МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информационные технологии»

Тема: **Парадигмы программирования**

Студент гр. 3344 _____ Хангулян С. К.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2024

Цель работы

Целью работы является изучение основных парадигм программирования, ознакомление с ООП и написание кода с несколькими классами и собственными методами.

Задание

Вариант 3

Базовый класс - транспорт Transport:

class Transport:

Поля объекта класс Transport:

средняя скорость (в км/ч, положительное целое число)

максимальная скорость (в км/ч, положительное целое число)

цена (в руб., положительное целое число)

грузовой (значениями могут быть или True, или False)

цвет (значение может быть одной из строк: w (white), g(gray), b(blue)).

При создании экземпляра класса Transport необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

Автомобиль - Car:

class Car: #Наследуется от класса Transport

Поля объекта класс Car:

средняя скорость (в км/ч, положительное целое число)

максимальная скорость (в км/ч, положительное целое число)

цена (в руб., положительное целое число)

грузовой (значениями могут быть или True, или False)

цвет (значение может быть одной из строк: w (white), g(gray), b(blue)).

мощность (в Вт, положительное целое число)

количество колес (положительное целое число, не более 10)

При создании экземпляра класса Car необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод __str__():

Преобразование к строке вида: Car: средняя скорость <средняя скорость>, максимальная скорость <максимальная скорость>, цена <цена>, грузовой <грузовой>, цвет <цвет>, мощность <мощность>, количество колес <количество колес>.

Метод __add__():

Сложение средней скорости и максимальной скорости автомобиля. Возвращает число, полученное при сложении средней и максимальной скорости.

Метод __eq__():

Метод возвращает True, если два объекта класса равны, и False иначе. Два объекта типа Car равны, если равны количество колес, средняя скорость, максимальная скорость и мощность.

Самолет - Plane:

class Plane: #Наследуется от класса Transport

Поля объекта класс Plane:

средняя скорость (в км/ч, положительное целое число)

максимальная скорость (в км/ч, положительное целое число)

цена (в руб., положительное целое число)

грузовой (значениями могут быть или True, или False)

цвет (значение может быть одной из строк: w (white), g(gray), b(blue)).

грузоподъемность (в кг, положительное целое число)

размах крыльев (в м, положительное целое число)

При создании экземпляра класса Plane необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод __str__():

Преобразование к строке вида: Plane: средняя скорость <средняя скорость>, максимальная скорость <максимальная скорость>, цена <цена>, грузовой <грузовой>, цвет <цвет>, грузоподъемность <грузоподъемность>, размах крыльев <размах крыльев>.

Метод __add__():

Сложение средней скорости и максимальной скорости самолета. Возвращает число, полученное при сложении средней и максимальной скорости.

Метод __eq__():

Метод возвращает True, если два объекта класса равны по размерам, и False иначе. Два объекта типа Plane равны по размерам, если равны размах крыльев.

Корабль - Ship:

class Ship: #Наследуется от класса Transport

Поля объекта класс Ship:

средняя скорость (в км/ч, положительное целое число)

максимальная скорость (в км/ч, положительное целое число)

цена (в руб., положительное целое число)

грузовой (значениями могут быть или True, или False)

цвет (значение может быть одной из строк: w (white), g(gray), b(blue)).

длина (в м, положительное целое число)

высота борта (в м, положительное целое число)

При создании экземпляра класса Ship необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод __str__():

Преобразование к строке вида: Ship: средняя скорость <средняя скорость>, максимальная скорость <максимальная скорость>, цена <цена>, грузовой <грузовой>, цвет <цвет>, длина <длина>, высота борта <высота борта>.

Метод __add__():

Сложение средней скорости и максимальной скорости корабля. Возвращает число, полученное при сложении средней и максимальной скорости.

Метод __eq__():

Метод возвращает True, если два объекта класса равны по размерам, и False иначе. Два объекта типа Ship равны по размерам, если равны их длина и высота борта.

