Практическое занятие № 16

1. Наименование практического занятия: составление программ с использованием ООП.

2. Цели практического занятия: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с ООП в IDE PyCharm Community

Постановка задачи: Создайте класс "Машина" с атрибутами "марка", "модель" и "год выпуска".

Напишите метод, который выводит информацию о машине в формате "Марка:

марка, Модель: модель, Год выпуска: год".

блок 2:Создайте класс "Автомобиль", который содержит информацию о марке, модели и

годе выпуска. Создайте класс "Грузовик", который наследуется от класса

"Автомобиль" и содержит информацию о грузоподъемности. Создайте класс

"Легковой автомобиль", который наследуется от класса "Автомобиль" и содержит

информацию о количестве пассажиров

блок 3:Для задачи из блока 1 создать две функции, save\_def и load\_def, которые позволяют

сохранять информацию из экземпляров класса (3 шт.) в файл и загружать ее обратно.

Использовать модуль pickle для сериализации и десериализации объектов Python в

бинарном формате.

Тип алгоритма: линейный

Текст программы: import pickle

class Car:

    def \_\_init\_\_(self, brand, model, year):

        self.brand = brand

        self.model = model

        self.year = year

    def car\_info(self):

        return f"Бренда: {self.brand}, Модель: {self.model}, Год выпуска: {self.year}"

#сохранение в файл

def save\_def(filename, obj):

    with open(filename, 'wb') as f:

        pickle.dump(obj, f)

#подгрузка объекта из файла

def load\_def(filename):

    with open(filename, 'rb') as f:

        obj = pickle.load(f)

    return obj

#экземпляры

car1 = Car("Toyota", "Corolla", 2020)

car2 = Car("Honda", "Civic", 2019)

car3 = Car("Ford", "Mustang", 1967)

cars = [car1, car2, car3]

save\_def('cars.pickle', cars)

loaded\_cars = load\_def('cars.pickle')

for car in loaded\_cars:

    print(car.car\_info())

class Car:

    def \_\_init\_\_(self, brand, model, year):

        self.brand = brand

        self.model = model

        self.year = year

    def car\_info(self):

        return f"Бренда: {self.brand}, Модель: {self.model}, Год выпуска: {self.year}"

class Truck(Car):

    def \_\_init\_\_(self, brand, model, year, load\_capacity):

        super().\_\_init\_\_(brand, model, year)

        self.load\_capacity = load\_capacity

    def truck\_info(self):

        return f"{self.car\_info()}, Грузоподъемность: {self.load\_capacity} т"

class PassengerCar(Car):

    def \_\_init\_\_(self, brand, model, year, num\_passengers):

        super().\_\_init\_\_(brand, model, year)

        self.num\_passengers = num\_passengers

    def passenger\_car\_info(self):

        return f"{self.car\_info()}, Количество пассажиров: {self.num\_passengers}"

truck = Truck("Volvo", "FH16", 2019, 42)

print(truck.truck\_info())

passenger\_car = PassengerCar("Mercedes-Benz", "S-Class", 2021, 5)

print(passenger\_car.passenger\_car\_info())

протокол работы программы: Бренда: Toyota, Модель: Corolla, Год выпуска: 2020

Бренда: Honda, Модель: Civic, Год выпуска: 2019

Бренда: Ford, Модель: Mustang, Год выпуска: 1967

Бренда: Volvo, Модель: FH16, Год выпуска: 2019, Грузоподъемность: 42 т

Бренда: Mercedes-Benz, Модель: S-Class, Год выпуска: 2021, Количество пассажиров: 5

Вывод: В процессе выполнения практического занятие выработал навыки составление программ с использованием ООП и работы с ними в IDE PyCharm Community. Выполнены разработка кода, отладка ,тестирование, оптимизация, программного кода. Готовые программные коды выложены на GitHub