СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc185675704)

[1. Постановка задачи и анализ предметной области 4](#_Toc185675705)

[1.1 Постановка задачи 4](#_Toc185675706)

[1.2 Анализ предметной области 4](#_Toc185675707)

[1.2.1 Вычисление процентов 4](#_Toc185675708)

[1.2.2 Вычисление площади квадрата 5](#_Toc185675709)

[1.2.3 Пользовательский интерфейс 6](#_Toc185675710)

[1.2.4 Тестирование и отладка 6](#_Toc185675711)

[2. Разработка технического задания 7](#_Toc185675712)

[2.1 Основания для разработки приложения 7](#_Toc185675713)

[2.2 Назначение приложения 7](#_Toc185675714)

[2.2.1 Функциональное назначение 7](#_Toc185675715)

[2.2.2 Эксплуатационное назначение 7](#_Toc185675716)

[2.3 Требования к приложению 7](#_Toc185675717)

[2.3.1 Требования к функциональным характеристикам 7](#_Toc185675718)

[2.3.2 Требования к надёжности 7](#_Toc185675719)

[2.3.3 Время восстановления после отказа 7](#_Toc185675720)

[2.3.4 Отказы из-за некорректных действий оператора 8](#_Toc185675721)

[2.3.5 Условия эксплуатации 8](#_Toc185675722)

[2.3.6 Требования к видам обслуживания 8](#_Toc185675723)

[2.3.7 Требования к составу и параметрам технических средств 8](#_Toc185675724)

[2.3.8 Требования к информационной и программной совместимости 8](#_Toc185675725)

[2.3.9 Требования к программной документации 9](#_Toc185675726)

[2.3.10 Технико-экономические показатели 9](#_Toc185675727)

[2.4 Паспорт модуля 9](#_Toc185675728)

[2.4.1 Функции программы 9](#_Toc185675729)

[3. Проектирование 13](#_Toc185675730)

[3.1 Поведенческие диаграммы 13](#_Toc185675731)

[3.1.1 Диаграмма вариантов использования 13](#_Toc185675732)

[3.1.2 Диаграмма последовательности 13](#_Toc185675733)

[3.1.3 Диаграмма деятельности 14](#_Toc185675734)

[3.1.4 Диаграмма состояний 15](#_Toc185675735)

[3.2 Структурные диаграммы 17](#_Toc185675736)

[3.2.1 Диаграмма классов 17](#_Toc185675737)

[3.2.2 Диаграмма компонентов 18](#_Toc185675738)

[3.2.3 Диаграмма коопераций 18](#_Toc185675739)

[3.2.4 Диаграмма развёртывания 19](#_Toc185675740)

[3.3 Модель С4 19](#_Toc185675741)

[3.3.1 Уровень контекста 20](#_Toc185675742)

[3.3.2 Уровень контейнеров 20](#_Toc185675743)

[3.3.3 Уровень компонентов 21](#_Toc185675744)

[3.3.4 Уровень кода 22](#_Toc185675745)

[4. Тестирование программного обеспечения 23](#_Toc185675746)

[4.1 Объект испытаний 23](#_Toc185675747)

[4.2 Цель испытаний 23](#_Toc185675748)

[4.2.1 Основание для проведения испытаний 23](#_Toc185675749)

[4.2.2 Место и продолжительность испытаний 23](#_Toc185675750)

[4.2.3 Перечень документов, предъявляемых на испытании 23](#_Toc185675751)

[4.2.4 Объём испытаний 23](#_Toc185675752)

[4.3 Требования к программе 23](#_Toc185675753)

[4.3.1 Требования к программной документации 23](#_Toc185675754)

[4.3.2 Технические средства, используемые во время испытаний 24](#_Toc185675755)

[4.4 Методика испытаний 24](#_Toc185675756)

[4.5 Результаты предварительных испытаний 25](#_Toc185675757)

[ЗАКЛЮЧЕНИЕ 27](#_Toc185675758)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc185675759)

[ПРИЛОЖЕНИЕ 29](#_Toc185675760)

# ВВЕДЕНИЕ

Разработка приложения "Простой калькулятор" представляет собой интересный и полезный проект, который позволит пользователям выполнять базовые арифметические операции, такие как сложение, вычитание, умножение и деление.

Цель данной практической работы заключается в создании интуитивно понятного интерфейса, который обеспечит удобное взаимодействие с приложением и позволит пользователям быстро и точно выполнять необходимые вычисления.

Это приложение станет незаменимым инструментом для студентов, бухгалтеров, инженеров и всех, кто сталкивается с необходимостью быстро выполнять математические вычисления.

В процессе работы над проектом мы также уделим внимание обработке ошибок, чтобы гарантировать стабильность и надежность приложения.

