СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_gjdgxs)

[ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_30j0zll)

[ПРАКТИЧЕСКАЯ ЧАСТЬ 13](#_1fob9te)

[ЗАКЛЮЧЕНИЕ 28](#_fzuloosvskli)

# ВВЕДЕНИЕ

Введение к проекту приложения «Калькулятор» играет важную роль для определения цели работы, контекста разработки и значимости приложения в повседневной жизни. Калькуляторы являются одними из самых распространенных и необходимых инструментов, используемых как в образовательных учреждениях, так и в профессиональной сфере. С усложнением математических вычислений и ростом потребности в удобстве, мобильные приложения становятся незаменимыми помощниками для пользователей, нуждающихся в быстрой и точной обработке числовых данных.

В современном мире, где время и эффективность имеют первостепенное значение, наличие под рукой практичного калькулятора на смартфоне позволяет выполнять различные вычисления в любой момент. Основная цель нашего проекта заключается в разработке мобильного приложения, которое обеспечит пользователей интуитивно понятным и функциональным интерфейсом для выполнения базовых математических операций. Приложение будет включать традиционные функции, такие как сложение, вычитание, умножение и деление, а также более сложные математические вычисления, что позволит расширить его функциональность и сделать его актуальным для более широкого круга пользователей.

Основные задачи, которые мы ставим перед собой в рамках данного проекта, включают анализ потребностей целевой аудитории, выбор подходящих технологий для разработки, создание удобного и привлекательного интерфейса, а также реализацию надежной и эффективной логики расчета. В ходе работы над проектом мы стремимся не только создать функциональное приложение, но и гарантировать его надежность и простоту в использовании.

Таким образом, разработка калькулятора не только обогатит мой опыт в области программирования и мобильной разработки, но и предоставит полезный инструмент, способствующий более удобному выполнению вычислений в повседневной жизни пользователей. В дальнейшем данное приложение может быть улучшено и дополнено новыми функциями, что сделает его еще более востребованным.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Разработка интерфейса пользователя (UI) является ключевым аспектом создания любого приложения, поскольку именно он определяет, как пользователи будут взаимодействовать с вашим продуктом. Этот процесс включает в себя несколько этапов, начиная с исследования и анализа потребностей пользователей и заканчивая созданием прототипов и визуальной частью интерфейса.

Первым шагом в разработке интерфейса является изучение целевой аудитории. Понимание кто ваши пользователи – это основа для создания интерфейса, который будет интуитивен и удобен в использовании. Вам следует провести опросы, фокус-группы или наблюдения за пользователями, чтобы собрать данные о том, какие функции им наиболее нужны и как они обычно взаимодействуют с аналогичными приложениями.

После того как вы соберете первичную информацию о ваших пользователях, следующим шагом будет создание пользовательских персонажей. Эти персонажи представляют собой обобщенные профили целевой аудитории и помогают направить ваш дизайн, учитывая различные потребности и сценарии использования. Персонажи позволят команде лучше понимать контекст, в котором будут использоваться приложения, а также поможет принимать более обоснованные решения в процессе разработки интерфейса.

Следующим этапом является создание пользовательских сценариев, которые описывают, как разные типы пользователей будут взаимодействовать с вашим приложением. Этот процесс включает в себя описание различных задач, которые пользователи должны выполнить, и шаг за шагом объясняет, как они зайдут в приложение, какие функции они будут использовать и каким образом будут достигать своих целей.

После того как у вас есть четкие пользовательские сценарии, можно переходить к созданию прототипа интерфейса. Прототип – это нечто среднее между эскизом и готовым интерфейсом; он позволяет визуализировать структуру приложения и основные элементы дизайна, такие как кнопки, формы и меню. В этом этапе можно использовать инструменты, такие как Figma, Sketch или Adobe XD для создания интерактивных прототипов. Интерактивные прототипы позволяют вашим пользователям тестировать интерфейс и предоставлять обратную связь на каждом этапе, что помогает выявить потенциальные проблемы или недочеты на раннем этапе.

