Для кнопок применены три стиля

Код программы расположен в приложении №1

**Решение задачи №2**

Логика приложения реализована в классе MainActivity.kt и включает обработку пользовательского ввода, выполнение математических операций и управление состоянием интерфейса.

Все элементы интерфейса связаны с переменными через findViewById.

Каждая кнопка имеет setOnClickListener, который добавляет оператор в строку ввода (operation) и для специальных кнопок выполняет отдельные действия

Были добавлены также такие особые функции, как очистка ввода и удаление последнего символа

Код программы расположен в приложении №2.

Для вычислений была подключена к проекту библиотека exp4j с помощью строки *implementation*("net.objecthunter:exp4j:0.4.8") в файле build.gradle.kts (:app) и позже добавлена в класс MainActivity.kt, где была задействована в кнопке «=» (код программы расположен в приложении №3).

**Решение задачи №3**

Был создан эмулятор на Android 16 и проверены:

* + Все базовые операции (сложение, вычитание и др.)
  + Научные функции (тригонометрия, логарифмы)
  + Краевые случаи (деление на ноль, пустой ввод)

Тестирование на физическом устройстве:

* + Собран APK-файл (Build → Generate Signed app Bundle or APK)
  + Установлен на Huawei P40 Pro (Android 10)
  + Проверены скорость работы и корректность вычислений

Результаты тестирования:

* Все основные функции работают корректно
* Интерфейс отображается правильно на разных устройствах
* Была выявлена ошибка. При первоначальном запуске приложения, до нажатия кнопки «back» активны только следующие кнопки: «sqrt», «log2», «ln», «(«, «)», «x^y».
* Была выявлена вторая ошибка. При попытке вычислить **√**x (например, **√**9), приложение выдает «Error»

Исправления:

Причиной первой ошибки служило то, что обработчики нажатий для основной клавиатуры были ошибочно размещены внутри блока b\_back.setOnClickListener. В результате они активировались только после первого нажатия кнопки «back».

Причиной второй ошибки оказалось то, что библиотека exp4j ожидает синтаксис «sqrt(x)» для вычисления корня.

Исходные и исправленные версии кода программы расположены в приложении №4

**ЗАКЛЮЧЕНИЕ**

В ходе учебной практики я получил ценный опыт разработки Android-приложения на Kotlin. Основной задачей было создание научного калькулятора с интуитивно понятным интерфейсом и широким функционалом. Работа над проектом позволила мне глубже понять принципы мобильной разработки и особенности создания пользовательских интерфейсов.

Особое внимание уделялось реализации математического ядра приложения. Использование библиотеки exp4j дало возможность работать с различными типами выражений, от простых арифметических операций до сложных научных вычислений. В процессе разработки я научился анализировать и исправлять ошибки, такие как некорректная обработка ввода или проблемы с вычислением корней.

Важной частью работы стало тестирование приложения. Проверка функционала на эмуляторе и реальном устройстве помогла выявить и устранить критические проблемы, включая особенности работы интерфейса и точность вычислений. Этот опыт показал мне важность комплексного тестирования на разных платформах.

В ходе практики я освоил ключевые аспекты разработки под Android: создание адаптивных интерфейсов, работу с XML-разметкой, интеграцию сторонних библиотек, обработку пользовательского ввода и отладку приложений. Особенно полезным оказался опыт решения реальных проблем, с которыми сталкиваются разработчики.