Данное приложение будет иметь такие функции как:

* Деление;
* Умножение;
* Суммирование;
* Очистка.

Мы рассмотрим основы проектирования пользовательского интерфейса, а также реализуем логику работы калькулятора, обеспечивающую корректное выполнение арифметических операций.

# 1. Постановка задачи и анализ предметной области

# 1.1 Постановка задачи

Цель практики заключается в сборе требований к программному обеспечению, применении методов для разработки кода с заданной функциональностью и уровнем качества, а также в планировании и проведении тестирования. Для достижения этих целей необходимо решить следующие задачи:

- Сформулировать техническое задание;

- Разработать приложение;

- Создать диаграммы для визуализации процессов;

- Провести тестирование программного обеспечения.

# 1.2 Анализ предметной области

Целью данной практики является создание программы, предназначенной для вычисления суммы чисел вводимых пользователем, а также суммы деления и умножения. Эти функции могут применяться в различных сферах жизни и могут быть полезны как в образовательных целях, так и в практических задачах.

# 1.2.1 Вычисление суммы

Функция вычисления суммы чисел позволяет пользователю легко определить общую сумму двух чисел. Это может быть полезно в различных ситуациях, таких как:

* Оценка общего бюджета, расчёт итоговой стоимости покупок и расходов;
* Использование в учебных целях для объяснения основ арифметики и её применения в повседневной жизни.

Эта функция требует от пользователя ввода двух чисел, после чего программа производит необходимые вычисления и предоставляет результат.

# 1.2.2 Вычисление суммы деления

Функция вычисления частного позволяет пользователю быстро и удобно определить результат деления одного числа на другое. Она может быть полезна в различных ситуациях, таких как:

* Анализ соотношений и пропорций, расчёт средних значений и статистических данных;
* Помощь в финансовом планировании, определение долей и распределение бюджета;
* Использование в образовательных целях для объяснения принципов деления и его практического применения.

Для использования этой функции пользователю необходимо ввести делимое и делитель, после чего программа выполнит необходимые вычисления и предоставит результат.

### 1.2.3 Вычисление суммы умножения

Функция вычисления произведения позволяет пользователю легко определить результат умножения двух чисел. Это имеет множество практических приложений, таких как:

* Определение общего количества при умножении групп объектов;
* Использование в финансовом анализе для расчета прибыли и убытков;
* Применение в образовательных целях для изучения основ арифметики и понимания принципов умножения.

Для использования этой функции пользователю необходимо ввести два числа, которые нужно умножить, после чего программа выполнит вычисления и отобразит результат произведения.

# 1.2.4 Пользовательский интерфейс

При разработке приложения «Простой калькулятор» важно акцентировать внимание на создании удобного и понятного интерфейса. Основные моменты, которые следует учесть:

* Все основные операции — сложение, вычитание, умножение и деление — должны быть представлены в виде легко узнаваемых кнопок, что позволит пользователям быстро находить нужные функции.
* Поля ввода должны быть четкими и доступными, чтобы пользователи могли без труда вводить числа и выбирать операции
* Результаты вычислений необходимо выводить в понятной форме, чтобы пользователи могли легко их интерпретировать и применять в дальнейших расчетах.

Соблюдение этих принципов поможет создать эффективный и удобный инструмент, который позволит пользователям сосредоточиться на решении математических задач без лишних усилий.

# 1.2.5 Тестирование и отладка

После завершения разработки приложения «Простой калькулятор» необходимо провести тестирование всех функций, чтобы убедиться в их корректности и надежности.

Тестирование должно включать:

* Проверку всех возможных сценариев использования функций;
* Оценку производительности приложения.

Таким образом, приложение «Простой калькулятор» станет не только полезным инструментом для выполнения математических вычислений, но и удобным в использовании, что повысит его привлекательность для пользователей.

# 2. Разработка технического задания

# 2.1 Основания для разработки приложения

Основанием является задание на практику.

# 2.2 Назначение приложения

# 2.2.1 Функциональное назначение

Функциональное назначение программы заключается в предоставлении инструмента, который будет полезен как в учебных, так и в профессиональных целях. Программа будет способствовать более глубокому пониманию математики и улучшению практических навыков пользователей.

# 2.2.2 Эксплуатационное назначение

Приложение не имеет специальных эксплуатационных назначений, однако должно обеспечивать стабильную работу в своих задачах.

# 2.3 Требования к приложению

# 2.3.1 Требования к функциональным характеристикам

Программа должна обеспечивать корректный расчёт суммы чисел, деления и умножения.

# 2.3.2 Требования к надёжности

Для обеспечения надёжного функционирования программы должны быть выполнены следующие требования:

- защита от некорректного ввода данных;

- обработка отрицательных и нулевых значений;

- правильность вычислений.

