**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#__RefHeading___4430)

[1 Постановка задачи 6](#__RefHeading___4431)

[1.1 Анализ предметной области 6](#__RefHeading___4432)

[2 Техническое задание на разработку ГОСТ 19.201-78 6](#__RefHeading___4433)

[2.1 Основания для проведения разработки 6](#__RefHeading___4434)

[2.2 Назначение разработки 6](#__RefHeading___4435)

[2.2.1 Функциональное назначение 6](#__RefHeading___4436)

[2.2.2 Эксплуатационное назначение 6](#__RefHeading___4437)

[2.3 Требования к программе или программному изделию 6](#__RefHeading___4438)

[2.3.1 Требования к функциональным характеристикам 6](#__RefHeading___4439)

[2.3.2 Требования к надежности 6](#__RefHeading___4440)

[2.3.3 Время восстановления после отказа 6](#__RefHeading___4441)

[2.3.4 Требования к составу и параметрам технических средств 7](#__RefHeading___4442)

[2.4 Паспорта основных модулей и метрики 7](#__RefHeading___4443)

[2.4.1 Паспорт модуля GetNumericValues 7](#__RefHeading___4444)

[2.4.2 Метрики 8](#__RefHeading___4445)

[3 Проектирование 10](#__RefHeading___4446)

[3.1 Wireframe-технология 10](#__RefHeading___4447)

[3.2 Поведенческие диаграммы 10](#__RefHeading___4448)

[3.3 Структурные диаграммы 13](#__RefHeading___4449)

[3.4 Модель С4 16](#__RefHeading___4450)

[4 Тестирование программы 18](#__RefHeading___4451)

[4.1 Программа, методика и результаты испытаний ГОСТ 19.301-79 18](#__RefHeading___4452)

[4.2 Результаты предварительных испытаний 19](#__RefHeading___4453)

[ЗАКЛЮЧЕНИЕ 21](#__RefHeading___4454)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 22](#__RefHeading___4455)

[ПРИЛОЖЕНИЕ А 23](#__RefHeading___4456)

# ВВЕДЕНИЕ

В современном мире, где информационные технологии играют важную роль в творении, автоматизация различных процессов становится необходимостью. В данном случае творение подразумевается создание творческих программ, которые выполняют необычную функцию.

Актуальность данной работы обусловлена необходимостью проверить наши творческие знания и создать что-то необычное, например отображение графика при помощи слов.

Целью проекта является разработка программы, которая будет выполнять указанные функции, используя объектно-ориентированное программирование. Важной задачей является соблюдение критериев, установленных в задании, включая разработку алгоритма основных функций в виде блок-схем, подготовку паспортов модулей программы, проектирование диаграмм и моделирование С4. Все этапы разработки будут сопровождаться снятием метрик, а готовый проект будет размещён на репозитории GitHub.

# 1 Постановка задачи

# 1.1 Анализ предметной области

Целью разработки проекта: пользователь вводит строку. Каждый символ преобразовать в цифровой показатель: А = -16, Б= -15 ... Я = 13 (через классы). Построить график на основании полученного «слова».

# 2 Техническое задание на разработку ГОСТ 19.201-78

# 2.1 Основания для проведения разработки

Основанием для проведения разработки является задание на учебную практику по УП 02.

# 2.2 Назначение разработки

# 2.2.1 Функциональное назначение

Функциональное назначение заключается построение графика при помощи ввода определенных слов вводя заглавные буквы.

# 2.2.2 Эксплуатационное назначение

Требования эксплуатационного назначения не предъявляются.

# 2.3 Требования к программе или программному изделию

# 2.3.1 Требования к функциональным характеристикам

Программа должна обладать следующими функциональными характеристиками:

– функция ввода слова,

– функция отображения графика,

– функция выхода,

– функция отображения окна «О программе».

# 2.3.2 Требования к надежности

Требования к обеспечению надежного функционирования программы не предъявляются.

# 2.3.3 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 20 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

# 2.3.4 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающая в себя:

* процессор Pentium-1000 с тактовой частотой, ГГц - 10, не менее;
* материнскую плату с FSB, ГГц - 5, не менее;
* оперативную память объемом - не менее 8ГБ.

# 2.4 Паспорта основных модулей и метрики

# 2.4.1 Паспорт модуля GetNumericValues

Таблица 1 – Спецификация модуля GetNumericValues

|  |  |  |
| --- | --- | --- |
| **Имя** | **Реализуемая функция** | **Параметры** |
| List<int> GetNumericValues () | Отображение графика в Canvas | Выходные данные: numericValues – отображение графика |

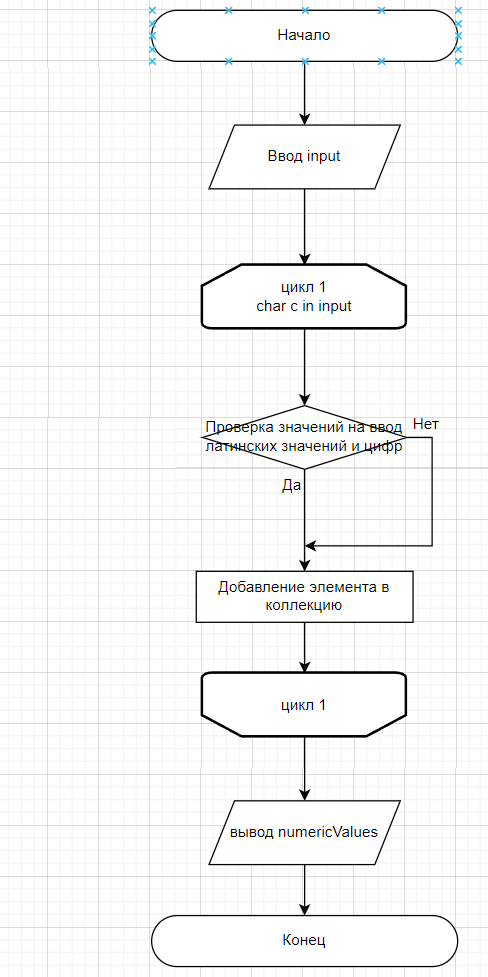


Рисунок 1 – Алгоритм класса для отображения графика

# 2.4.2 Метрики

Метрики – показатели, используемые для оценки характеристик как всего программного продукта, так и его отдельных компонентов.