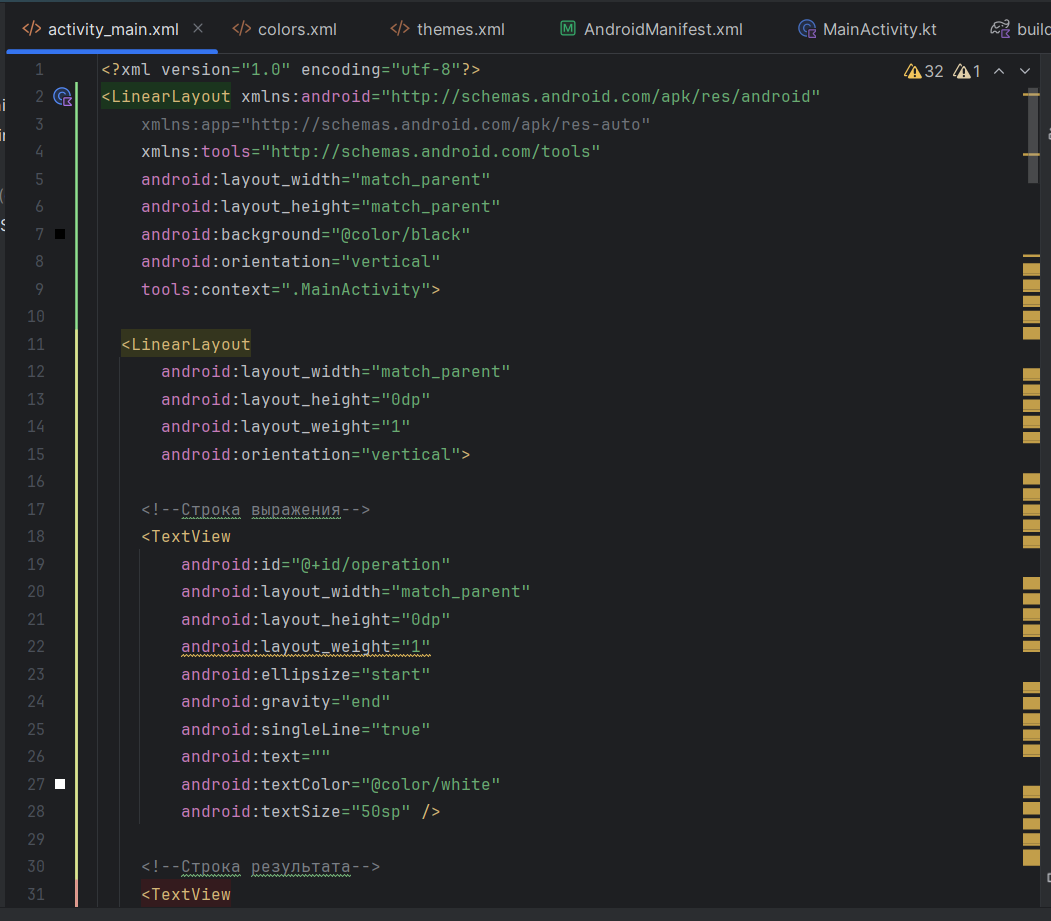
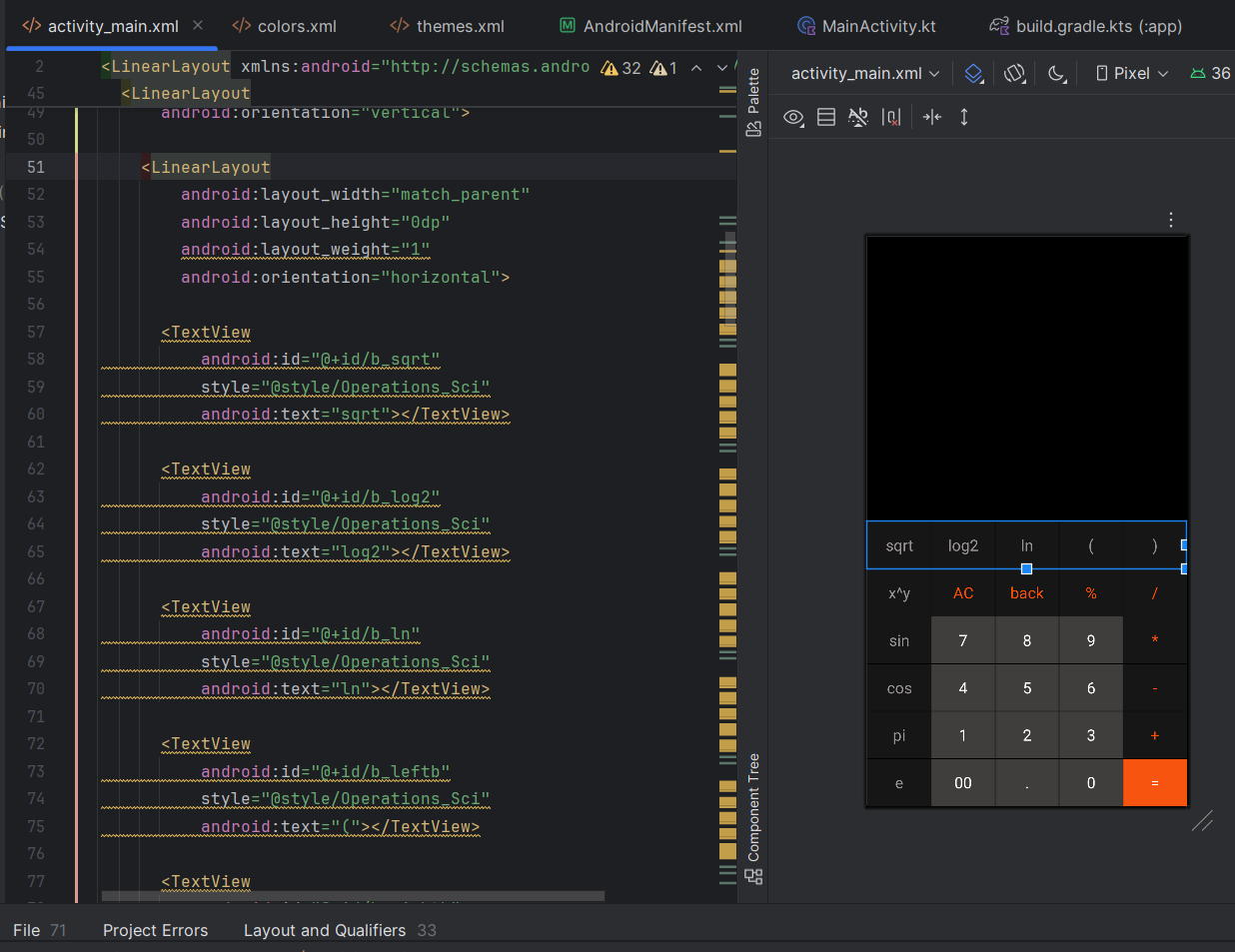
Все поставленные задачи были успешно выполнены. Созданное приложение обладает стабильным функционалом, удобным интерфейсом и корректно работает на различных устройствах.