# 2.3.3 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания или другими внешними факторами, не должно превышать 20 минут при условии соблюдения условий эксплуатации технических и программных средств. В случае фатального сбоя (краха) операционной системы время восстановления должно соответствовать времени, необходимому для устранения неисправностей и переустановки программного обеспечения.

# 2.3.4 Отказы из-за некорректных действий оператора

Отказы программы могут происходить из-за некорректных действий пользователя. Для предотвращения таких отказов необходимо обеспечить проверку заданных пользователем значений.

# 2.3.5 Условия эксплуатации

Программа не требует особых условий эксплуатаций.

# 2.3.6 Требования к видам обслуживания

Программа не требует регулярного обслуживания.

# 2.3.7 Требования к составу и параметрам технических средств

Для работы программы необходим персональный компьютер, соответствующий следующим требованиям:

* **Процессор:** AMD Athlon 3000G (2.0 ГГц) и более
* **ОЗУ:** 2 Гб и более
* **Необходимо место на жестком диске:** 50 Мб
* **Видеоадаптер:** NVIDIA GeForce GTX 750 Ti и лучше
* **Экран:** 1920 х 1080 и выше

# 2.3.8 Требования к информационной и программной совместимости

Программа должна обеспечивать отображение входных данных и результатов на экране.

Исходные коды программы должны быть написаны на языке C#. Для разработки следует использовать среду Microsoft Visual Studio 2022.

## Требования к программным средствам, используемым программой

Программа должна работать на лицензионной локализованной версии операционной системы Windows 7, 8, 8.1, 10 с платформой .NET 8.0.

## 

## Требования к защите информации и программ

К требованиям по защите информации и программ не предъявляются специальные требования.

Программа должна обеспечивать взаимодействие с пользователем через графический пользовательский интерфейс, разработанный в соответствии с рекомендациями.

# 2.3.9 Требования к программной документации

Программная документация должна включать:

* техническое задание на разработку;
* руководство пользователя;
* программа, методика и результаты испытаний.

# 2.3.10 Технико-экономические показатели

Коммерческое использование приложения не предусмотрено.

# 2.4 Паспорт модуля

# 2.4.1 Функции программы

Программа предназначена для выполнения математических расчетов. Основные функции программы реализованы в главном окне (MainWindow).

Основные функции:

* Вычисление суммы чисел:

Пользователь вводит числа в текстовые поля (TEXT1 и TEXT2).

Кнопка "СУММА" (SUMMA): При нажатии кнопки происходит проверка введенных данных. Если данные корректны, вызывается метод raschet.sum, который вычисляет сумму чисел. На рисунке 1 показано данное действие.

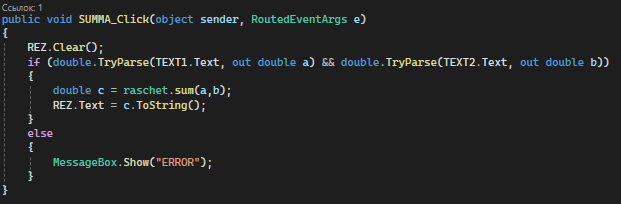


Рисунок 1 – Нажатие на кнопку «СУММА»

Результат может выводиться с дробями и выводится в текстовое поле REZ. В случае некорректного ввода отображается сообщение об ошибке.

* Вычисление суммы деления:

Пользователь вводит числа в текстовые поля (TEXT1 и TEXT2).

Кнопка "ДЕЛЕНИЕ" (DELEN): При нажатии кнопки происходит проверка введенных данных. Если данные корректны, вызывается метод raschet.del, который вычисляет сумму деления. На рисунке 2 показано данное действие.

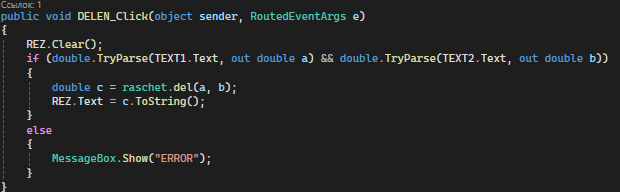


Рисунок 2 – Нажатие на кнопку «ДЕЛЕНИЕ»

Результат выводится в текстовое поле REZ. В случае некорректного ввода также отображается сообщение об ошибке.

* Вычисление суммы умножения:

Пользователь вводит числа в текстовые поля (TEXT1 и TEXT2).

Кнопка "УМНОЖЕНИЕ" (UMN): При нажатии кнопки происходит проверка введенных данных. Если данные корректны, вызывается метод raschet.umn, который вычисляет сумму умножения. На рисунке 3 показано данное действие.