Необходимо определить список list для работы с транспортом: Автомобили:

class CarList – список автомобилей - наследуется от класса list.

Конструктор:

Вызвать конструктор базового класса.

Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

Метод append(p_object): Переопределение метода append() списка. В случае, если p_object - автомобиль, элемент добавляется в список, иначе выбрасывается исключение TypeError с текстом: Invalid type <тип_объекта p_object> (результат вызова функции type)

Метод print_colors(): Вывести цвета всех автомобилей в виде строки (нумерация начинается с 1):

<i> автомобиль: <color[i]>

<j> автомобиль: <color[j]> ...

Метод print_count(): Вывести количество автомобилей.

Самолеты:

class PlaneList – список самолетов - наследуется от класса list.

Конструктор:

Вызвать конструктор базового класса.

Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

Метод extend(iterable): Переопределение метода extend() списка. В случае, если элемент iterable - объект класса Plane, этот элемент добавляется в список, иначе не добавляется.

Метод print_colors(): Вывести цвета всех самолетов в виде строки (нумерация начинается с 1):

<i> самолет: <color[i]>

<j> самолет: <color[j]> ...

Метод total_speed(): Посчитать и вывести общую среднюю скорость всех самолетов.

Корабли:

class ShipList – список кораблей - наследуется от класса list.

Конструктор:

Вызвать конструктор базового класса.

Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

Метод append(p_object): Переопределение метода append() списка. В случае, если p_object - корабль, элемент добавляется в список, иначе выбрасывается исключение TypeError с текстом: Invalid type <тип_объекта p_object>

Метод print_colors(): Вывести цвета всех кораблей в виде строки (нумерация начинается с 1):

<i> корабль: <color[i]>

<j> корабль: <color[j]> ...

Метод print_ship(): Вывести те корабли, чья длина больше 150 метров, в виде строки:

Длина корабля №<i> больше 150 метров

Длина корабля №<j> больше 150 метров ...

Выполнение работы

Группа 1

В группе 1 находятся классы Transport, Car, Plane, Ship. Первый класс – родительский, остальные – дочерние.

В классе Transport определен метод __init__, инициализирующий следующие поля: average_speed, max_speed, price, cargo, color. В случае несоответствия какого-либо из полей появляется ошибка ValueError('Invalid value').

В дочерних классах с помощью метода __init__ и функции super() определены все поля родительского класса, а также по паре особенных полей.

Особенные поля класса Car - power, wheels. В случае несоответствия появляется ошибка ValueError('Invalid value').

Прочие методы класса Car:

- __str__ - возвращает строку в формате «Car: средняя скорость <средняя скорость>, максимальная скорость <максимальная скорость>, цена <цена>, грузовой <грузовой>, цвет <цвет>, мощность <мощность>, количество колес <количество колес>».
- __add__ - суммирует среднюю и максимальную скорость, возвращает полученное значение.
- __eq__ - сравнивает два экземпляра класса и возвращает True или False. Два объекта типа Car равны, если равны количество колес, средняя скорость, максимальная скорость и мощность.

Особенные поля класса Plane - load_capacity, wingspan. В случае несоответствия появляется ошибка ValueError('Invalid value').

Прочие методы класса Plane:

- __str__ - возвращает строку в формате «Plane: средняя скорость <средняя скорость>, максимальная скорость <максимальная

скорость>, цена <цена>, грузовой <грузовой>, цвет <цвет>, грузоподъемность <грузоподъемность>, размах крыльев <размах крыльев>».

- __add__ - суммирует среднюю и максимальную скорость, возвращает полученное значение.
- __eq__ - сравнивает два экземпляра класса и возвращает True или False. Два объекта типа Plane равны по размерам, если равны размах крыльев.

Особенные поля класса Ship - length, side_height. В случае несоответствия появляется ошибка ValueError('Invalid value').

