**ИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Языки программирования

|  |
| --- |
| Наследование |

Руководитель А.В. Проскурин

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221219040 К.В. Трифонов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Получение практических навыков разработки и отладки программ c использованием механизма наследования.

# порядок выполнения работы

1. Ознакомиться с общей постановкой задачи.

2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы (при нехватке заданий вариант задания вычисляется как номер\_в\_списке\_группы - количество\_заданий).

3. Разработать классы согласно варианту задания.

4. Подготовить отчет по лабораторной работе. Отчет должен включать в себя:

• титульный лист;

• цель лабораторной работы;

• постановку задачи;

• текст программы с комментариями;

• демонстрацию работы программы (Снимки экрана при выполнении действий программы с описанием).

• краткие ответы на контрольные вопросы;

• выводы по лабораторной работе.

5. Защитить лабораторную работу перед преподавателем.

# Задания

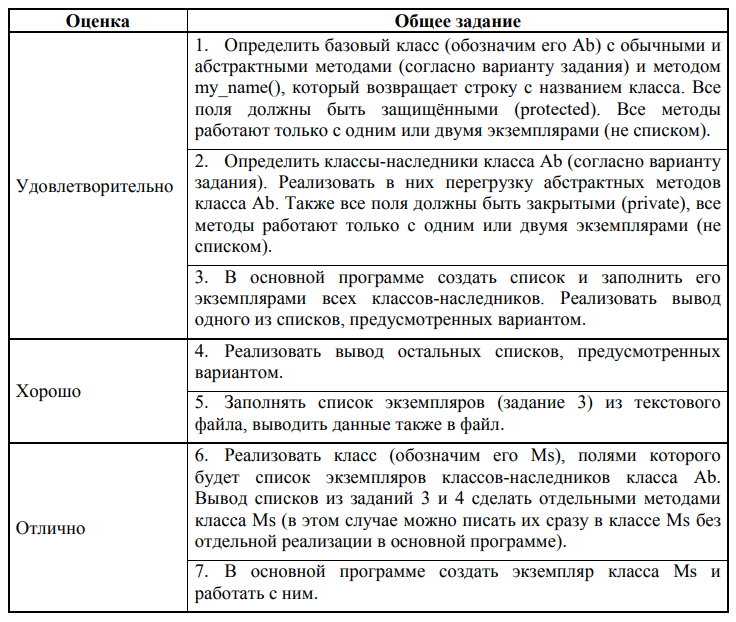
Необходимо разработать программу на языке Python, реализующую задачу ранее решенную на языке С++ в лабораторной работе №2.

Необходимо разработать программу, в которой будет реализовано несколько классов согласно варианту задания.

В реализуемой программе необходимо предусмотреть возможность демонстрации работы методов созданных классов.

Организацию взаимодействия с пользователем реализовать на основе меню.

В зависимости от оценки, на которую вы претендуете, необходимо выполнить следующие задания (для каждой следующей оценки нужно выполнить **ВСЕ** предыдущие задания, если обратное не сказано явно):



**Вариант №21.** Создать класс Line (Линия) с чистым виртуальным методом нахождения длины линии. На его основе создать классы:

- **ColoredLine** (Цветная линия) с полем «цвет»;

- **PolyLine** (Полилиния) с массивом длин участков ломаной линии (кроме первого). Для массива указателей на объекты этих классов предусмотреть возможность:

- вывода характеристик объектов;

- вывода объектов, длина которых находится в заданном диапазоне;

- вывода массива, отсортированного по длине линий.

Создать класс Picture, содержащий массив указателей на объекты этих классов. Написать демонстрационную программу, в которой будут использоваться все методы классов.

# ХОД РАБОТЫ

**Текст программы:**

**Содержимое файла aero.py:**

*#Рейс*

*class* Aeroflot:

*#Конструктор с параметрами*

*def* \_\_init\_\_(*self*, *destination* ="", *number* ="", *Type* ="", *time* ="", *day* =""):

*self*.\_\_destination = destination

*self*.\_\_number = number

*self*.\_\_Type = Type

*self*.\_\_time = time

*self*.\_\_day = day

*#Сетер из строки*

*def* set\_st(*self*, *st*):

*self*.\_\_destination,*self*.\_\_number,*self*.\_\_Type,*self*.\_\_time,*self*.\_\_day = st.strip().split('\t')

*#Гетеры*

*def* get(*self*):

        return *f*"Рейс №{*self*.\_\_number} в {*self*.\_\_destination} самолётом {*self*.\_\_Type} прилетает в {*self*.\_\_time} по {*self*.\_\_day}"

*def* getU(*self*):

        return *f*"{*self*.\_\_destination}\t{*self*.\_\_number}\t{*self*.\_\_Type}\t{*self*.\_\_time}\t{*self*.\_\_day}"

*def* get\_des(*self*):

        return *self*.\_\_destination

*def* get\_type(*self*):

        return *self*.\_\_Type

*def* get\_time(*self*):

        return (*int*(*self*.\_\_time),*self*.\_\_day)

Исходный класс Aeroflot (Рейс): Поля: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Конструктор с параметрами, методы задания полей через строку, возврата полей в виде форматированной и неформатированной строки, возврата полей пункта назначения, типа и времени (для выборок).