Каждый из показателей, представленных в нашем обзоре метрик, имеет свои формулы и обоснования. Рассмотрим их ниже:

– индекс удобства поддержки (ИУП). Комплексный показатель качества кода. Эта метрика варьируется от 0 до 100, демонстрируя, насколько сложен код и каковы возможности его оптимизации. Наш ИУП составляет 76. Его можно улучшить, дорабатывая наш код.

– Сложность организации циклов. Структурный показатель, который оценивает нагрузку, создаваемую такими операторами, как if-else, while, for, do-while и другими. В нашем проекте сложность организации циклов равна 25.

– Глубина наследования. Метрика, отражающая взаимосвязи между классами, включая наследование, которое может уходить вплоть до базовых классов, необходимых для реализации протоколов решения. Эта концепция схожа со связями между классами, позволяя понимать, как они взаимодействуют друг с другом. Глубина наследования равна 9.

– Взаимозависимость классов. Показатель, который оценивает уровень взаимодействия данного класса с другими классами посредством параметров. Эти параметры могут включать в себя локальные переменные, типы, которые возвращаются, вызовы как шаблонов, так и экземпляров, а также передачу объектов. Уровень взаимодействия классов равен 29.

– Строки исходного кода. Оценивает компактность кода и указывает точное количество строк исходного кода для каждого программного компонента. Уровень строк исходного кода равен 246.

– Строки исполняемого кода. Метрика измеряет количество операций в исполняемом коде. На данный момент этот показатель равен 55.

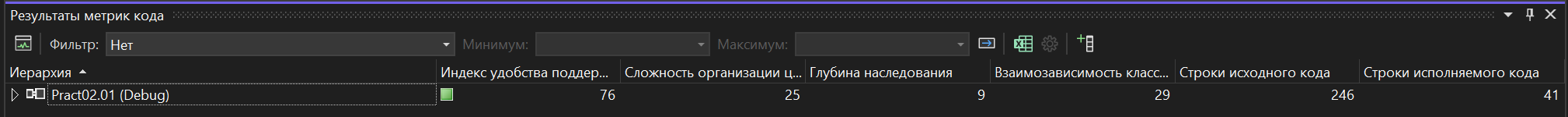


Рисунок 2 – Метрики в Visual Studio

# 3 Проектирование

# 3.1 Wireframe-технология

Ниже отображен график слова, созданный в draw.io с помощью Wireframe-технологии.

