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

Студентка гр. 3343 _____ Лобова Е. И.

Преподаватель _____ Иванов Д. И.

Санкт-Петербург
2024

Цель работы

Целью работы является освоение работы с объектно-ориентированным программированием, исключениями и создание программы на основе полученных знаний.

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

В ходе выполнения лабораторной работы была создана следующая иерархия классов:

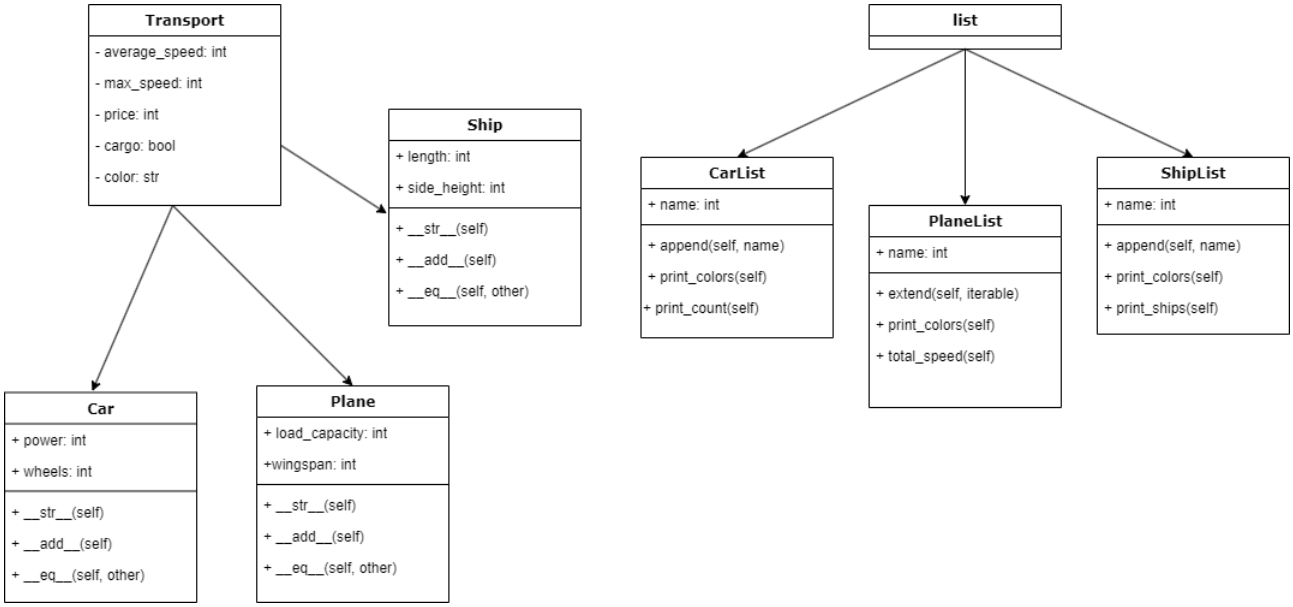

Рисунок 1 – Иерархия классов в программе

Были переопределены методы класса object __str__(), __add__(), __eq__() в классах Car, Plane, Ship. При вызове str(…) будет использоваться возвращаемое значение в переопределенном методе. Оператор == в вызывает метод __eq__(), который определяет поведение оператора равенства для объектов данного класса. Для классов CarList, PlaneList, ShipList были переопределены методы класса list append и extend, они будут работать корректно за счет использования super().

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии

1.	`transport = Transport(70, 200, 50000, True, 'w') #транспорт`
	`print(transport.average_speed, transport.max_speed, transport.price, transport.cargo, transport.color)`
	
	`car1 = Car(70, 200, 50000, True, 'w', 100, 4) #авто`
	`car2 = Car(70, 200, 50000, True, 'w', 100, 4)`
	`print(car1.average_speed, car1.max_speed, car1.price, car1.cargo, car1.color, car1.power, car1.wheels)`
	`print(car1.__str__())`
	`print(car1.__add__())`
	`print(car1.__eq__(car2))`
	
	`plane1 = Plane(70, 200, 50000, True, 'w', 1000, 150) #самолет`
	`plane2 = Plane(70, 200, 50000, True, 'w', 1000, 150)`
	`print(plane1.average_speed, plane1.max_speed, plane1.price, plane1.cargo, plane1.color, plane1.load_capacity, plane1.wingspan)`
	`print(plane1.__str__())`
	`print(plane1.__add__())`
	`print(plane1.__eq__(plane2))`
	