После окончания тестирования прототипа и внесения всех необходимых исправлений, можно переходить к визуальному дизайну интерфейса. Это включает в себя выбор цветовой палитры, шрифтов и общей стилистики приложения. Важно, чтобы все элементы дизайна были согласованы и соответствовали общей концепции бренда. Пользовательский интерфейс должен быть не только красивым, но и функциональным, обеспечивая легкий доступ к информации и интуитивное взаимодействие с элементами управления.

На финальном этапе необходимо протестировать интерфейс, чтобы убедиться, что все элементы работают корректно на разных устройствах и разрешениях экранов.

Реализация логики калькулятора — это ключевой аспект разработки программного обеспечения, который позволяет пользователям выполнять арифметические и логические операции. В данном случае мы поговорим о создании базового калькулятора, который способен выполнять простые операции: сложение, вычитание, умножение и деление. При реализации логики калькулятора стоит уделить внимание основным компонентам, таким как обработка ввода пользователя, выполнение расчетов и отображение результатов.

Первый шаг в реализации логики калькулятора — это создание интерфейса для ввода данных. Пользователь должен иметь возможность вводить числа и выбирать операции. Это можно сделать с помощью кнопок и текстовых полей. Интерфейс должен быть интуитивно понятным, чтобы пользователи могли легко вводить свои данные и получать результаты. Например, текстовое поле может быть использовано для отображения ввода пользователя, а кнопки — для выбора арифметических операций.

После того как интерфейс готов, следует переходить к обработке ввода. На этом этапе важно создать функцию, которая будет принимать пользовательские данные и определять, какие операции необходимо выполнить. Это можно сделать с помощью базового условного оператора или конструкции switch-case. Например, если пользователь нажимает кнопку «+», необходимо сохранить текущее число, ожидать следующий ввод и затем выполнить сложение.

Далее необходимо реализовать функции для выполнения арифметических операций. Например, можно создать отдельные функции для каждой операции: для сложения, вычитания, умножения и деления. Каждая из этих функций будет принимать два аргумента (числа) и возвращать результат операции. Важно также учитывать обработку ошибок: например, при делении нельзя делить на ноль, поэтому стоит предусмотреть соответствующую проверку и уведомление пользователя о неправильно введенных данных.

Когда операции реализованы, следует подумать о выводе результатов. После того как пользователь закончил вводить данные и выбрал операцию, калькулятор должен отобразить результат в текстовом поле или в отдельной области интерфейса. Это позволяет пользователю быстро увидеть результат своих действий без лишних усилий.

Необходимо также предусмотреть возможность сброса ввода. Кнопка «Сброс» или «Очистить» должна очищать все данные, которые были введены пользователем, и возвращать калькулятор в исходное состояние. Это важно для создания удобного и эффективного взаимодействия.

Архитектура Android-приложений основана на многослойной модели, которая обеспечивает гибкость, масштабируемость и простоту в разработке. Android-приложение состоит из одного или нескольких компонентов, которые взаимодействуют друг с другом для выполнения задач.

Основными компонентами приложения являются Activity, Service, Broadcast Receiver и Content Provider. Activity — это основной элемент пользовательского интерфейса, который отображает экран и взаимодействует с пользователем. Каждая Activity представляет собой отдельный экран приложения. Service выполняет длительные операции в фоновом режиме без взаимодействия с пользователем, например, загрузку данных или воспроизведение музыки. Broadcast Receiver принимает и обрабатывает сообщения (broadcasts) от других приложений или системы, такие как уведомления о завершении загрузки. Content Provider управляет доступом к данным приложения и позволяет различным приложениям обмениваться данными, используя стандартные интерфейсы.

Каждый компонент Android имеет свой жизненный цикл, который управляет состоянием приложения. Например, Activity может находиться в состоянии "активно", "приостановлено" или "уничтожено". Понимание жизненного цикла компонентов важно для эффективного управления ресурсами и пользовательским интерфейсом.

Пользовательский интерфейс (UI) в Android состоит из различных элементов, таких как кнопки, текстовые поля и списки, которые можно комбинировать для создания интерфейса приложения. Эти элементы определяются в XML-файлах и могут быть динамически изменены с помощью кода.

Android Manifest — это файл конфигурации, который описывает основные характеристики приложения, такие как компоненты, разрешения, необходимые для работы приложения, и другие метаданные.