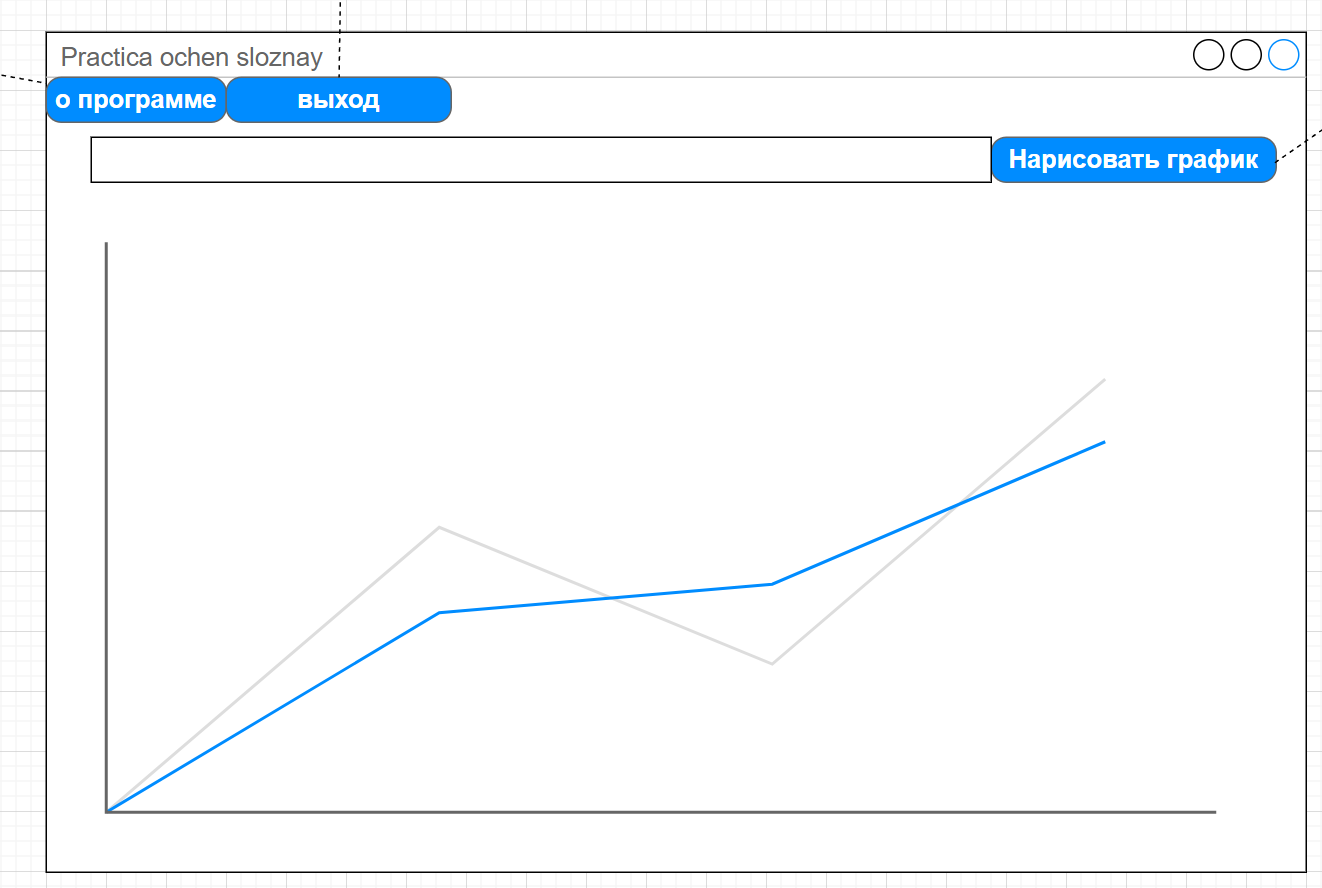


Рисунок 3 – Схема графика слов

Wireframe-технологии – скелет (каркас) будущей разработки, данное графическое представление интерфейса. Эта технология отображает связи между элементами, последовательность окон, но не отображает функциональность и уточняющие элементы.

# 3.2 Поведенческие диаграммы

В данном пункте будут отображены основные поведенческие диаграммы.

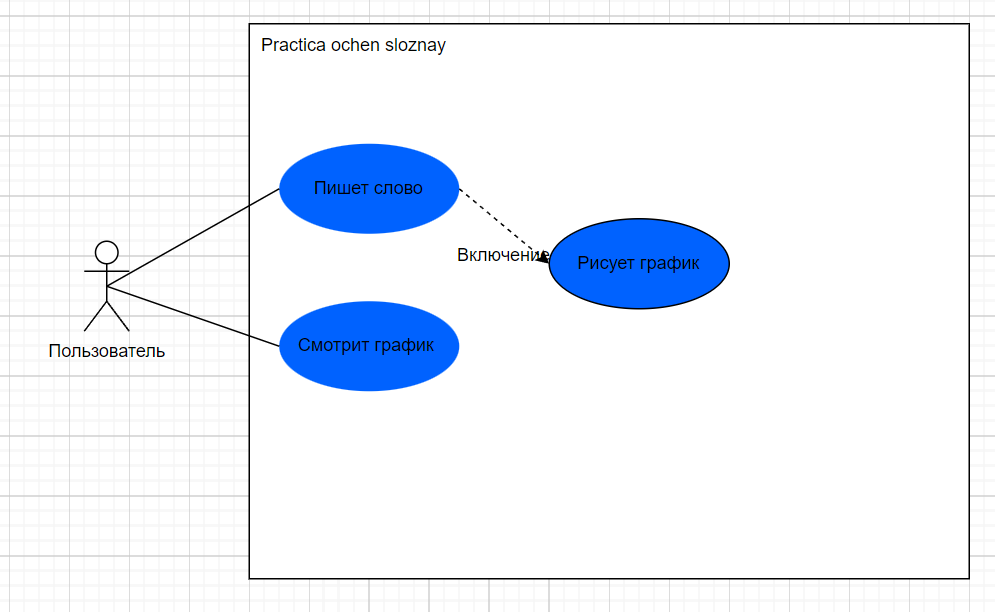


Рисунок 4 – Диаграмма вариантов использования

Диаграмма варианта использования – поведенческая диаграмма, которая показывает воздействие действий субъекта безвременного параметра.

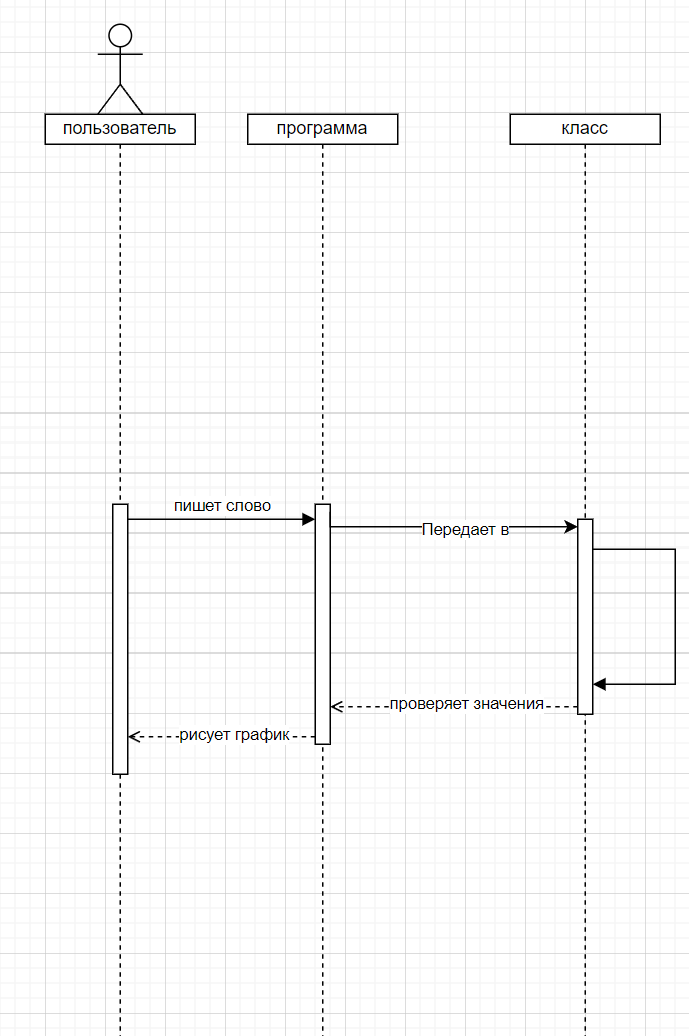


Рисунок 5 – Диаграмма последовательности

Диаграмма последовательности – вид графического изображения, существующая во временном промежутке.