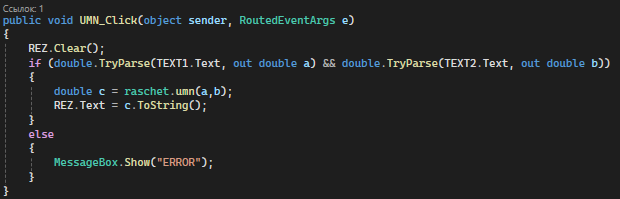


Рисунок 3 – Нажатие на кнопку «УМНОЖЕНИЕ»

Результат выводится в текстовое поле REZ. В случае некорректного ввода также отображается сообщение об ошибке.

* Очистка данных:

Кнопка "ОЧИСТКА" (CLEAR): В главном окне имеется возможность очистки всех текстовых полей, что позволяет пользователю вводить новые данные без необходимости перезапуска приложения. На рисунке 4 показано данное действие.

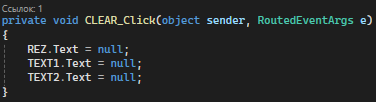


Рисунок 4 – Нажатие на кнопку «ОЧИСТКА»

* Вспомогательный функционал программы:

Кнопка "ВЫХОД" (EXIT): Закрывает текущее окно и завершает работу приложения. На рисунке 5 показано данное действие.

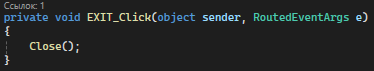


Рисунок 5 – Нажатие на кнопку «Расчёт»

Кнопка "О ПРОГРАММЕ" (ABOUT): Отображает информацию о разработчике приложения в виде сообщения. На рисунке 6 показано данное действие.

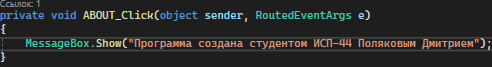


Рисунок 6 – Нажатие на кнопку «О ПРОГРАММЕ»

# 3. Проектирование

# 3.1 Поведенческие диаграммы

# 3.1.1 Диаграмма вариантов использования

Диаграмма вариантов использования (или диаграмма прецедентов, user-case диаграмма) — это тип диаграммы в UML (Unified Modeling Language), который визуально описывает взаимодействие между пользователями (или "операторами") и системой. Она иллюстрирует функциональные требования системы, показывая, какие действия могут выполнять пользователи, а также какие результаты они могут ожидать от системы. На рисунке 7 показано ДВИ к данной программе.

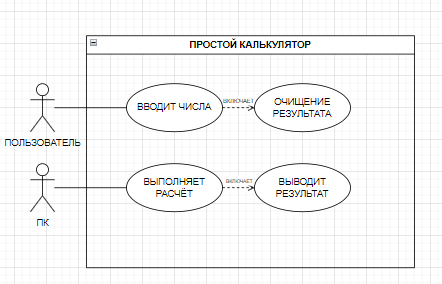


Рисунок 7 – Диаграмма вариантов использования

# 3.1.2 Диаграмма последовательности

Диаграмма последовательности (sequence diagram) — это тип диаграммы в UML (Unified Modeling Language), который визуально описывает взаимодействие между объектами в системе в рамках определенного сценария (варианта использования) с акцентом на временные аспекты этих взаимодействий. На рисунке 8 показана диаграмма последовательности к данной программе.

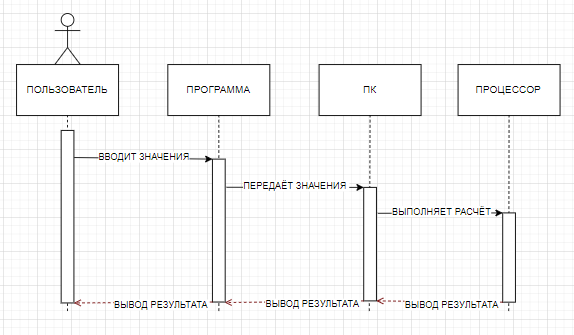


Рисунок 8 – Диаграмма последовательности

# 3.1.3 Диаграмма деятельности

Диаграмма состояния (state diagram) — это тип диаграммы в UML (Unified Modeling Language), который описывает различные состояния объекта и переходы между этими состояниями в ответ на события. Она визуально представляет жизненный цикл объекта, показывая, как он изменяет свое состояние в зависимости от внутренних и внешних факторов. Диаграмма деятельности показана на рисунке 9.