Для организации кода и упрощения разработки часто используются архитектурные паттерны, такие как Model-View-Controller (MVC) и Model-View-ViewModel (MVVM). Эти паттерны помогают отделить бизнес-логику от пользовательского интерфейса и делают приложение более поддерживаемым.

Кроме того, Android SDK предоставляет множество библиотек и инструментов для упрощения разработки, включая библиотеки для работы с сетевыми запросами, базами данных, анимацией и многим другим.

Понимание архитектуры Android-приложений является ключом к успешной разработке качественных и производительных приложений. Это знание помогает разработчикам эффективно управлять компонентами и ресурсами, а также создавать интуитивно понятные интерфейсы для пользователей.

Activity является одним из ключевых компонентов Android-приложений и представляет собой экран, с которым взаимодействует пользователь. Каждая Activity может содержать различные элементы пользовательского интерфейса, такие как кнопки, текстовые поля и изображения. Жизненный цикл Activity управляет состоянием приложения и определяет, как Activity реагирует на различные события, такие как запуск, приостановка или уничтожение.

Жизненный цикл Activity включает несколько основных методов. Метод onCreate() вызывается при создании Activity и отвечает за инициализацию пользовательского интерфейса и установку необходимых ресурсов. Затем вызывается onStart(), когда Activity становится видимой для пользователя, но еще не находится на переднем плане. После этого следует onResume(), когда Activity начинает активно взаимодействовать с пользователем.

Когда система начинает приостанавливать Activity, вызывается метод onPause(), который позволяет сохранить состояние приложения и освободить ресурсы. Затем, если Activity больше не видима, срабатывает onStop(), где можно высвободить те ресурсы, которые не требуются в данный момент. Наконец, перед уничтожением Activity вызывается метод onDestroy(), который используется для окончательной очистки ресурсов.

Понимание жизненного цикла Activity помогает разработчикам управлять состоянием приложения, обеспечивать плавный пользовательский интерфейс и эффективно использовать ресурсы устройства.

Layout в Android представляет собой способ организации элементов пользовательского интерфейса на экране, определяя, как различные компоненты, такие как кнопки, текстовые поля и изображения, располагаются и взаимодействуют друг с другом. Android предлагает несколько типов Layout, каждый из которых подходит для различных сценариев.

LinearLayout располагает элементы в одном направлении — вертикально или горизонтально, что делает его простым и удобным для организации интерфейса. RelativeLayout позволяет размещать элементы относительно друг друга или родительского контейнера, обеспечивая большую гибкость в дизайне. ConstraintLayout является более мощным и современным подходом, который позволяет создавать сложные интерфейсы с помощью ограничений, точно определяя, как элементы должны располагаться относительно друг друга. FrameLayout используется для отображения одного элемента на экране, который может накладываться на другие элементы, что полезно для создания простых интерфейсов.

Элементы пользовательского интерфейса определяются в XML-файлах, которые описывают, как элементы должны выглядеть и как они должны взаимодействовать с пользователем. Например, кнопки могут иметь различные стили и размеры, а текстовые поля могут иметь разные подсказки. Пользовательский интерфейс также может динамически изменяться с помощью кода на Java или Kotlin, что позволяет создавать интерактивные и адаптивные приложения.

Android SDK (Software Development Kit) представляет собой набор инструментов, библиотек и документации, необходимых для разработки Android-приложений. Он включает в себя компиляторы, отладчики и различные утилиты, которые позволяют разработчикам создавать приложения, тестировать их и развертывать на устройствах Android. Android SDK предоставляет доступ ко всем основным компонентам платформы, включая Activity, Services, Content Providers и Broadcast Receivers, что позволяет разработчикам эффективно использовать возможности Android.

SDK также включает в себя API для работы с различными функциями устройства, такими как камера, GPS, сенсоры и сеть. Это позволяет разработчикам интегрировать в свои приложения сложные функции, такие как геолокация, работа с мультимедиа и взаимодействие с интернет-сервисами. Регулярные обновления Android SDK обеспечивают доступ к новым функциям и улучшениям платформы, что позволяет разработчикам оставаться на переднем крае технологий.