**СПИСОК ЛИТЕРАТУРЫ**

**Перечень ресурсов информационно-телекоммуникационной сети «Интернет»**

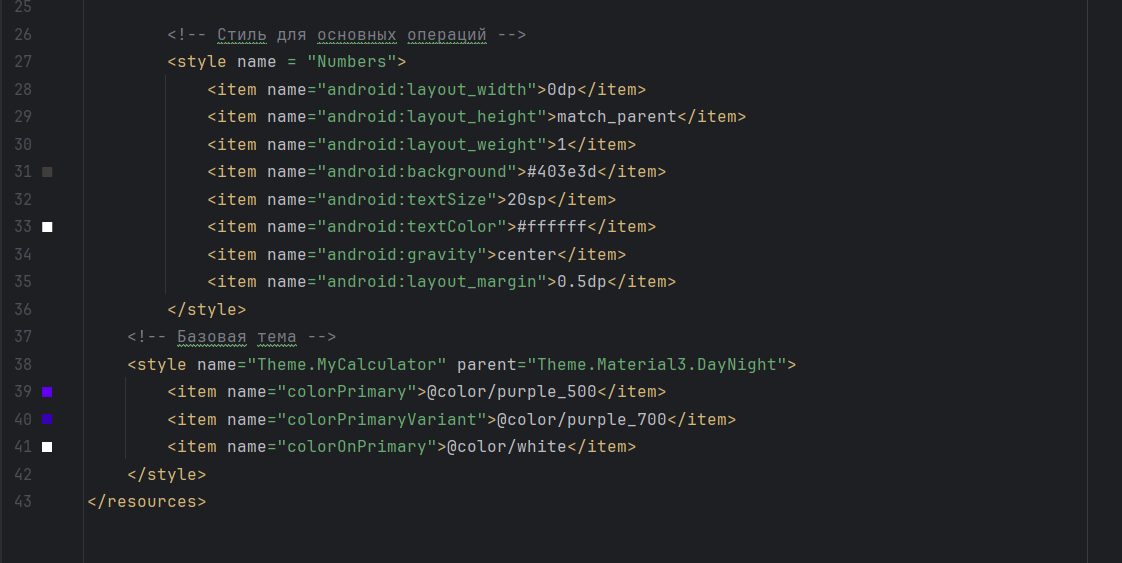
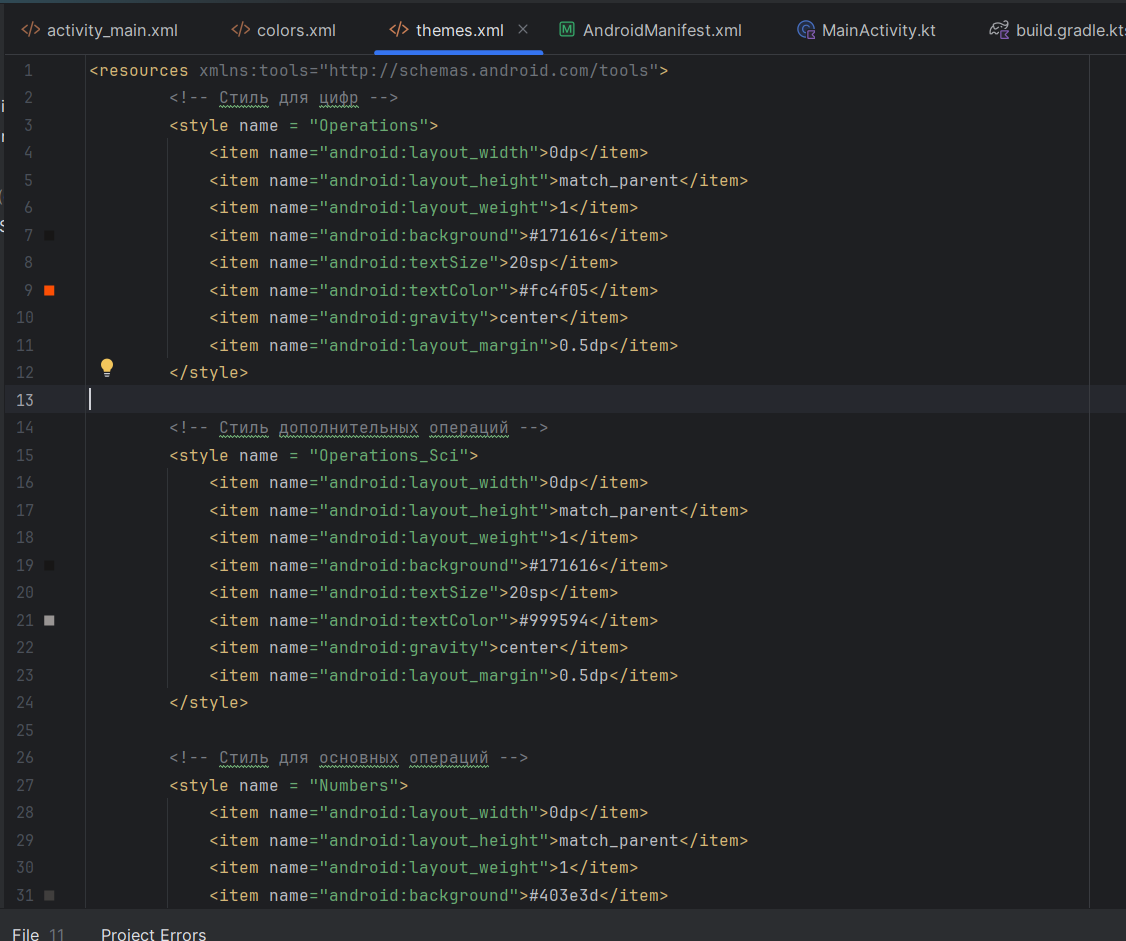
1. Разработка Android-приложений на Kotkin. [URL: https://stepik.org/course/4792/promo](URL:%20https://stepik.org/course/4792/promo)
2. Kotlin – язык программирования: для чего используется, плюсы и минусы. URL: <https://habr.com/ru/articles/783456/>
3. Разработка мобильного приложения на Kotlin. URL: <https://sibdev.pro/kotlin>
4. Flutter или Kotlin: что выбрать бизнесу в 2025? URL: <https://worksolutions.ru/useful/flutter-ili-kotlin-chto-vybrat-biznesu/>
5. Язык андроидов: почему почти все приложения для Android написаны на Kotlin. URL: <https://practicum.yandex.ru/blog/yazyk-programmirovaniya-kotlin-dlya-android-razrabotki/>
6. Создание простого калькулятора в Android-Studio. URL: <https://habr.com/ru/articles/779464/>