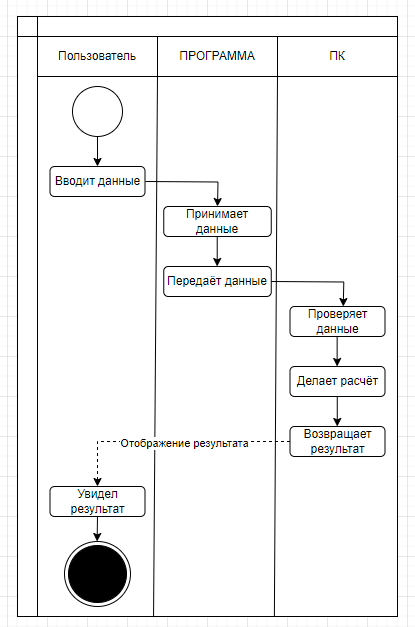


Рисунок 9 – Диаграмма деятельности

# 3.1.4 Диаграмма состояний

Диаграмма состояния (state diagram) — это поведенческая диаграмма в UML (Unified Modeling Language), которая описывает все возможные состояния системы или объекта, а также переходы между этими состояниями в ответ на определенные события. Основная цель диаграммы состояния — графически представить жизненный цикл объекта, отображая все этапы его существования и взаимодействия с другими элементами системы.

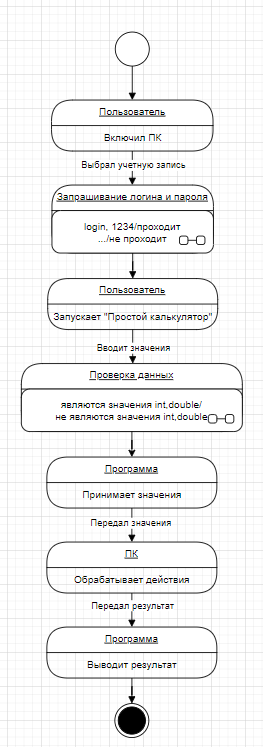


Рисунок 10 – Диаграмма состояний

# 3.2 Структурные диаграммы

# 3.2.1 Диаграмма классов

Диаграмма классов в языке моделирования UML (Unified Modeling Language) — это структурная диаграмма, предназначенная для визуализации структуры классов в системе, их атрибутов, методов, интерфейсов и отношений между ними. Она служит для описания статических аспектов системы, таких как классы, их свойства и поведение, а также динамических аспектов, включая связи между объектами и выполнение методов в процессе работы программы. Диаграмма классов показана на рисунке 11.

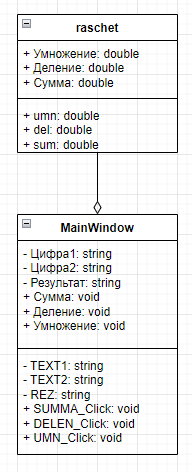


Рисунок 11 – диаграмма классов

# 3.2.2 Диаграмма компонентов

Диаграмма компонентов — это структурная диаграмма, которая описывает архитектуру системы, отображая взаимосвязи между ее компонентами. Она фокусируется на обобщенных элементах, таких как классы, интерфейсы, базы данных и серверы, предоставляя более широкий обзор системы по сравнению с диаграммой классов. Диаграмма компонентов может включать аннотации и описания, уточняющие взаимодействия между компонентами, используемые интерфейсы, а также дополнительную информацию о компонентах, включая технологии, языки программирования и фреймворки. Эта диаграмма помогает разработчикам и архитекторам лучше понять структуру системы и ее взаимосвязи, а также служит инструментом для документирования архитектурных решений. Диаграмма компонентов отображена на рисунке 12.

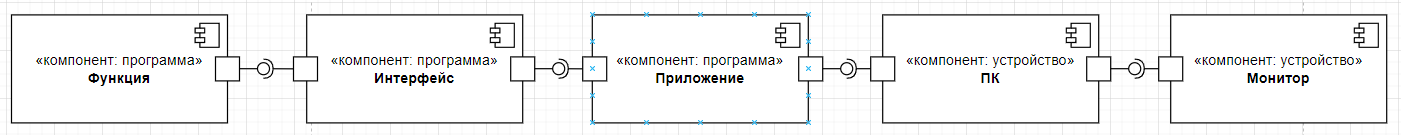


Рисунок 12 – диаграмма компонентов

# 3.2.3 Диаграмма коопераций

Диаграмма кооперации — это структурная диаграмма, предназначенная для описания поведения системы на уровне взаимодействия самостоятельных объектов. Эти объекты могут обмениваться сообщениями для достижения поставленных задач. Диаграмма кооперации может пересекаться с диаграммой вариантов использования, особенно когда рассматривается взаимодействие без привязки ко времени. В общем, она представляет собой совокупность объектов, актеров и их взаимодействий, отражая объединение свойств системы и подчеркивая, как объекты работают вместе для выполнения определенных функций. Диаграмма коопераций отображена на рисунке 13.

