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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Парадигмы программирования

Студент гр. 3342 _____ Галеев А.Д.

Преподаватель _____ Шалагинов И.В.

Санкт-Петербург

2024

Цель работы

Изучить понятие парадигмы программирования, понять способы их работы и применения.

Задание

Реализовать некоторые методы в программе, которые будут обрабатывать данные о трех различных видах транспорта и выводить обработанную информацию пользователю

Основные теоретические положения

Для решения задач в программе использовались функции стандартной библиотеки языка Python

Выполнение работы

Иерархия классов:

Transport > Car > Carlist; Transport > Plane > Planelist; Transport > Ship > Shiplist

Методы, которые переопределены:

__init__: Переопределен во всех классах для инициализации объектов с определенными атрибутами

__str__: Переопределен в каждом классе для представления объекта в виде строки

__add__: Переопределен в классах Car, Plane и Ship для определения операции сложения, которая возвращает сумму средней и максимальной скоростей

__eq__: Переопределен в классах Car и Plane для сравнения объектов на равенство. Для класса Car сравниваются мощность и количество колес, а для Plane - размах крыльев

Случаи использования методов __str__ и __eq__:

Метод __str__ используется, когда объекты классов Car, Plane и Ship преобразуются в строки, например, при вызове функции print() или при преобразовании объекта в строку

Метод __eq__ используется, когда происходит сравнение объектов на равенство

Переопределенные методы класса list для CarList, PlaneList и ShipList будут работать в соответствии с их спецификацией, так как эти классы являются подклассами list. Например, методы append() и extend() будут добавлять элементы в список, а методы print_colors() и print_count() будут работать как описано внутри соответствующих классов.

Тестирование

Результаты тестирования представлены в табл. 1

Таблица 1 – Результаты тестирования

№ проверки	Входные данные	Выходные данные

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))
		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

	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()
	
	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()
		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
		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

Было изучено понятие парадигм программирования.

Разработана программа выполняющая обработку данных о различных видах транспорта.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1

```
class Transport:
    def __init__(self, average_speed, max_speed, price, cargo, color):
        if not(isinstance(average_speed, int) and average_speed > 0 and
isinstance(max_speed, int) and max_speed > 0 and isinstance(price, int)
and price > 0 and isinstance(cargo, bool) and color in ['w', 'g', 'b']):
            raise ValueError('Invalid value')
        self.average_speed = average_speed
        self.max_speed = max_speed
        self.price = price
        self.cargo = cargo
        self.color = color
class Car(Transport):
    def __init__(self, average_speed, max_speed, price, cargo, color,
power, wheels):
        if not(isinstance(power, int) and power > 0 and
isinstance(wheels, int) and 0 < wheels <= 10):
            raise ValueError('Invalid value')
        super().__init__(average_speed, max_speed, price, cargo, color)
        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 isinstance(other, Car) and self.wheels == other.wheels and
self.average_speed == other.average_speed and self.max_speed ==
other.max_speed and self.power == other.power
class Plane(Transport):
    def __init__(self, average_speed, max_speed, price, cargo, color,
load_capacity, wingspan):
        if not(isinstance(load_capacity, int) and load_capacity > 0 and
isinstance(wingspan, int) and wingspan > 0):
            raise ValueError('Invalid value')
        super().__init__(average_speed, max_speed, price, cargo, color)
        self.load_capacity = load_capacity
        self.wingspan = wingspan

    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):
        return isinstance(other, Plane) and self.wingspan ==
other.wingspan


class Ship(Transport):
    def __init__(self, average_speed, max_speed, price, cargo, color,
length, side_height):
        if not(isinstance(length, int) and length > 0 and
                isinstance(side_height, int) and side_height > 0):
            raise ValueError('Invalid value')
        super().__init__(average_speed, max_speed, price, cargo, color)
        self.length = length
        self.side_height = side_height

    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):
        return isinstance(other, Ship) and self.length == other.length
and self.side_height == other.side_height


class CarList(list):
    def __init__(self, name):
        super().__init__()
        self.name = name
    def append(self, p_object):
        if not isinstance(p_object, Car):
            raise TypeError(f"Invalid type {type(p_object)}")
        super().append(p_object)

    def print_colors(self):
        for i, car in enumerate(self, start=1):
            print(f"{i} автомобиль: {car.color}")

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


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

```
    def extend(self, iterable):
        for item in iterable:
            if isinstance(item, Plane):
                self.append(item)

    def print_colors(self):
        for i, plane in enumerate(self, start=1):
            print(f"{i} самолет: {plane.color}")

    def total_speed(self):
        total = sum(plane.average_speed for plane in self)
        print(total)


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

    def append(self, p_object):
        if not isinstance(p_object, Ship):
            raise TypeError(f"Invalid type {type(p_object)}")
        super().append(p_object)

    def print_colors(self):
        for i, ship in enumerate(self, start=1):
            print(f"{i} корабль: {ship.color}")

    def print_ship(self):
        for i, ship in enumerate(self, start=1):
            if ship.length > 150:
                print(f"Длина корабля №{i} больше 150 метров")
```