**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))  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() | 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 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")