Этот набор импортов является основой для создания пользовательского интерфейса в приложении Android. Он позволяет использовать различные компоненты и классы, необходимые для взаимодействия с пользователем и управления данными.

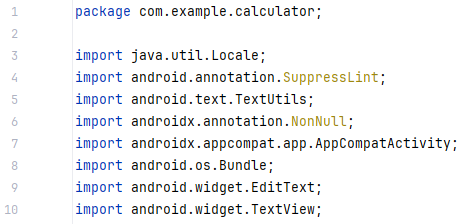


Рисунок 1 - библиотеки

Использование этих библиотек позволяет разработчикам значительно ускорить процесс разработки, улучшить производительность приложений и сосредоточиться на логике бизнес-приложений, вместо того чтобы тратить время на реализацию базовых математических функций. В коде используются несколько библиотек и классов, каждый из которых выполняет определенные функции в приложении Android.

import java.util.Locale предоставляет классы для работы с локалями, что позволяет управлять форматированием текста, чисел и дат в соответствии с языковыми и культурными стандартами. Это особенно полезно для приложений, которые поддерживают несколько языков или работают с данными, зависящими от региона.

import android.annotation.SuppressLint используется для подавления предупреждений компилятора. Например, если используется устаревшие методы или классы, которые могут вызвать предупреждения, можно применить аннотацию @SuppressLint, чтобы игнорировать эти предупреждения.

import android.text.TextUtils содержит утилитарные методы для работы со строками. Этот класс предоставляет функции для проверки строк на пустоту, объединения строк и других операций, которые упрощают работу с текстовыми данными в вашем приложении.

import androidx.annotation.NonNull — это аннотация, которая указывает, что параметр или возвращаемое значение не должно быть равно null. Использование этой аннотации помогает улучшить читаемость кода и позволяет инструментам статического анализа выявлять потенциальные проблемы с null-значениями.

import androidx.appcompat.app.AppCompatActivity — это класс, который расширяет функциональность стандартной Activity в Android, предоставляя поддержку современных интерфейсов и компонентов, таких как ActionBar и темы Material Design. Использование AppCompatActivity позволяет вашему приложению выглядеть более современно и совместимо с различными версиями Android.

import android.os.Bundle используется для передачи данных между компонентами Android, такими как Activity. Bundle позволяет сохранять состояние приложения, передавать параметры между Activity и хранить данные, которые должны быть восстановлены при перезапуске приложения.

import android.widget.EditText представляет собой элемент пользовательского интерфейса, который позволяет пользователям вводить текст. Этот класс предоставляет функциональность для работы с текстовыми полями, включая обработку ввода и получение текста, введенного пользователем.

import android.widget.TextView — это элемент интерфейса, который отображает текст на экране. TextView позволяет отображать статический текст или изменять его в зависимости от взаимодействия пользователя с приложением.

import android.widget.Toast используется для отображения коротких сообщений пользователю в виде всплывающих уведомлений. Toast позволяет показывать сообщения на экране, которые исчезают через несколько секунд, что делает его удобным для отображения информации, не требующей взаимодействия пользователя.

# 

# ПРАКТИЧЕСКАЯ ЧАСТЬ

Данный класс представляет собой основу для калькулятора, где пользователи могут вводить числа, выполнять операции и видеть результаты. Это создаст функциональное приложение, которое будет взаимодействовать с пользователем и предоставлять ему необходимые вычисления.

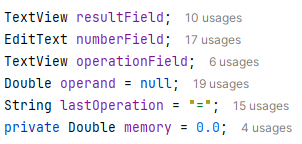


Рисунок 2 - класс

В методе onCreate инициализируется интерфейс калькулятора и связываются элементы управления с соответствующими переменными. Устанавливаются обработчики кликов для кнопок операций, чисел и памяти, которые вызывают соответствующие методы для выполнения действий, таких как сложение, вычитание, вычисление корня и управление памятью. Это создает интерактивный интерфейс для пользователя.



Рисунок 3 - onCreate

Методы onSaveInstanceState и onRestoreInstanceState сохраняют и восстанавливают состояние активности, сохраняя текущую операцию и операнд в Bundle.