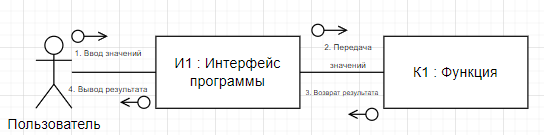


Рисунок 13 – диаграмма коопераций

# 3.2.4 Диаграмма развёртывания

Диаграмма развертывания — это представление архитектуры исполнения системы, которое включает узлы, такие как аппаратные и программные среды исполнения, а также промежуточное программное обеспечение. Она показывает, как компоненты системы развертываются на физических устройствах и как они взаимодействуют друг с другом через соединительное программное обеспечение. Диаграмма развертывания помогает понять, как аппаратное обеспечение и программные компоненты работают вместе, и подчеркивает важность соединительного программного обеспечения для функционирования системы в целом. Диаграмма развёртывания отображена на рисунке 14.

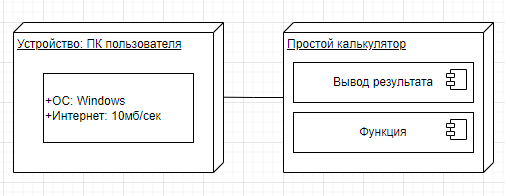


Рисунок 14 – Диаграмма развёртывания

# 3.3 Модель С4

Модель С4 (от англ. С4 model, Context-Container-Component-Code model, модель «контекст-контейнер-компонент-код») – это архитектурная модель для проектирования системы. Работает по принципу масштабирования картинки. То есть, увеличиваем масштаб - увеличиваем детализацию схемы.

# 3.3.1 Уровень контекста

Этот этап «поверхностно» описывает систему для пользователя, которому не нужно вдаваться в технические подробности системы, показан на рисунке 15.



Рисунок 15 – Уровень контекста

# 3.3.2 Уровень контейнеров

Следующий уровень необходим для описания системы с точки зрения архитектуры приложения. Техническая часть при этом снова не затрагивается. Данная часть наиболее практична для написания документации, показан на рисунке 16.

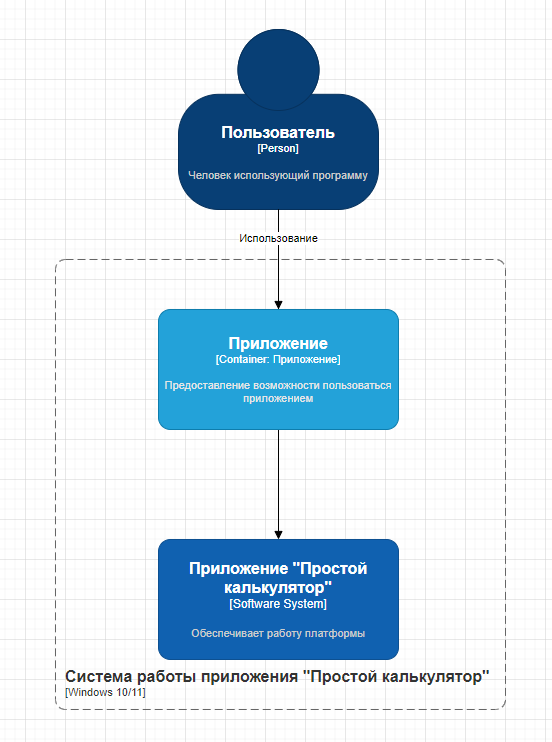


Рисунок 16 – уровень контейнеров

# 3.3.3 Уровень компонентов

Целевая аудитория этого уровня абстракции – программисты и архитекторы. Речь пойдет напрямую о технических характеристиках и рисках, показан на рисунке 17.

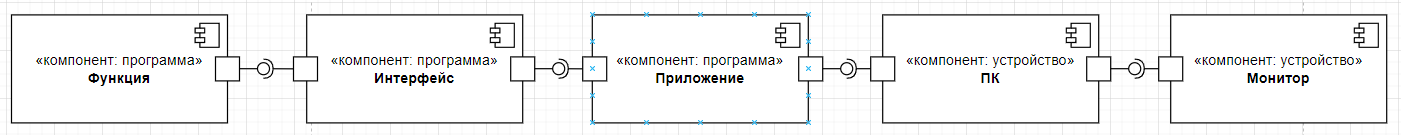


Рисунок 17 – уровень компонентов

# 3.3.4 Уровень кода

Диаграмма кода используется для низкоуровневой детализации системы. На официальной странице модели С4 рекомендуют использовать диаграмму классов, ER-диаграмму или схожие нотации, показан на рисунке 18.