*#Класс массива рейсов*

*class* Flights:

*#Конструктор по умолчанию*

*def* \_\_init\_\_(*self*):

*self*.\_\_aeroflot = []

*#Ввод из файла*

*def* read(*self*,*filename*):

*self*.\_\_aeroflot = []

        with open(filename, 'r') as file:

            lines = file.readlines()

        for line in lines:

            flight = Aeroflot()

            flight.set\_st(line)

*self*.add(flight)

*#Вывод в файл*

*def* save(*self*, *filename*):

        with open(filename, 'w') as file:

            for flight in *self*.\_\_aeroflot:

                file.write(*f*"{flight.getU()}\n")

*def* add(*self*, *A*):

*self*.\_\_aeroflot.append(A)

*def* get(*self*):

        for i in *self*.\_\_aeroflot:

            print(i.get())

*#ВЫБОРКИ*

*#а) списка рейсов для заданного пункта назначения*

*def* get\_dest(*self*, *dest*):

        for i in *self*.\_\_aeroflot:

            if i.get\_des() == dest:

                print(i.get())

*#б) списка рейсов самолетов указанного типа*

*def* get\_type(*self*, *Type*):

        for i in *self*.\_\_aeroflot:

            if i.get\_type() == Type:

                print(i.get())

*#в) списка рейсов для заданного дня недели, время вылета которых находится в заданном интервале*

*def* get\_time(*self*, *day*, *time1*, *time2*):

        for i in *self*.\_\_aeroflot:

            if i.get\_time()[1] == day and *int*(i.get\_time()[0]) > *int*(time1) and *int*(i.get\_time()[0]) < *int*(time2):

                print(i.get())

*#Сортировка по полю времени вылета*

*def* sort\_by\_time(*self*):

*self*.\_\_aeroflot = sorted(*self*.\_\_aeroflot, *key*=*lambda* *flight*: flight.get\_time()[0])

Класс flights списка объектов типа Aeroflot с методами чтения, записи из файла, вывода в консоль, выборок по пункту назначения, типу, времени и сортировки списка по полю времени.

**Содержимое файла aero.py:**

from aero import \*

if \_\_name\_\_ == "\_\_main\_\_":

    a = Flights()

    while True:

        print("Главное меню")

        print("1) Добавить с консоли")

        print("2) Добавить из файла")

        print("3) Вывести в консоль")

        print("4) Вывести в файл")

        print("5) Выборка по назначению")

        print("6) Выборка по типу")

        print("7) Выборка по времени")

        print("8) Отсортировать по времени")

        print("0) <<< Выход")

        ans = *int*(input())

        match ans:

            case 1:

                des, num, ty, t1, day = input("Введите назначение, номер, тип, время вылета и день через пробел: ").split()

                m = Aeroflot(des,num,ty,t1,day)

                a.add(m)

            case 2:

                a.read("input.txt")

            case 3:

                a.get()

            case 4:

                a.save("output.txt")

            case 5:

                des = input("Введите назначение: ")

                a.get\_dest(des)

            case 6:

                ty = input("Введите тип: ")

                a.get\_type(ty)

            case 7:

                d,t1,t2 = input("Введите день, время от и до через пробел d\_t1\_t2: ").split()

                a.get\_time(d,t1,t2)

            case 8:

                a.sort\_by\_time()

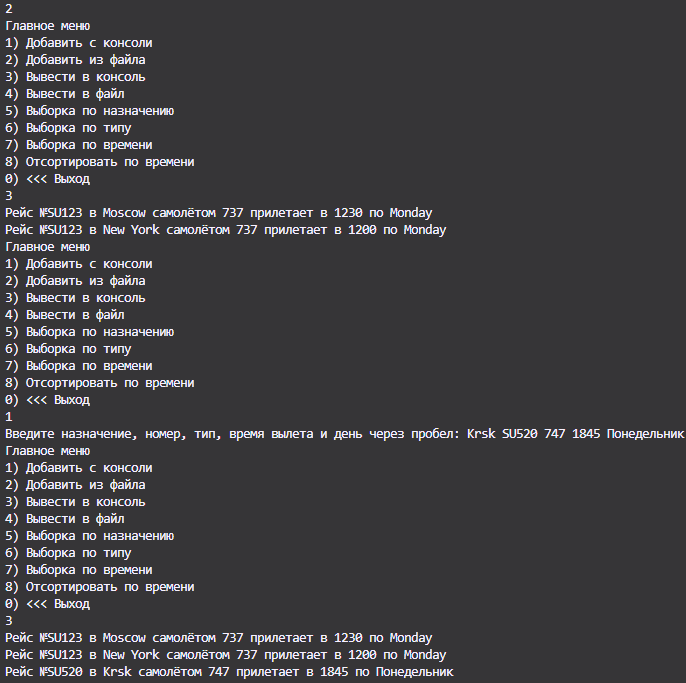
            case 0:

                break

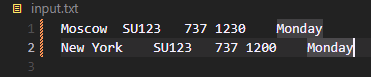
Меню функций модуля aero, для выбора используется оператор match-case.