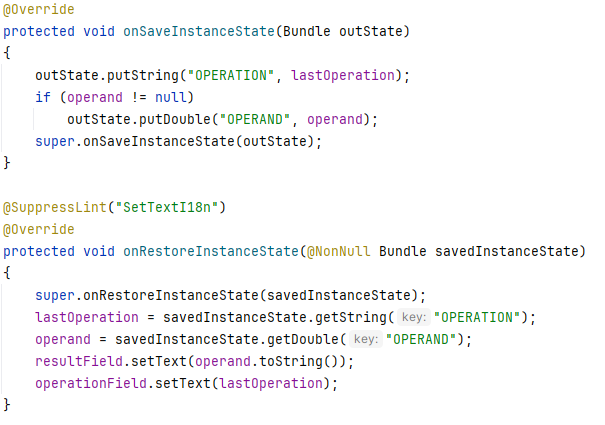


Рисунок 4 - onSaveInstanceState и onRestoreInstanceState

Метод onNumberClick добавляет нажатую цифру или запятую в поле ввода, при этом предотвращая ввод нескольких запятых подряд, а также сбрасывает операнд, если предыдущая операция была равенством. Метод onOperationClick получает введённое число, проверяет, что оно не пустое, заменяет запятую на точку для корректного преобразования в число и вызывает performOperation с этим числом и выбранной операцией. В случае ошибки ввода очищает поле, а затем обновляет текущую операцию и отображает её.



Рисунок 5 - onNumberClick и performOperation

Метод performOperation выполняет арифметические операции над числом и обновляет результат. Если операнд ещё не установлен, он принимает значение текущего числа. Если последняя операция была равенством, она обновляется на новую. В зависимости от последней операции выполняется соответствующее вычисление. Если происходит деление на ноль, выводится ошибка и очищаются поля. Результат отображается с двумя знаками после запятой, а поле ввода очищается для следующего числа.



Рисунок 6 - performOperation

Методы memoryAdd и memorySubtract предназначены для работы с памятью калькулятора. В первом методе, memoryAdd, сначала считывается текущее значение из поля ввода. Если это значение не пустое, оно преобразуется в число и добавляется к текущему значению памяти. После этого вызывается метод clearFields, который очищает поле ввода, и пользователю отображается уведомление о том, что число успешно добавлено в память. Если же поле ввода пустое, выводится сообщение с просьбой ввести число для добавления.

Во втором методе, memorySubtract, процесс аналогичен. Сначала считывается значение из поля ввода. Если оно не пустое, то это значение вычитается из текущего значения памяти. После выполнения операции также очищается поле ввода, и пользователю показывается уведомление о том, что число вычтено из памяти. Если поле ввода пустое, выводится сообщение, призывающее пользователя ввести число для вычитания.

Таким образом, оба метода обеспечивают удобное и безопасное взаимодействие с памятью калькулятора, информируя пользователя о результатах операций и предотвращая ошибки, связанные с пустым вводом.

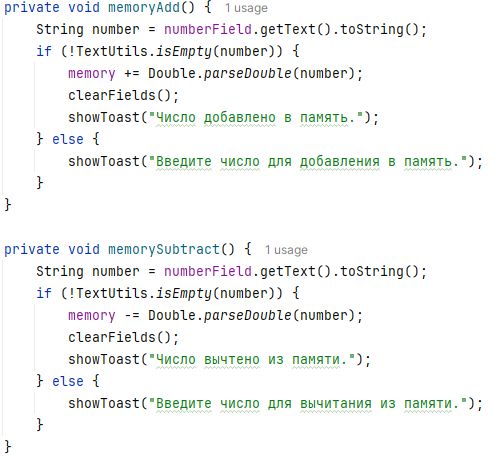


Рисунок 7 - memoryAdd и memorySubtract

Метод memoryClear отвечает за очистку памяти калькулятора. При вызове этого метода переменная memory, которая хранит значение в памяти, устанавливается в 0.0. Это позволяет пользователю сбросить сохраненное значение, что особенно важно в контексте калькулятора, поскольку пользователи могут захотеть начать новые вычисления без влияния предыдущих значений. После очистки памяти вызывается метод showToast(), который отображает уведомление пользователю о том, что память была очищена. Это улучшает взаимодействие с пользователем, позволяя ему быть в курсе действий, происходящих в приложении.