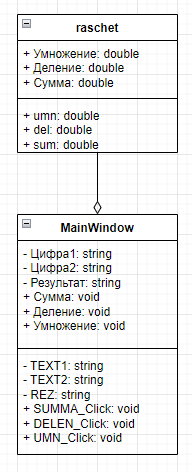


Рисунок 18 – Уровень кода

# 4. Тестирование программного обеспечения

# 4.1 Объект испытаний

Наименование программы «Простой калькулятор».

# 4.2 Цель испытаний

Цель проведения испытаний - проверка соответствия характеристик разработанной программы функциональным и отдельным иным видам требований, изложенным в документе Техническое задание.

# 4.2.1 Основание для проведения испытаний

Основанием является задание на практику.

# 4.2.2 Место и продолжительность испытаний

Дом, Колледж. Срок 2 недели.

# 4.2.3 Перечень документов, предъявляемых на испытании

Состав программной документации должен включать в себя:

* техническое задание;
* руководство оператора;
* программа, методика и результаты испытаний.

# 4.2.4 Объём испытаний

Перечень этапов испытаний:

- ознакомительный;

- испытания.

# 4.3 Требования к программе

При проведении испытаний функциональные характеристики программы подлежат проверке на соответствие требованиям, изложенным в п. «Требования к функциональным выполняемых функций» технического задания.

# 4.3.1 Требования к программной документации

Состав программной документации должен включать в себя:

* техническое задание;
* руководство оператора;
* программа, методика и результаты испытаний.

# 4.3.2 Технические средства, используемые во время испытаний

В состав технических средств должен входить персональный компьютер включающий в себя:

* **Процессор:** AMD Athlon 3000G (2.0 ГГц) и более
* **ОЗУ:** 2 Гб и более
* **Необходимо место на жестком диске:** 50 Мб
* **Видеоадаптер:** NVIDIA GeForce GTX 750 Ti и лучше
* **Экран:** 1920 х 1080 и выше

# 4.4 Методика испытаний

**Методы проведения проверки комплектности программной документации**

Проверка комплектности программной документации на программное изделие производится визуально преподавателем. В ходе проверки сопоставляется состав и комплектность программной документации, представленной исполнителем, с перечнем программной документации.

Проверка считается завершенной в случае соответствия состава и комплектности программной документации, представленной исполнителем, перечню программной документации, приведенному в указанном выше пункте.

**Методы проведения проверки степени выполнения требований функционального назначения программы**

Тестирование чёрного ящика или поведенческое тестирование — стратегия (метод) тестирования функционального поведения программы с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта. Под методом понимаются систематические методы отбора и создания тестов для тестового набора. Стратегия поведенческого теста исходит из технических требований и их спецификаций.

# 4.5 Результаты предварительных испытаний

Результаты предварительного тестирования в соответствии с таблицей 1.

Таблица 1 - Результаты тестов.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Описание** | **Шаги теста** | **Ожидаемые результаты** | **Реальные результаты** | **Прошел**  **/Провалился** | **Тестер**  **/Дата** |
| 1 | Тестирование функции расчёт суммы чисел | Ввести необходимые данные и нажать кнопку «СУММА» | 5 | 5 | Прошёл | Гергин Александр  23.12.2024 |
| 2 | Тестирование функции расчёт суммы деления | Ввести необходимые данные и нажать кнопку «ДЕЛЕНИЕ» | 9 | 9 | Прошёл | Гергин Александр  23.12.2024 |
| 3 | Тестирование функции расчёт суммы умножения | Ввести необходимые данные и нажать кнопку «УМНОЖЕНИЕ» | 25 | 25 | Прощёл | Гергин Александр  23.12.2024 |

Продолжение таблицы 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | Тестирование кнопки «ОЧИСТКА» | Ввести требуемые данные, нажать кнопку расчёт. После нажать на кнопку «ОЧИСТКА» | Очистились все элементы | Очистились все элементы | Прошёл | Гергин Александр  23.12.2024 |
| 4 | Тестирование кнопки «О ПРОГРАММЕ» | Нажать на кнопку «О ПРОГРАММЕ» | Вышло сообщение в модальном окне | Вышло сообщение в модальном окне | Прошёл | Гергин Александр  23.12.2024 |
| 5 | Тестирование кнопки «ВЫХОД» | Запустить программу, после нажать кнопку «ВЫХОД» | Программа закроется | Программа закроется | Прошёл | Гергин Александр  23.12.2024 |

