Практическое занятие № 17

Тема: составление программ с использованием ООП.

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

1) Постановка задачи:

27. Создайте класс «Студент», который имеет атрибуты имя, фамилия и оценки. Добавьте методы для вычисления среднего балла и определения, является ли студент отличником.

Тип алгоритма: ветвящийся.

Текст программы:

```
# Вариант 27

class Student:
    def __init__(self, name: str, surname: str, grades: list):
        self.name = name
        self.surname = surname
        self.grades = grades

    def gpa(self):
        return sum(lst) / len(lst) if (lst := self.grades) else 0.0

    def is_excellent(self):
        return 'Отличник' if set(self.grades) == {5} else 'Не отличник'


excellent = Student('Иван', 'Иванов', [5, 5, 5, 5, 5])
poor = Student('Петр', 'Петров', [2, 3, 2, 3, 2])
no_grades = Student('Анна', 'Сидорова', [])

print('Студент с отличными оценками:')
print(f'Средний балл - {excellent.gpa()} -> {excellent.is_excellent()}\n')

print('Студент с плохими оценками:')
print(f'Средний балл - {poor.gpa()} -> {poor.is_excellent()}\n')

print('Студент без оценок:')
print(f'Средний балл - {no_grades.gpa()} -> {no_grades.is_excellent()}\n')
```

Протокол работы программы:

Студент с отличными оценками:

Средний балл - 5.0 -> Отличник

Студент с плохими оценками:

Средний балл - 2.4 -> Не отличник

Студент без оценок:

Средний балл - 0.0 -> Не отличник

Process finished with exit code 0

2) **Постановка задачи**:

27. 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.

Тип алгоритма: линейный.

Текст программы:

```
class Automobile:

    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year


class Truck(Automobile):

    def __init__(self, brand, model, year, cargo_capacity):
        super().__init__(brand, model, year)
        self.cargo_capacity = cargo_capacity
```

```
    def carry_cargo(self):
        print("This truck can carry", self.cargo_capacity, "tons of cargo.")


class Car(Automobile):

    def __init__(self, brand, model, year, num_passengers):
        super().__init__(brand, model, year)
        self.num_passengers = num_passengers

    def carry_passengers(self):
        print("This car can carry", self.num_passengers, "passengers.")


# создание объектов классов Truck и PassengerCar
truck = Truck("Volvo", "FH16", 2022, 20)
passenger_car = Car("Tesla", "Model S", 2022, 5)

# вызов методов класса Truck
print(truck.brand)
print(truck.model)
print(truck.year)
truck.carry_cargo()
print()

# вызов методов класса PassengerCar
print(passenger_car.brand)
print(passenger_car.model)
print(passenger_car.year)
passenger_car.carry_passengers()
```

Протокол работы программы:

Volvo

FH16

2022

This truck can carry 20 tons of cargo.

Tesla

Model S

2022

This car can carry 5 passengers.

Process finished with exit code 0

Вывод: в процессе выполнения практического занятия я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрел навыки составления программ с ООП в IDE PyCharm Community.