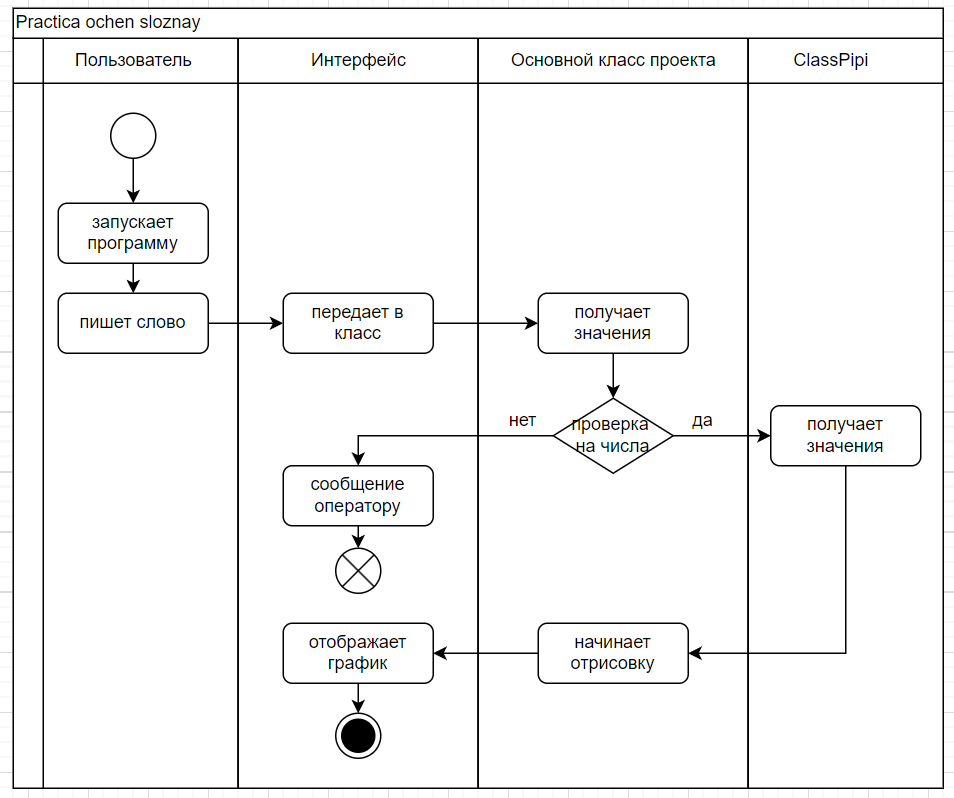


Рисунок 6 – Диаграмма деятельности

Диаграмма деятельности или активности – помогает отследить все процессы, выполняемые определенными объектами и субъектами.