Метод memoryRecall предназначен для восстановления значения из памяти калькулятора. Если значение в памяти не равно 0.0, оно отображается в поле результата (resultField), что позволяет пользователю быстро использовать ранее сохраненное значение в новых вычислениях. Если же память пуста, вызывается showToast(), который уведомляет пользователя об этом. Это также способствует улучшению пользовательского опыта, так как предотвращает путаницу и ошибки при попытке использовать пустую память.



Рисунок 8 - memoryClear и memoryRecall

Метод showToast используется для отображения коротких уведомлений пользователю. Он принимает строку с сообщением и создает всплывающее уведомление, которое исчезает через короткий промежуток времени. Это позволяет быстро информировать пользователя о состоянии приложения, например, о том, что память была очищена или что она пуста.

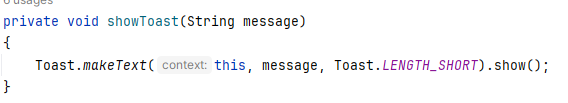


Рисунок 9 - showToast

Метод calculateSqrt предназначен для вычисления квадратного корня из числа, введённого пользователем в поле numberField. Сначала метод проверяет, не является ли поле ввода пустым. Если оно пустое, выполнение метода просто прекращается, и никакие действия не выполняются.

Далее, если поле не пустое, происходит замена запятых на точки, чтобы обеспечить корректное преобразование строки в число. Затем метод пытается преобразовать строку в число типа double. Если преобразование не удаётся, выбрасывается исключение NumberFormatException, и поле ввода очищается.

Если преобразование прошло успешно, метод проверяет, не является ли значение отрицательным. Если значение меньше нуля, в поле результата (resultField) выводится сообщение об ошибке, указывающее на то, что нельзя извлечь квадратный корень из отрицательного числа. После этого поле ввода очищается, и выполнение метода завершается.

Если значение положительное, вычисляется квадратный корень с помощью функции Math.sqrt. Результат округляется до двух десятичных знаков и отображается в поле результата. Поле ввода также очищается, а переменная operand обновляется значением квадратного корня. Переменная lastOperation устанавливается в «=», и это значение отображается в поле операций (operationField).

Таким образом, метод обеспечивает безопасное и корректное вычисление квадратного корня, информируя пользователя о возможных ошибках и выводя результат в удобочитаемом формате.



Рисунок 10 - calculateSqrt

Метод calculateInverse вычисляет обратное значение числа, введённого пользователем в поле numberField. Сначала он проверяет, пусто ли поле ввода, и если да, то просто завершает выполнение. Если поле не пустое, метод заменяет запятые на точки, чтобы корректно преобразовать строку в число. Затем происходит попытка преобразования строки в число типа double. Если преобразование не удаётся, вызывается исключение NumberFormatException, и поле ввода очищается. Если число успешно получено, метод проверяет, не равно ли оно нулю. В случае нуля в поле результата появляется сообщение об ошибке, что деление на ноль невозможно, после чего поле ввода очищается и выполнение метода завершается. Если число отлично от нуля, вычисляется его обратное значение, которое округляется до двух десятичных знаков и отображается в поле результата. После этого поле ввода очищается, переменная operand обновляется новым значением, а переменная lastOperation устанавливается в знак равенства, который отображается в поле операций. Таким образом, метод обеспечивает корректное вычисление обратного значения с обработкой ошибок и информированием пользователя.