Прочие методы класса Ship:

- __str__ - возвращает строку в формате «Ship: средняя скорость <средняя скорость>, максимальная скорость <максимальная скорость>, цена <цена>, грузовой <грузовой>, цвет <цвет>, длина <длина>, высота борта <высота борта>».
- __add__ - суммирует среднюю и максимальную скорость, возвращает полученное значение.
- __eq__ - сравнивает два экземпляра класса и возвращает True или False. Два объекта типа Ship равны по размерам, если равны их длина и высота борта.

Группа 2

В группе 2 находятся классы CarList, PlaneList, ShipList – дочерние классы стандартного класса list. Все инициализируются с помощью метода __init__ и super(). Каждый класс имеет свои особенные методы.

Особенные методы класса CarList:

- append – добавляет в конец списка экземпляр класса, иначе выводится ошибка TypeError с текстом: Invalid type <тип_объекта p_object>.
- print_colors – печатает порядковые номера всех автомобилей и их цвет путем итерации и обращения к полю color.
- print_count – выводит количество автомобилей (длину экземпляра).

Особенные методы класса PlaneList:

- extend - переопределение метода extend() списка. В случае, если элемент iterable - объект класса Plane, этот элемент добавляется в список, иначе не добавляется.
- print_colors – печатает порядковые номера всех самолетов и их цвет путем итерации и обращения к полю color.
- total_speed – считает и выводит суммарную среднюю скорость всех самолетов.

Особенные методы класса ShipList:

- append – добавляет в конец списка экземпляр класса, иначе выводится ошибка TypeError с текстом: Invalid type <тип_объекта p_object>.
- print_colors – печатает порядковые номера всех автомобилей и их цвет путем итерации и обращения к полю color.
- print_ship – печатает номера всех кораблей, длина которых больше 150 метров.

Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования

Ожидание	Выходные данные	Комментарии
70 200 50000 True w 70 200 50000 True w 100 4 Car: средняя скорость 70, максимальная скорость 200, цена 50000, грузовой True, цвет w, мощность 100, количество колес 4. 270 True 70 200 50000 True w 1000 150 Plane: средняя скорость 70, максимальная скорость 200, цена 50000, грузовой True, цвет w, грузоподъемность 1000, размах крыльев 150. 270 True 70 200 50000 True w 200 100 Ship: средняя скорость 70, максимальная скорость 200, цена 50000, грузовой True, цвет w, длина 200, высота борта 100. 270	70 200 50000 True w 70 200 50000 True w 100 4 Car: средняя скорость 70, максимальная скорость 200, цена 50000, грузовой True, цвет w, мощность 100, количество колес 4. 270 True 70 200 50000 True w 1000 150 Plane: средняя скорость 70, максимальная скорость 200, цена 50000, грузовой True, цвет w, грузоподъемность 1000, размах крыльев 150. 270 True 70 200 50000 True w 200 100 Ship: средняя скорость 70, максимальная скорость 200, цена 50000, грузовой True, цвет w, длина 200, высота борта 100. 270	Корректно

True 1 автомобиль: w 2 автомобиль: w 2 1 самолет: w 2 самолет: w 140 1 корабль: w 2 корабль: w Длина корабля №1 больше 150 метров Длина корабля №2 больше 150 метров	True 1 автомобиль: w 2 автомобиль: w 2 1 самолет: w 2 самолет: w 140 1 корабль: w 2 корабль: w Длина корабля №1 больше 150 метров Длина корабля №2 больше 150 метров	

