**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА   
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ**

**ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ – филиал РАНХиГС**

**Отчет по программированию №9**

Выполнил: студент группы Иб-321

Комышков Владислав Дмитриевич

Оглавление

[Задание 1 3](#_Toc122898885)

[Текст задания: 3](#_Toc122898886)

[Скрипт 3](#_Toc122898887)

[Результат 4](#_Toc122898888)

[Итог 4](#_Toc122898889)

[Задание 2 5](#_Toc122898890)

[Текст задания: 5](#_Toc122898891)

[Схемы 5](#_Toc122898892)

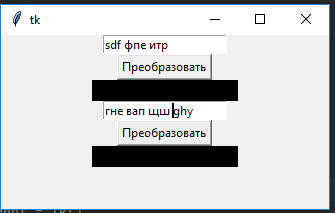
[Ход работы 6](#_Toc122898893)

[Проверка программы 12](#_Toc122898894)

[Итог 13](#_Toc122898895)

# Задание 1

### Текст задания:



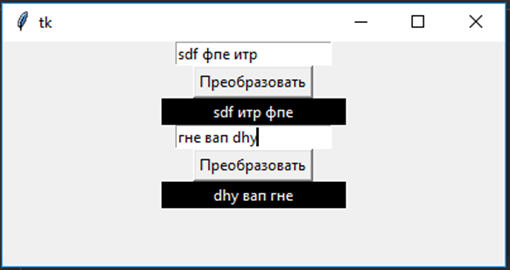
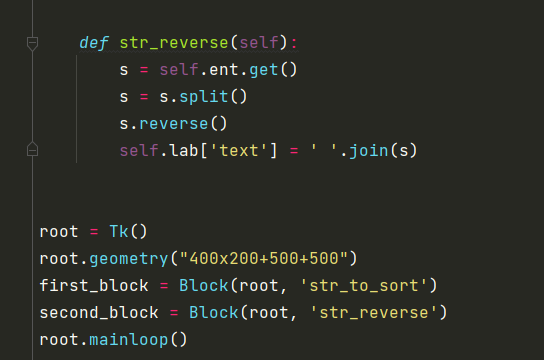


Рисунок 9.4.

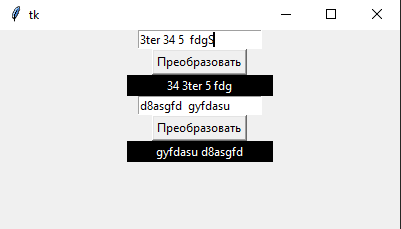
Получить результат, показанный на рисунке 9.4.

## Скрипт





## Результат



## Итог

Мы реализовали скрипт и получили результат показанный на рисунке 9.4

# Задание 2

### Текст задания:

Напишите простейший калькулятор, состоящий из двух текстовых полей, куда пользователь вводит числа, и четырех кнопок "+", "-", "\*", "/". Результат вычисления должен отображаться в метке. Если арифметическое действие выполнить невозможно (например, если были введены буквы, а не числа), то в метке должно появляться слово "ошибка" (Пример на рисунке 9.5). .