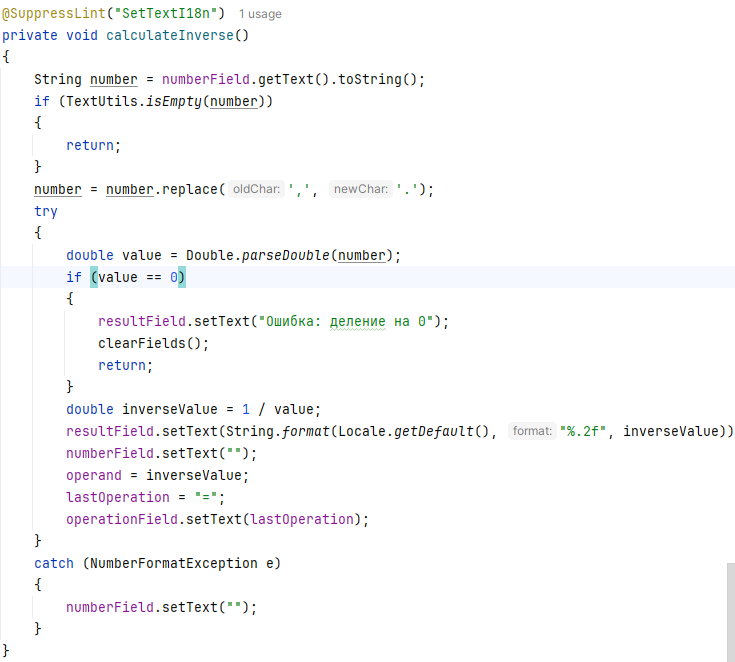


Рисунок 11 - calculateInverse

Метод deleteLastCharacter отвечает за удаление последнего символа из текста, введённого пользователем в поле numberField. Сначала метод получает текущий текст из поля. Если текст не пустой, он обновляет содержимое поля, удаляя последний символ с помощью метода substring. Это позволяет пользователю корректировать ввод, удаляя ненужные цифры или символы.

Метод clearFields предназначен для очистки всех полей ввода и вывода. Он устанавливает текст в поле numberField и resultField на пустую строку, а также сбрасывает переменные operand и lastOperation. Переменная lastOperation устанавливается в знак равенства, который затем отображается в поле операций (operationField). Это обеспечивает удобство работы с калькулятором, позволяя пользователю начинать новый ввод без остаточных данных.



Рисунок 12 - deleteLastCharacter и clearFields

В интерфейсе приложения «Калькулятор», разработанного с использованием Android. Интерфейс создан с применением XML и фреймворка ConstraintLayout, что обеспечивает адаптивность и удобство при использовании на различных устройствах.

Интерфейс калькулятора состоит из нескольких ключевых элементов, которые обеспечивают пользователю простоту и удобство в работе с приложением. В верхней части экрана расположены два TextView: один из них, с ID resultField, отображает результат вычислений, а второй, с ID operationField, показывает текущую математическую операцию. Эти элементы имеют достаточный размер шрифта, что позволяет легко читать информацию.



Рисунок 13 - TextView

Под полем для результата находится EditText с ID numberField, который предназначен для ввода чисел. Этот элемент настроен на прием чисел, что делает процесс ввода более удобным. Ввод данных осуществляется с помощью виртуальной клавиатуры, а также через кнопки на экране, что обеспечивает гибкость в использовании.

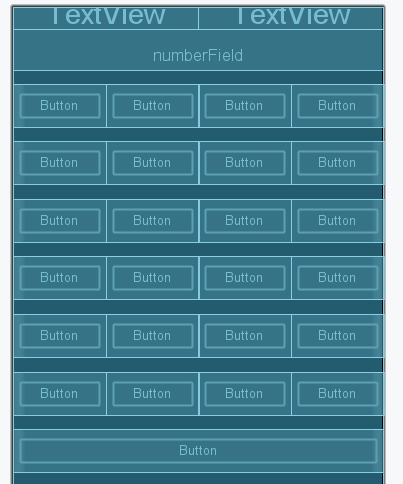


Рисунок 14 - EditText

Кнопки калькулятора организованы в несколько панелей, каждая из которых отвечает за определенные функции.

В данном фрагменте кода представлена панель кнопок, используемая в Android-приложении в калькуляторе. Эта панель разделена на четыре кнопки, каждая из которых выполняет свою функцию. Элемент <LinearLayout> используется для создания горизонтального списка дочерних элементов. В данном случае, это горизонтальная панель, содержащая четыре кнопки. У панели есть уникальный идентификатор android:id="@+id/zeroButtonPanel", который позволяет обращаться к ней в коде. Ширина панели установлена в 0dp, что позволяет использовать вес для распределения пространства, а высота автоматически подстраивается под содержимое благодаря android:layout\_height="wrap\_content". Отступ сверху, равный 16dp, отделяет панель от других элементов интерфейса.