**ПРИЛОЖЕНИЕ 1**

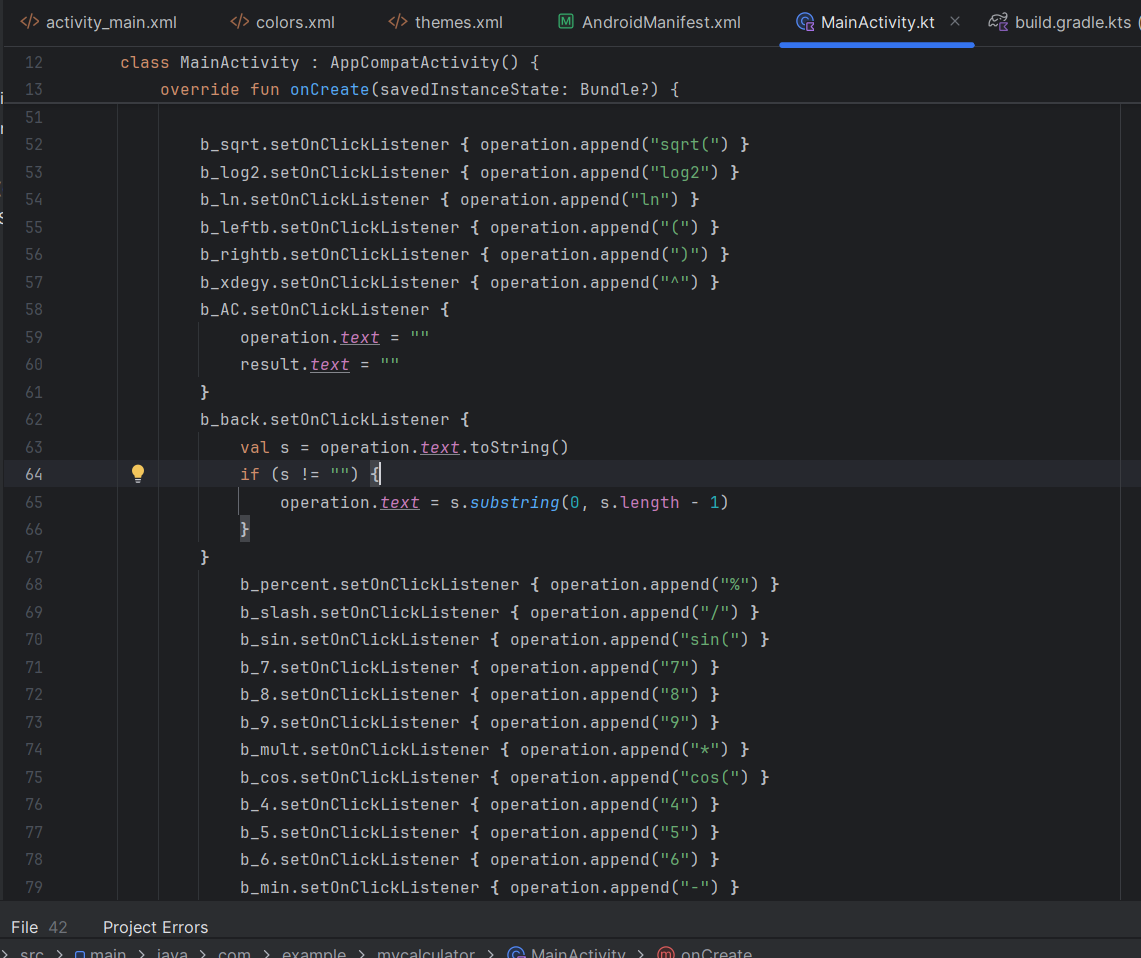