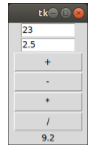


Рисунок 9.5

## Схемы

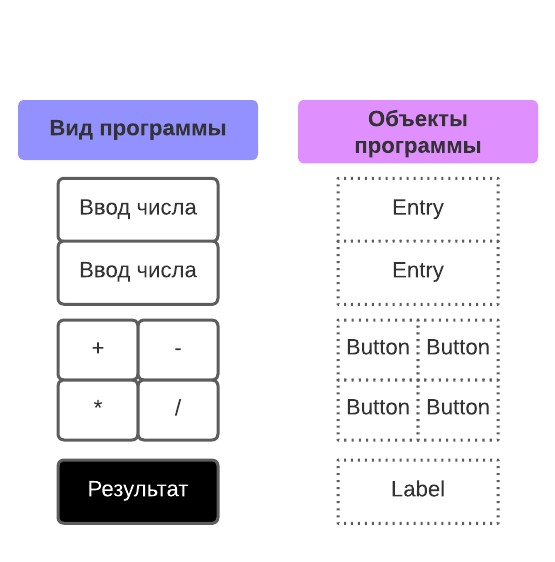


Рис 2.1 – вид программы и объекты

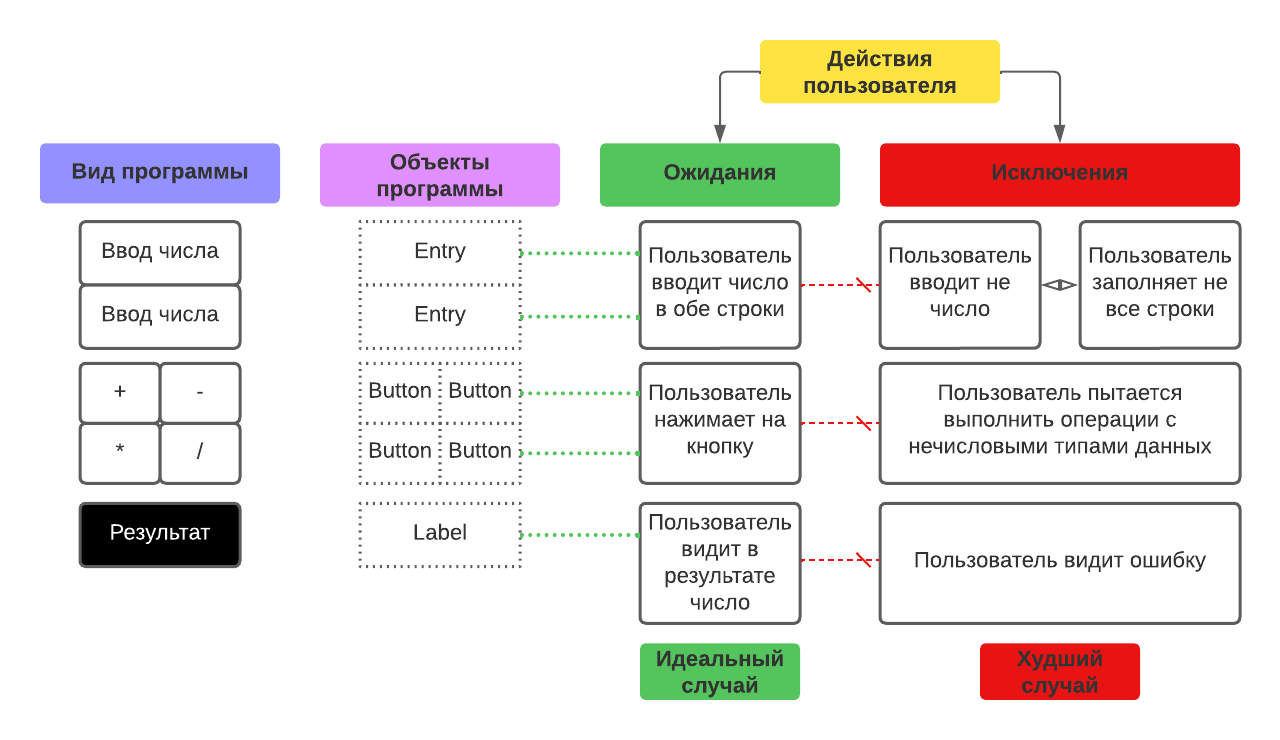


Рис 2.2 – действия пользователя

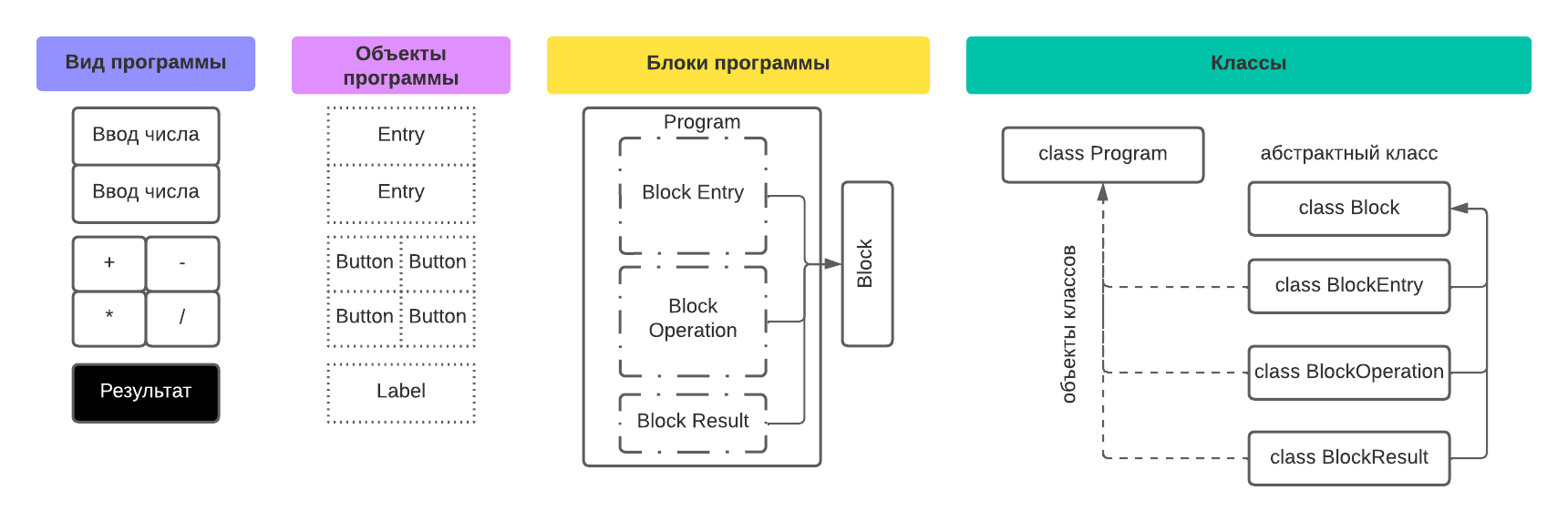


Рис 2.3 – Блоки программы и классы

## Ход работы

Сперва создаем два файла **main.py** и **ClassProgram.py**.

В **main.py** будем импортировать класс **Program** и запускать программу.

В **ClassProgram.py** пропишем классы **Program**, **Block**, **BlockEntry**, **BlockOperation**, **BlockResult** (рис 2.4).



Рис 2.4 – классы файла **ClassProgram.py**

Для написания программы нам понадобятся модули:

* **tkinter** – это графическая библиотека, позволяющая создавать программы с оконным интерфейсом.
* **abc** - Абстрактные базовые классы¶. Данный модуль предоставляет инфраструктуру для определения абстрактных базовых классов (ABC) в Python.
* **re** -  модуль предоставляет операции сопоставления регулярных выражений в Python.

Класс **Program** будет наследоваться от класса **Tk**, содержащего все нужные для нашей программы виджеты (рис 2.5).

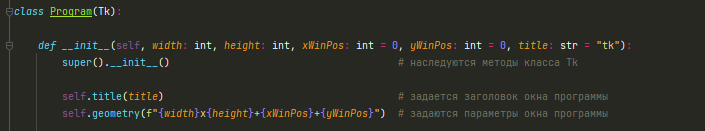


Рис 2.5 – наследование от класса **Tk**