# Тестирование

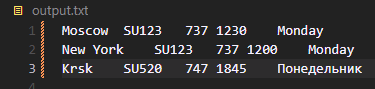
Тестирование чтения из файла input.txt, добавления одного рейса и вывода в файл output.txt**:**



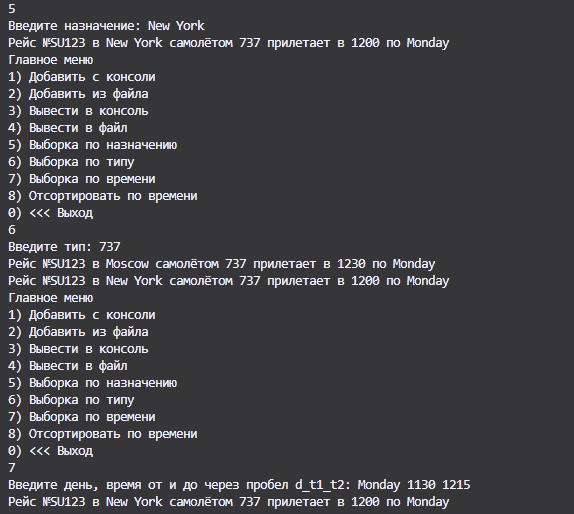
Содержимое файла input.txt:



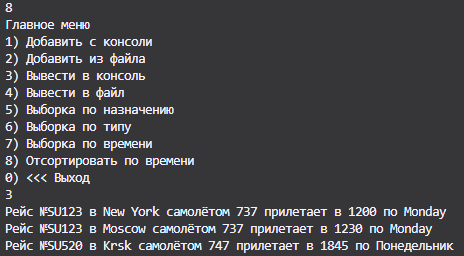
Содержимое файла ouput.txt:



Тестирование выборок:



Тестирование сортировки рейсов по полю времени:



# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. **В чем состоит сущность механизма наследования как принципа ООП?**
   * Наследование в объектно-ориентированном программировании позволяет создавать новые классы на основе существующих, унаследовав их свойства и методы.
2. **Как объявляется производный класс?**
   * Производный класс объявляется с использованием ключевого слова **class**, за которым следует имя производного класса, а в скобках указывается имя базового класса.
3. **Как осуществляется доступ к полям базового класса из производного при наследовании? Какие при этом существуют договорѐнности?**
   * Доступ к полям базового класса из производного осуществляется через обращение к ним через экземпляр производного класса. Согласно принципам инкапсуляции, поля базового класса могут быть закрытыми (private), и их доступ осуществляется через методы.
4. **Как происходит перегрузка методов при наследовании?**
   * Перегрузка методов при наследовании позволяет в производном классе предоставить свою реализацию метода с тем же именем, что и у базового класса.
5. **Каким образом выполняется инициализация наследуемых элементов?**
   * Инициализация наследуемых элементов происходит при вызове конструктора производного класса. Конструктор производного класса может вызывать конструктор базового класса с помощью ключевого слова **super()**.
6. **В каком порядке вызываются конструкторы класса при создании экземпляра производного класса?**
   * При создании экземпляра производного класса вызывается сначала конструктор базового класса, затем конструктор производного класса.
7. **Что такое абстрактный класс?**
   * Абстрактный класс в Python создается с использованием модуля **abc** и содержит хотя бы один абстрактный метод. Экземпляры абстрактного класса не могут быть созданы напрямую.
8. **Что такое множественное наследование?**
   * Множественное наследование позволяет классу наследовать свойства и методы от нескольких базовых классов. Python поддерживает множественное наследование, что означает, что класс может наследовать от нескольких классов.

# ВЫВОД

Наследование способствует повторному использованию кода, структурирует программу и улучшает понимание и её поддерживаемость. При наследовании важными аспектами являются доступ к элементам базового класса, перегрузка методов, инициализация наследуемых элементов, порядок вызова конструкторов, а также возможность создания абстрактных классов и использование множественного наследования в тех случаях, когда это оправдано. Все эти аспекты обеспечивают гибкость и расширяемость кода, что является важным принципом объектно-ориентированного программирования.