# ЗАКЛЮЧЕНИЕ

В ходе выполнения практического задания по разработке приложения «Простой калькулятор» были созданы две основные функции: расчёт суммы чисел, делений и умножений. Пользовательский интерфейс получился простым и понятным, а функции доступны через соответствующие кнопки. Возможность ввода данных пользователем и отображения результатов также была предусмотрена. После проведения тестирования было подтверждено корректное функционирование всех функций и устранение возможных ошибок. Приложение «Простой калькулятор» готово к использованию и может быть полезным для пользователей в различных ситуациях.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мартин, Роберт. *Чистый код: создание, анализ и рефакторинг*. 2-е изд. Санкт-Петербург: Издательский дом «Питер», 2020. 480 с.
2. Бах, Д. *Современный C#: Полное руководство по программированию*. 1-е изд. Москва: Издательство «Вильямс», 2021. 600 с.
3. Грейс, Р. *C# для начинающих: Пошаговое руководство*. 3-е изд. Москва: Издательство «Наука», 2019. 350 с.
4. Керниган, Б., Ричи, Д. *Язык программирования C: Полное руководство*. 2-е изд. Санкт-Петербург: Издательский дом «Питер», 2018. 720 с.
5. Фаулер, М. *Рефакторинг: улучшение существующего кода*. 3-е изд. Москва: Издательство «Вильямс», 2021. 450 с.

# ПРИЛОЖЕНИЕ

Код интерфейса:

<Window x:Class="RSSK\_PRACT.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:RSSK\_PRACT"

mc:Ignorable="d"

Title="ПРОСТОЙ КАЛЬКУЛЯТОР" MinHeight="450" MinWidth="800" Height="450" Width="800">

<Grid Background="Bisque">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<StackPanel Grid.Column="2" Grid.Row="2">

<Button x:Name="SUMMA" Click="SUMMA\_Click" Grid.Column="2" Grid.Row="2" Content="СУММА"/>

<Button x:Name="UMN" Click="UMN\_Click" Grid.Column="2" Grid.Row="3" Content="УМНОЖЕНИЕ"/>

<Button x:Name="DELEN" Click="DELEN\_Click" Grid.Column="2" Grid.Row="4" Content="ДЕЛЕНИЕ"/>

</StackPanel>

<Button x:Name="EXIT" Click="EXIT\_Click" Grid.Column="5" Grid.Row="5" Content="ВЫХОД"/>

<Button x:Name="ABOUT" Click="ABOUT\_Click" Grid.Column="4" Grid.Row="5" Content="О ПРОГРАММЕ"/>

<Button x:Name="CLEAR" Click="CLEAR\_Click" Grid.Column="3" Grid.Row="5" Content="ОЧИСТКА"/>

<StackPanel Grid.Column="1" Grid.Row="2" Grid.RowSpan="2">

<Label Content="ЧИСЛО 1"/>

<TextBox x:Name="TEXT1" TextChanged="TEXT1\_TextChanged" />

<Label Content="ЧИСЛО 2"/>

<TextBox x:Name="TEXT2" TextChanged="TEXT2\_TextChanged" />

</StackPanel>

<TextBox IsReadOnly="True" x:Name="REZ" Grid.Column="3" Grid.Row="2"/>

</Grid>

</Window>

Код программы:

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace RSSK\_PRACT

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void EXIT\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void ABOUT\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("dd");

}

public void SUMMA\_Click(object sender, RoutedEventArgs e)

{

REZ.Clear();

if (double.TryParse(TEXT1.Text, out double a) && double.TryParse(TEXT2.Text, out double b))

{

double c = raschet.sum(a,b);

REZ.Text = c.ToString();

}

else

{

MessageBox.Show("ERROR");

}

}

public void UMN\_Click(object sender, RoutedEventArgs e)

{

REZ.Clear();

if (double.TryParse(TEXT1.Text, out double a) && double.TryParse(TEXT2.Text, out double b))

{

double c = raschet.umn(a,b);

REZ.Text = c.ToString();

}

else

{

MessageBox.Show("ERROR");

}

}

public void DELEN\_Click(object sender, RoutedEventArgs e)

{

REZ.Clear();

if (double.TryParse(TEXT1.Text, out double a) && double.TryParse(TEXT2.Text, out double b))

{

double c = raschet.del(a, b);

REZ.Text = c.ToString();

}

else

{

MessageBox.Show("ERROR");

}

}

private void CLEAR\_Click(object sender, RoutedEventArgs e)

{

REZ.Text = null;

TEXT1.Text = null;

TEXT2.Text = null;

}

private void TEXT1\_TextChanged(object sender, TextChangedEventArgs e)

{

REZ.Text = null;

}

private void TEXT2\_TextChanged(object sender, TextChangedEventArgs e)

{

REZ.Text = null;

}

}

}

Код класса raschet:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RSSK\_PRACT

{

public class raschet(double a,double b)

{

public static double umn(double a,double b)

{

double c = a \* b;

return c;

}

public static double del(double a,double b)

{

double c = a / b;

return c;

}

public static double sum(double a,double b)

{

double c = a + b;

return c;

}

}

}