Рисунок 15 - панель

Каждая кнопка внутри панели имеет ширину 0dp, что также позволяет равномерно распределять пространство между кнопками, и высоту, автоматически подстраивающуюся под текст. Поскольку у нас четыре кнопки, каждая из них будет занимать 1/4 доступного пространства.

Таким образом, каждая панель в этом интерфейсе разделена на четыре кнопки, каждая из которых выполняет определенные функции. Использование layout\_weight позволяет равномерно распределять пространство между кнопками, создавая аккуратный и удобный интерфейс для пользователя.

Первая панель кнопок памяти включает кнопки для операций памяти: M+, M-, MC и MR. Эти кнопки выделены ярким желтым цветом, что делает их легко заметными и интуитивно понятными для пользователей, часто использующих функции памяти.



Рисунок 16 - первая панель

Вторая панель чисел включает кнопки 7, 8, 9 и кнопку деления, расположенные слева направо. Третья панель содержит кнопки 4, 5, 6 и кнопку умножения. Четвертая панель включает кнопки 1, 2, 3 и кнопку вычитания. Пятая панель содержит кнопку для ввода запятой, кнопку 0, кнопку сложения и кнопку очистки ввода. Все кнопки на этих панелях выполнены в одном цвете и имеют одинаковый размер, что способствует единству дизайна и облегчает восприятие.

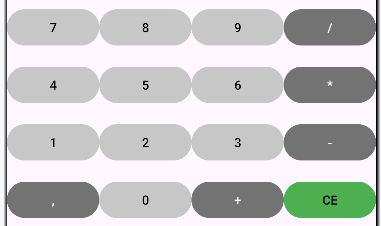


Рисунок 17 - основные кнопки

Шестая панель содержит кнопки для вычисления квадратного корня, процента и обратного числа, а также кнопку полной очистки. Цвет кнопок на этой панели синий и зеленой, чтобы выделять их среди остальных и подчеркивать их функциональность. Кнопка равенства выделена более темным оттенком, например, темно-зеленым, чтобы подчеркнуть ее важность в завершении вычислений. Размещение этой панели в нижней части интерфейса позволяет пользователям легко к ней добираться, не отвлекаясь от основных вычислений, что улучшает общую навигацию по приложению и делает его более интуитивным.

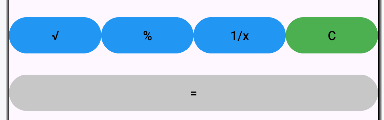


Рисунок 18 - шестая панель

Весь интерфейс калькулятора выполнен в современном стиле, с использованием различных оттенков для кнопок, что делает его визуально привлекательным и легким для восприятия. Цветовая палитра выбрана так, чтобы обеспечить хорошую видимость кнопок и их функциональности. Адаптивный дизайн и логичное расположение элементов позволяют пользователю интуитивно ориентироваться в приложении и быстро выполнять необходимые вычисления.

# ЗАКЛЮЧЕНИЕ

В заключение отчета о моей учебной практике следует отметить, что выполненная работа по разработке калькулятора оказалась весьма продуктивной и насыщенной. В ходе практики я успешно освоил ключевые аспекты программирования, включая проектирование пользовательского интерфейса и реализацию основных арифметических операций, что является важным навыком в области разработки программного обеспечения.

Процесс разработки позволил мне на практике применить теоретические знания, полученные в учебном заведении. Я столкнулся с различными техническими вызовами, что способствовало углублению понимания принципов работы программ и важности тестирования. Особенно полезным был опыт по исправлению ошибок и оптимизации программы, что подчеркнуло необходимость тщательной проверки всех компонентов системы.

Кроме того, взаимодействие с командой разработчиков позволило мне развить навыки коммуникации и учиться работать в условиях совместной деятельности. Это взаимодействие было важным для понимания важности командной работы и распределения задач для достижения общего результата.

Обобщая, можно сказать, что опыт, который я получил в ходе своей учебной практики, является неоценимым. Он не только способствовал углублению профессиональных навыков, но и открыл новые горизонты для дальнейшего обучения и развития в сфере информационных технологий. Я уверен, что знания и практический опыт, полученные в ходе этой практики, будут полезны в моей будущей профессиональной деятельности.