	`ship1 = Ship(70, 200, 50000, True, 'w', 200, 100) #корабль`
	`ship2 = Ship(70, 200, 50000, True, 'w', 200, 100)`
	`print(ship1.average_speed, ship1.max_speed, ship1.price, ship1.cargo, ship1.color, ship1.length, ship1.side_height)`
	`print(ship1.__str__())`
	`print(ship1.__add__())`
	`print(ship1.__eq__(ship2))`
	
	`car_list = CarList(Car) #список авто`
	`car_list.append(car1)`
	`car_list.append(car2)`
	`car_list.print_colors()`
	`car_list.print_count()`
		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
		True
		1 автомобиль: w
		2 автомобиль: w
		2
			Для корректных данных работает верно.

	plane_list = PlaneList(Plane)
	#список самолетов
	plane_list.extend([plane1, plane2])
	plane_list.print_colors()
	plane_list.total_speed()
	
	ship_list = ShipList(Ship)
	#список кораблей
	ship_list.append(ship1)
	ship_list.append(ship2)
	ship_list.print_colors()
	ship_list.print_ship()
		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	

2.	try: #неправильные данные для самолета
	    plane1 = Plane(-70, 200, 50000, True, 'w', 1000, 150)
	except (TypeError, ValueError):
	    print('OK')
	
	try:
	    plane1 = Plane(70, -200, 50000, True, 'w', 1000, 150)
	except (TypeError, ValueError):
	    print('OK')
	
	try:
	    plane1 = Plane(70, 200, -50000, True, 'w', 1000, 150)
	except (TypeError, ValueError):
	    print('OK')
	try:
	    car1 = Car(70, 200, 50000, -1, 'w', 100, 4)
	except (TypeError, ValueError):
	    print('OK')
	
	try:
	    car1 = Car(70, 200, 50000, True, -1, 100, 4)
	except (TypeError, ValueError):
	    print('OK')
	
	try:
	    car1 = Car(70, 200, 50000, True, 'w', -100, 4)
	except (TypeError, ValueError):
	    print('OK')
	try:
	    ship1 = Ship('a', 200, 50000, True, 'w', 200, 100)
	except (TypeError, ValueError):
	    print('OK')
	
	try:
	    ship1 = Ship(70, 'a', 50000, True, 'w', 200, 100)
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
			Для неверных данных работает также корректно.

	except (TypeError, ValueError):
	    print('OK')
	
	try:
	    ship1 = Ship(70, 200, 'a', True, 'w', 200, 100)
	except (TypeError, ValueError):
	    print('OK')		

Выводы

Были изучены основы объектно-ориентированного программирования и его реализация на Python. Также были изучены исключения: конструкции try-except-else-finally и raise для самостоятельной генерации исключений.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class Transport:
    def __init__(self, average_speed, max_speed, price, cargo, color):
        if any((type(x) is not int) for x in [average_speed, max_speed, price])\
                or (type(cargo) is not bool) or (type(color) is not str):
            raise ValueError("Invalid value")
        if average_speed <= 0 or max_speed <= 0 or price <= 0 or not(color == 'w' or color == 'g' or color == 'b'):
            raise ValueError("Invalid value")
        self.average_speed = average_speed
        self.max_speed = max_speed
        self.price = price
        self.cargo = cargo
        self.color = color
    pass

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 (type(power) is not int) or (type(wheels) is not int):
            raise ValueError("Invalid value")
        if power <= 0 or not(0 < wheels <= 10):
            raise ValueError("Invalid value")
        self.power = power
        self.wheels = wheels

    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):
        return self.wheels == other.wheels and self.average_speed == other.average_speed and self.max_speed == other.max_speed and self.power == other.power
    pass

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 (type(load_capacity) is not int) or (type(wingspan) is not int):
```

```
                raise ValueError("Invalid value")
            if load_capacity <= 0 or wingspan <= 0:
                raise ValueError("Invalid value")
            self.load_capacity = load_capacity
            self.wingspan = wingspan

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

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

        def __eq__(self, other):
            return self.wingspan == other.wingspan
        pass

    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 (type(length) is not int) or (type(side_height) is not
int):
                raise ValueError("Invalid value")
            if length <= 0 or side_height <= 0:
                raise ValueError("Invalid value")
            self.length = length
            self.side_height = side_height

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

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

        def __eq__(self, other):
            return self.length == other.length and self.side_height ==
other.side_height
        pass

    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("Invalid type {type(p_object)}")
```

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

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

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

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

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

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

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("Invalid type {type(p_object)}")

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

    def print_ship(self):
        s = ""
        for i in range(len(self)):
            if self[i].length > 150:
                s += f"Длина корабля №{i+1} больше 150 метров\n"
        print(s[:-1])
    pass
```