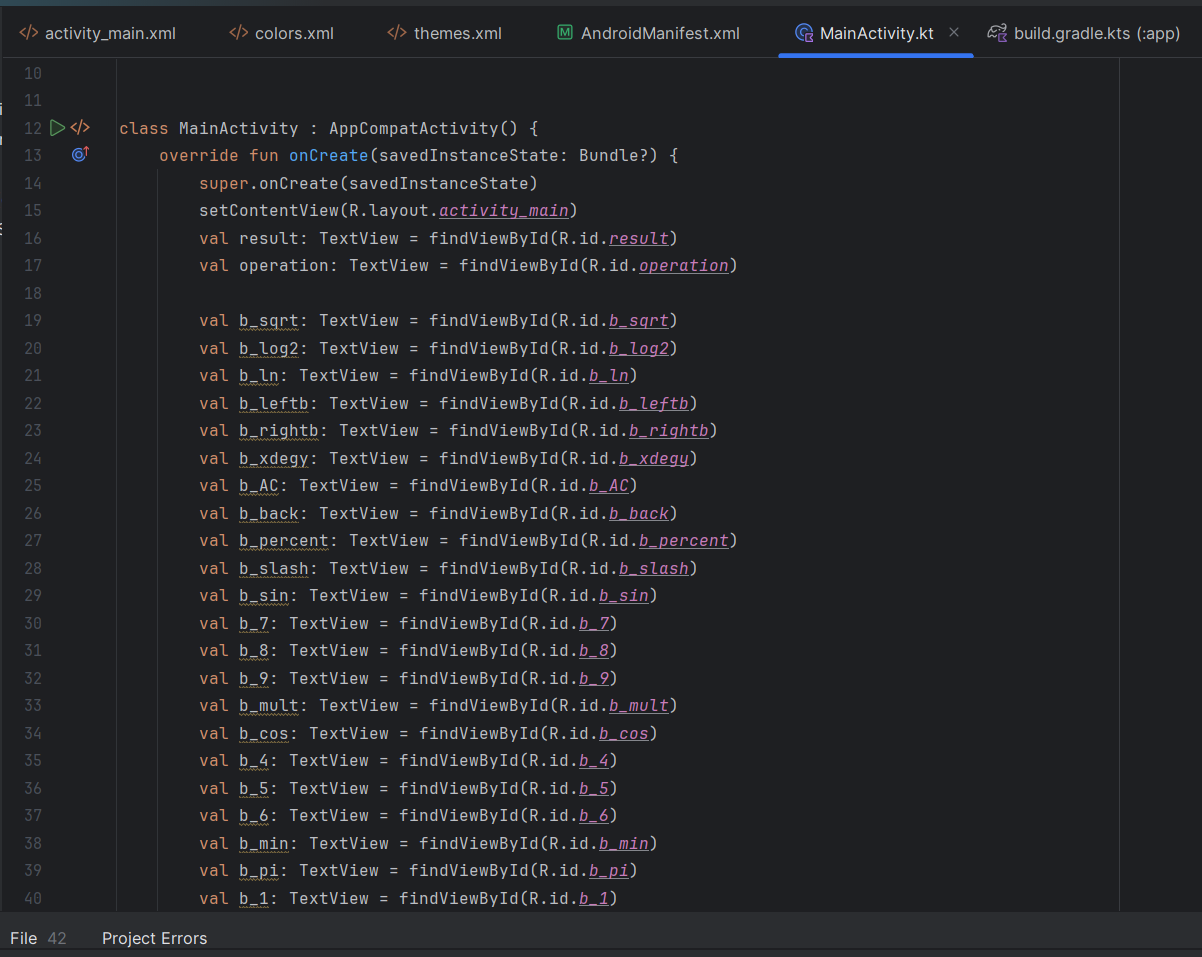
 