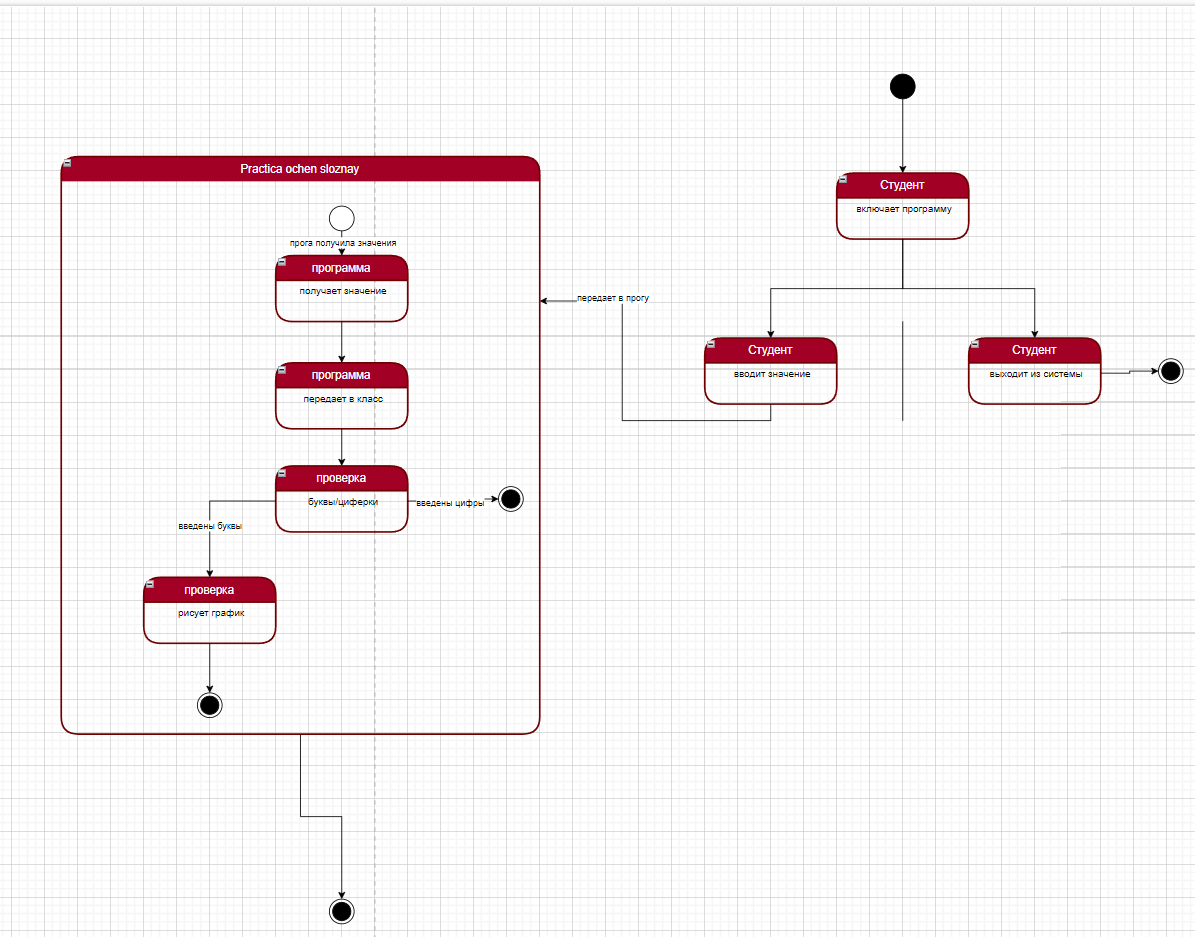


Рисунок 7 – Диаграмма состояний

Диаграмма состояния – поведенческая диаграмма, служащая для рассмотрения всех объектов и субъектов.

# 3.3 Структурные диаграммы

В данном пункте будут отображены все основные структурные диаграммы.

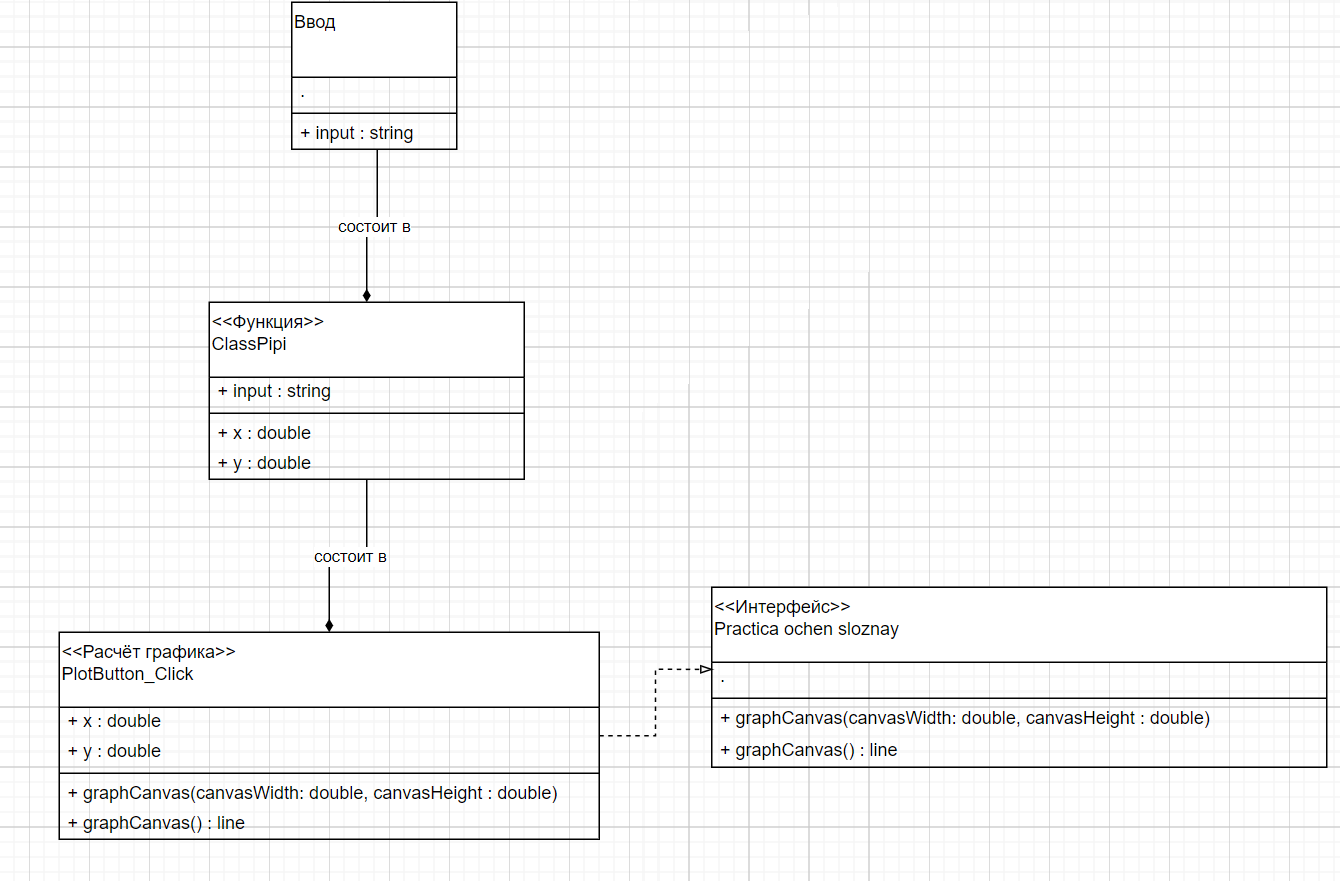


Рисунок 8 – Диаграмма классов

Диаграмма классов – структурная диаграмма, отражающая программные компоненты такие как классы, процедуры, функции, методы, события, базы данных. С помощью диаграмм классов невозможно отобразить аппаратные компоненты.