Класс **Block** будет наследоваться от вспомогательного класса **ABC** модуля **abc**. Т.е. класс **Block** станет базовым абстрактным классом для классов **BlockEntry**, **BlockOperation**, **BlockResult.** Также в классе **Block** будет определен абстрактный метод **setGrid**, обязательный для всех подклассов (рис 2.6).

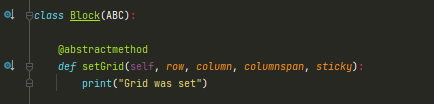


Рис 2.6 – абстрактный класс **Block** и абстрактный метод **setGrid**

Абстрактный метод **setGrid** позволит задать сетку для элементов (виджетов) программы. Параметры метода:

* **row** – индекс строки
* **column** – индекс столбца
* **columnspan** – кол-во ячеек занимаемых элементом
* **sticky** – позволит растянуть элементы по всей площади окна программы

При инициализации классы **BlockEntry**, **BlockOperation**, **BlockResult** будут принимать параметр ***master***, который является родительским виджетом – класс **Tk** (рис 2.7). Также будут задаваться параметры соответствующие параметрам определенных в классах-блоках виджетов.

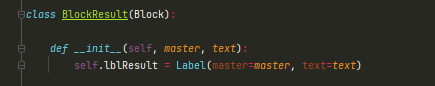
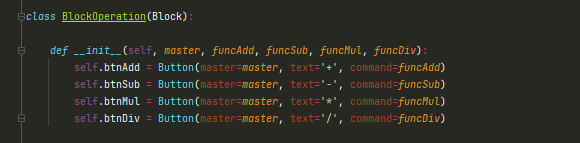
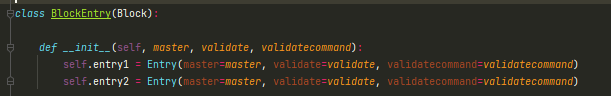


Рис 2.7 – классы-блоки

В классе **Program** зададим переменные соответствующие классам-блокам и установим ячейки для каждого элемента с помощью абстрактного метода **setGrid** (рис 2.8). Затем зададим вес для столбцов, чтобы растянуть виджеты кнопок равномерно по окну (рис 2.9).