Выводы

Были изучены основные парадигмы программирования, был написан код, создающий несколько классов и методы для работы с ними.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khangulyan_Sargis_lb1.py

```
class Transport:
    def __init__(self, average_speed, max_speed, price, cargo, color):
        if (isinstance(average_speed, int) and isinstance(max_speed,
int)
            and isinstance(price, int) and isinstance(cargo, bool)
            and average_speed > 0 and max_speed > 0 and price > 0
            and (color == "w" or color == "b" or color == "g")):
                self.average_speed = average_speed
                self.max_speed = max_speed
                self.price = price
                self.cargo = cargo
                self.color = color
        else: raise ValueError('Invalid value')



class Car(Transport):
    def __init__(self, average_speed, max_speed, price, cargo, color,
power, wheels):
        super().__init__(average_speed, max_speed, price, cargo, color)
        if (isinstance(power, int) and isinstance(wheels, int)
            and power > 0 and 0 < wheels < 11):
                self.power = power
                self.wheels = wheels
        else: raise ValueError('Invalid value')

    def __str__(self):
        return f"Car: средняя скорость {self.average_speed},
максимальная скорость {self.max_speed}, цена {self.price}, грузовой
{self.cargo}, цвет {self.color}, мощность {self.power}, количество
колес {self.wheels}."

    def __add__(self):
        return self.average_speed + self.max_speed

    def __eq__(self, other):
        if (self.wheels == other.wheels
            and self.average_speed == other.average_speed
            and self.max_speed == other.max_speed
            and self.power == other.power):
                return True
        return False



class Plane(Transport):
    def __init__(self, average_speed, max_speed, price, cargo, color,
load_capacity, wingspan):
        super().__init__(average_speed, max_speed, price, cargo, color)
        if (isinstance(load_capacity, int) and isinstance(wingspan,
int)
            and load_capacity > 0 and wingspan > 0):
```

```
                self.load_capacity = load_capacity
                self.wingspan = wingspan
        else: raise ValueError('Invalid value')

    def __str__(self):
        return f"Plane: средняя скорость {self.average_speed}, максимальная скорость {self.max_speed}, цена {self.price}, грузовой {self.cargo}, цвет {self.color}, грузоподъемность {self.load_capacity}, размах крыльев {self.wingspan}."

    def __add__(self):
        return self.average_speed + self.max_speed

    def __eq__(self, other):
        if (self.wingspan == other.wingspan):
            return True
        return False


class Ship(Transport):
    def __init__(self, average_speed, max_speed, price, cargo, color, length, side_height):
        super().__init__(average_speed, max_speed, price, cargo, color)
        if (isinstance(length, int) and isinstance(side_height, int)
            and length > 0 and side_height > 0):
                self.length = length
                self.side_height = side_height
        else: raise ValueError('Invalid value')

    def __str__(self):
        return f"Ship: средняя скорость {self.average_speed}, максимальная скорость {self.max_speed}, цена {self.price}, грузовой {self.cargo}, цвет {self.color}, длина {self.length}, высота борта {self.side_height}."

    def __add__(self):
        return self.average_speed + self.max_speed

    def __eq__(self, other):
        if (self.length == other.length and self.side_height == other.side_height):
            return True
        return False



class CarList(list):
    def __init__(self, name):
        super().__init__()
        self.name = name

    def append(self, p_object):
        if (isinstance(p_object, Car)):
            super().append(p_object)
        else:
            raise TypeError(f"Invalid type {type(p_object)}")
```

```
    def print_colors(self):
        for i in range(len(self)):
            print(f"{i+1} автомобиль: {self[i].color}")

    def print_count(self):
        print(len(self))



class PlaneList(list):
    def __init__(self, name):
        super().__init__()
        self.name = name

    def extend(self, iterable):
        filtered_iterable = []
        for i in iterable:
            if (isinstance(i, Plane)):
                filtered_iterable.append(i)
        super().extend(filtered_iterable)

    def print_colors(self):
        for i in range(len(self)):
            print(f"{i+1} самолет: {self[i].color}")

    def total_speed(self):
        sum = 0
        for i in range(len(self)):
            sum += self[i].average_speed
        print(sum)



class ShipList(list):
    def __init__(self, name):
        super().__init__()
        self.name = name

    def append(self, p_object):
        if (isinstance(p_object, Ship)):
            super().append(p_object)
        else:
            raise TypeError(f"Invalid type {type(p_object)}")

    def print_colors(self):
        for i in range(len(self)):
            print(f"{i+1} корабль: {self[i].color}")

    def print_ship(self):
        for i in range(len(self)):
            if self[i].length > 150:
                print(f'Длина корабля №{i+1} больше 150 метров')
```