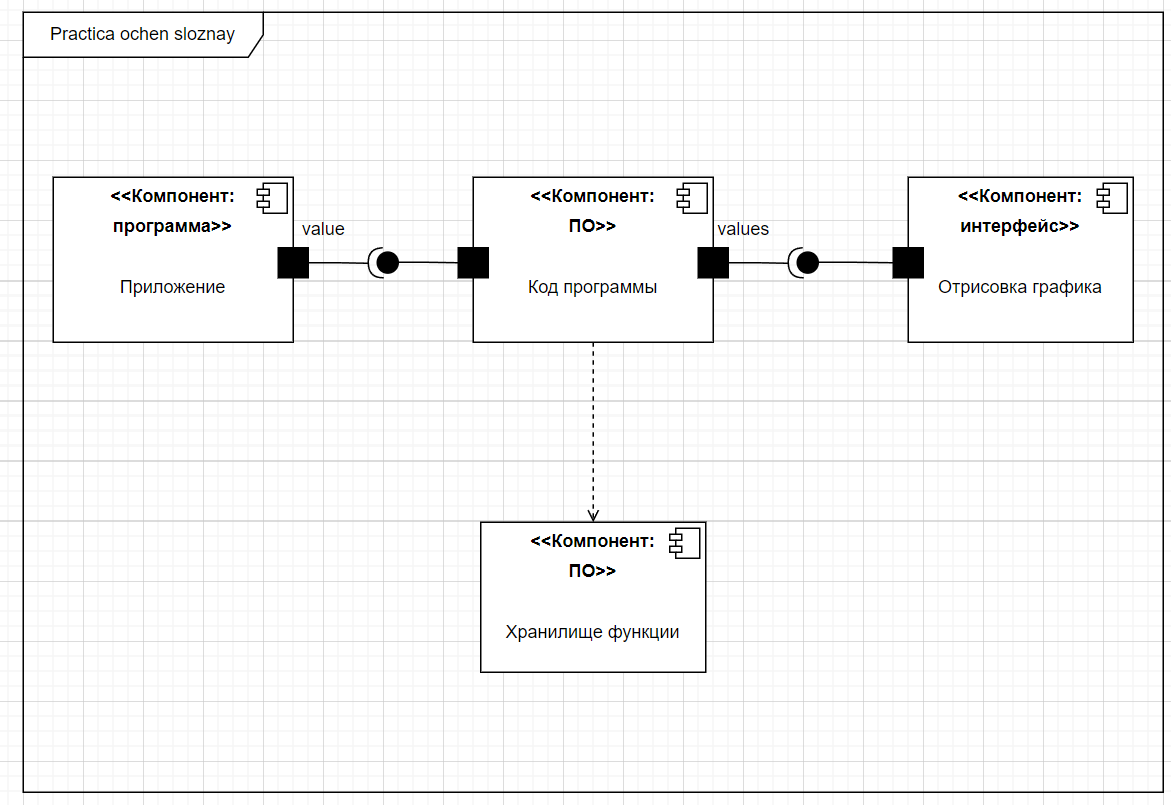


Рисунок 9 – Диаграмма компонентов

Диаграмма компонентов – структурная диаграмма, которая показывает общую структуру проекта, при этом модели первой программы сужаются до компонентов и рассматривает зависимости между каждой структурной частью.

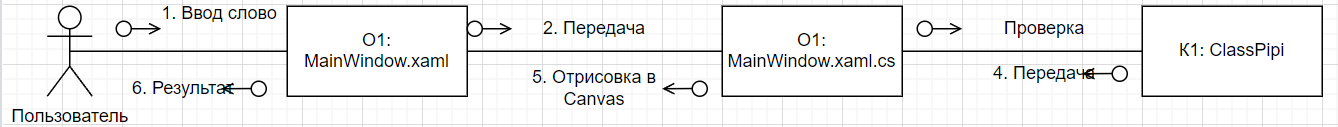


Рисунок 10 – Диаграмма кооперации

Диаграмма кооперации – структурная диаграмма предназначенная для описания системы и взаимодействия аппаратных, программных и человеческих уровней.

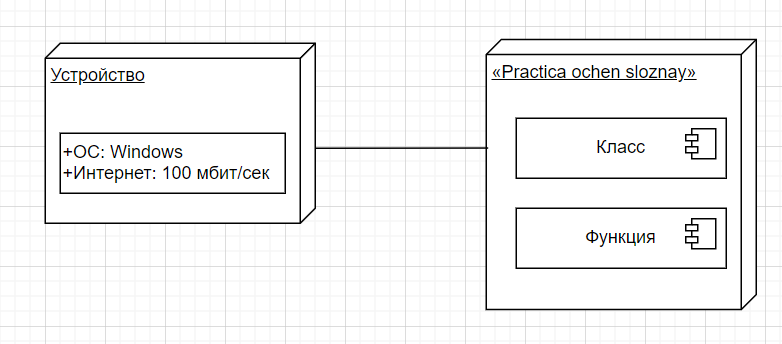


Рисунок 11 – Диаграмма развертывания

Диаграмма развёртывания – структурная диаграмма, которая позволяет описать аппаратуру и компоненты, которые в этот момент будут задействованы.

# 3.4 Модель С4

Модель С4 включается в себя 4 уровня, которые изображены на рисунках ниже.



Рисунок 12 – Первый уровень модели С4

Первый уровень разрабатывает диаграммы контекста, необходимую для описания структуры пользователю без необходимости вдаваться в код.