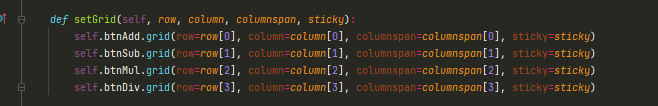
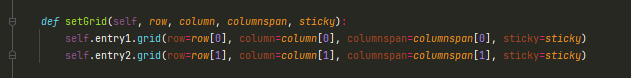




Рис 2.8 – переменные класса **Program** и установка сетки

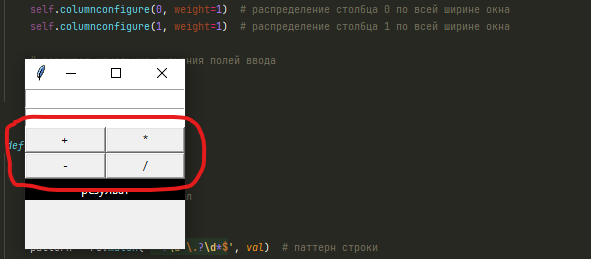


Рис 2.9 – распределение столбцов по ширине окна

Для блока-класса операций создадим методы для сложения, вычитания, умножения и деления чисел (рис 2.10), которые пользователь будет вводить в элементы блока-класса полей ввода. Соответственно, в блок-класса полей ввода мы должны передать параметры для проверки ввода чисел (рис 2.11). Проверка вводимых символов будет происходить в методе **isValid** (рис 2.12), который будет возвращать строку соответствующую паттерну.

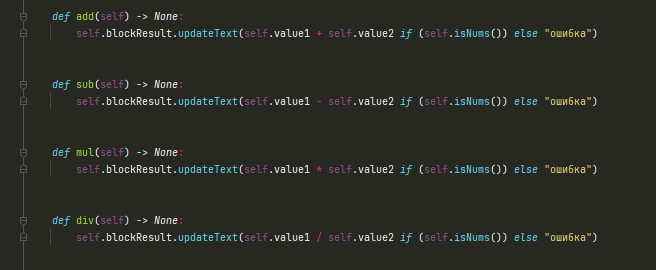


Рис 2.10 – операции



Рис 2.11 – проверка ввода

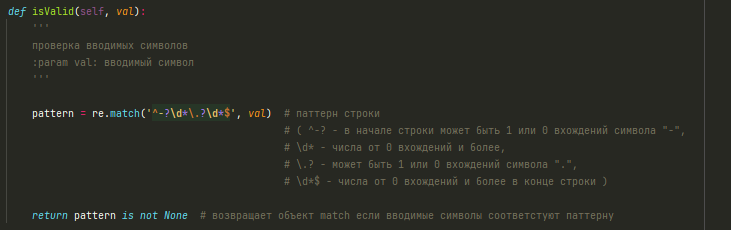
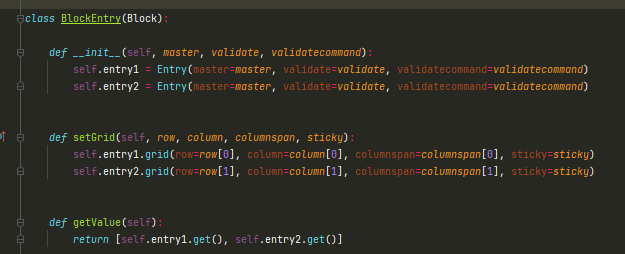


Рис 2.12 – проверка вводимых символов

В методе **isNums** будем проверять наличие символов в полях ввода, символы будем получать с помощью метода **getValue** блока-класса полей ввода. Если поля заполнены, текст полей будет конвертирован в тип **float** для дальнейшего выполнения операций над числами (рис 2.13).



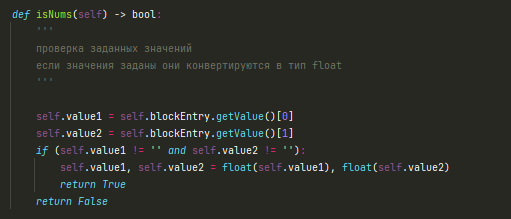


Рис 2.13 – проверка заданных значений

Последним определим метод для запуска программы – run (рис 2.14).



Рис 2.14 – запуск программы

В файле **main.py** импортируем класс **Program** и запускаем программу (Рис 2.15).



Рис 2.15 – скрипт файла **main.py**

## Проверка программы

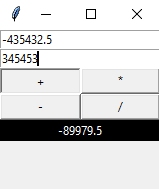
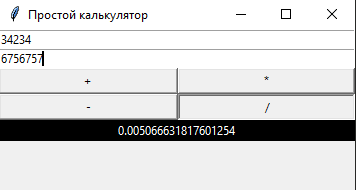


Рис 2.16 – проверка операций

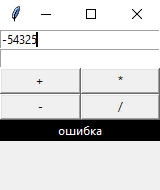


Рис 2.17 – вызов ошибки



Рис 2.18 – растягивание окна

## Итог

Мы создали программу с графическим интерфейсом, позволяющую выполнять операции сложения, вычитания, умножения и деления с двумя числами, а также видеть результат проведенных над ними операций.