**ПРИЛОЖЕНЕ 1 (продолжение)**

Стили:



**ПРИЛОЖЕНИЕ 2**

****

**ПРИЛОЖЕНИЕ 3**

b\_equal.setOnClickListener {  
 val optext = operation.*text*.toString()  
 if (optext != "") {  
 try {  
 val expr = ExpressionBuilder(operation.*text*.toString()).build()  
 val res = expr.evaluate()  
 val longres = res.toLong()  
 if (longres.toDouble() == res) {  
 result.*text* = longres.toString()  
 } else {  
 result.*text* = res.toString()  
 }  
 } catch (e: Exception) {  
 result.*text* = "Error"  
 }  
 }  
}

**ПРИЛОЖЕНИЕ 4**

Первая ошибка:

Исходный:

b\_back.setOnClickListener {  
 val s = operation.*text*.toString()  
 if (s != "") {

operation.*text* = s.*substring*(0, s.length - 1)

}

…

result.setOnClickListener {  
 val restext = result.*text*.toString()  
 if (restext != "Error" && restext != "")

{  
 operation.*text* = restext  
 result.*text* = ""  
 }  
 }  
 }  
 }

}

**ПРИЛОЖЕНИИЕ 4 (продолжение)**

Исправленный:

b\_back.setOnClickListener {  
 val s = operation.*text*.toString()  
 if (s != "") {  
 operation.*text* = s.*substring*(0, s.length - 1)  
 }  
}

…

result.setOnClickListener {  
 val restext = result.*text*.toString()  
 if (restext != "Error" && restext != "") {  
 operation.*text* = restext  
 result.*text* = ""  
 }  
 }  
 }

}

Вторая ошибка:

Исходный: b\_sqrt.setOnClickListener **{** operation.append("**√**"**}**

Исправленный: b\_sqrt.setOnClickListener **{** operation.append("sqrt(") **}**