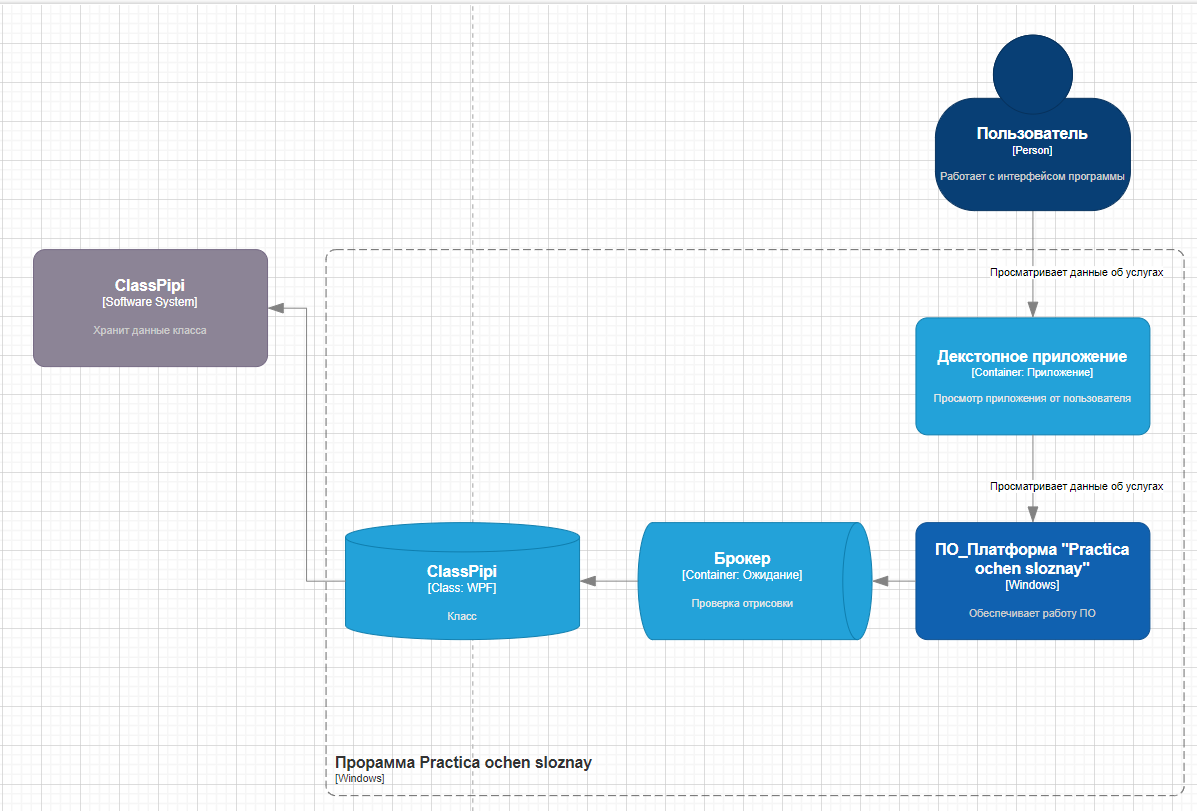


Рисунок 13 – Второй уровень модели С4

Второй уровень – диаграмма контейнеров. На этом этапе добавляется внедрение кода в ИС.

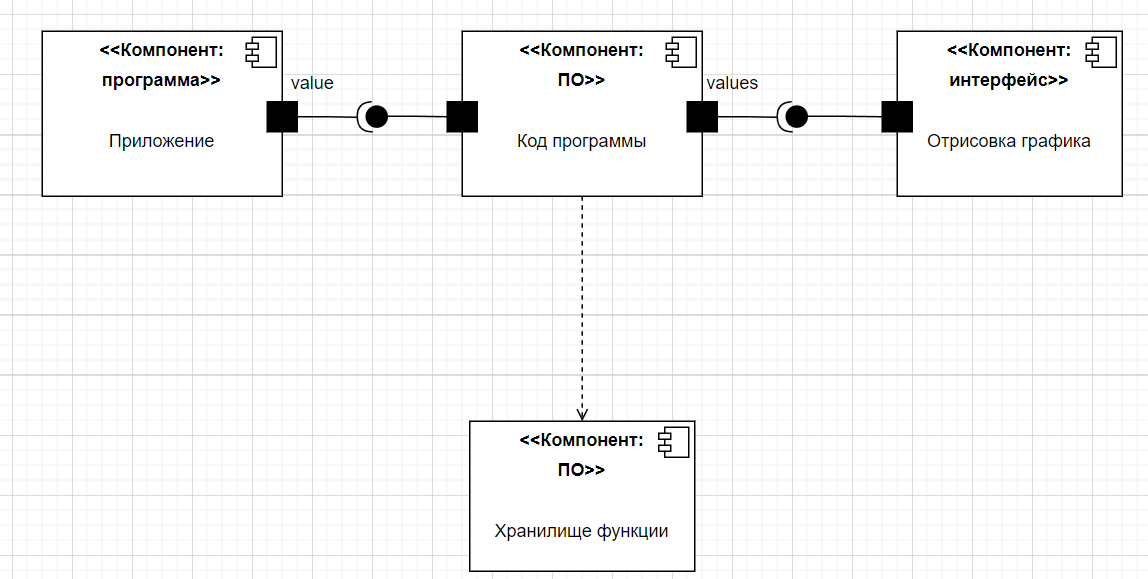


Рисунок 14 – Третий уровень модели С4

Третий уровень – компоненты. Данный уровень необходим для понимания разрабатываемым архитектором ИС.

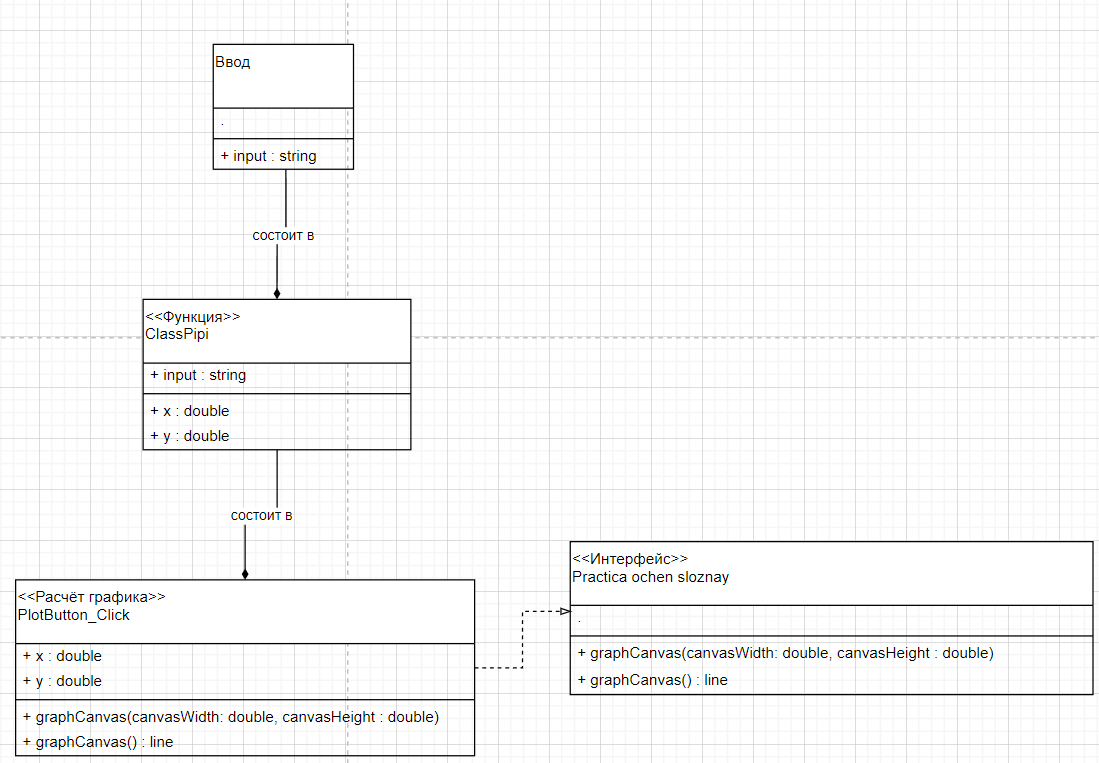


Рисунок 15 – Четвертый уровень модели С4

Четвертый уровень - диаграмма кода. Диаграмма классов или ER-диаграмма отображается в случае когда админ не вовлечен в процесс разработки.

Модель С4 – необходим для отображения ИС на интуитивном уровне для разработчиков, пользователей и тестировщиков.

# 4 Тестирование программы

# 4.1 Программа, методика и результаты испытаний ГОСТ 19.301-79

**Методы проведения проверки комплектности программной документации**

Проверка комплектности программной документации на программное изделие производится визуально тестировщиком. 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 белого ящика — это способ проверки ПО, который концентрируется на внутренней системе и коде программы. У QA-специалистов есть доступ к исходному коду и документации проекта, что позволяет им исследовать и проверять внутреннюю работу, инфраструктуру и интеграцию программного обеспечения.

# 4.2 Результаты предварительных испытаний

Результаты предварительного тестирования будут отображены ниже в таблице 2.

Таблица 2 – Функциональное тестирование «белого ящика»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тестовая спецификация** | | | | | | |
| **Разработал:** Мелёхин Дмитрий | | | **Начальные установки:** отсутствуют | | | |
| **Цель:** проверки степени выполнения требований функционального назначения программы. | | | | | | |
| **Test case**  **#** | **Описание** | **Шаги теста** | **Ожидаемый результат** | **Фактический результат** | **Пройден/не пройден** | **Тестер**  **/Дата** |
| **Тестирование основных функций** | | | | | | |
| 1 | Тестирование функциональности отображения графика | Нажатие на кнопку «Построить график» | Отображение графика | Отображение графика | Пройден | Мелёхин Дмитрий 19.12.2024 |
| 2 | Тестирование функциональности ввода слова | Заполнение текстового поля значением | Ввод слов | Ввод слов | Пройден | Мелёхин Дмитрий 19.12.2024 |

Продолжение таблицы 2– Функциональное тестирование «белого ящика»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | Тестирование ввода неправильных значений | Заполнение текстового поля неправильными значениями и нажатие на кнопку «Построить график» | Появления окна ошибки оператору | Появления окна ошибки оператору | Пройден | Мелёхин Дмитрий 19.12.2024 |
| 4 | Тестирование функции выхода из программы | Нажатие на кнопку «Выход» | Сворачивание программы | Сворачивание программы | Пройден | Мелёхин Дмитрий 19.12.2024 |
| 5 | Тестирование функции ввода цифр | Заполнение текстового поля цифрами и нажатие на кнопку «Построить график» | Появления окна ошибки оператору | Появления окна ошибки оператору | Пройден | Мелёхин Дмитрий 19.12.2024 |

# ЗАКЛЮЧЕНИЕ

Данный отчет продемонстрировал эффективность использования объектно-ориентированного подхода для создания программы, которая рисует график при помощи ввода слов.

Блок-схемы алгоритмов и паспорт модуля программы были разработаны в соответствии с установленными критериями. Диаграммы и модель С4, спроектированные в рамках работы, показали, насколько важно учитывать различные аспекты проектирования программного обеспечения для получения качественного результата.

Выгрузка готового проекта на репозиторий GitHub позволила сохранить все этапы разработки и сделать их доступными для дальнейшего изучения и возможного усовершенствования. Метрики дали возможность оценить продуктивность работы.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Урок 1. Wireframe-технология. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
2. Урок 2. Диаграммы варианта использования. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
3. Урок 3. Диаграмма последовательности. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
4. Урок 4. Диаграмма деятельности. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
5. Урок 5. Диаграмма состояния. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
6. Урок 6. Диаграмма классов. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
7. Урок 7. Диаграмма компонентов. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
8. Урок 8. Диаграмма кооперации. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
9. Урок 9. Диаграмма развертывания. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER
10. Урок 10. Модель С4. – [Электронный ресурс] – режим доступа: лекция на FILE-SERVER

# ПРИЛОЖЕНИЕ А

Листинг класса GetNumericValues

public static List<int> GetNumericValues(string input)

{

List<int> numericValues = new List<int>();

foreach (char c in input)

{

if (CharacterValues.TryGetValue(c, out int value))

{

numericValues.Add(value);

}

}

return numericValues;

}