Студентка группы ИС-23 Пономарева М.П.

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.

Постановка задачи.

1) Разработать программу, в которой создается класс «Круг», который имеет атрибут радиуса и методы для вычисления площади, длины окружности и диаметра.
2) Разработать программу, в которой создается базовый класс "Транспортное средство" и его наследование для создания классов "Автомобиль" и "Мотоцикл". В классе "Транспортное средство" будут общие свойства, такие как максимальная скорость и количество колес, а классы-наследники будут иметь свои уникальные свойства и методы

Текст программы:

1)

```
# Блок 1, вариант 18
# Создайте класс "круг", который имеет атрибут радиуса и
# методы для вычисления площади, длины окружности и диаметра.

import math

class Circle:
    def __init__ (self, radius):
        self.radius = radius

    def r(self):
        self.r = self.radius
        return self.r

    def one(self):
        self.sqrt = (self.radius * self.radius)*3.14
        return self.sqrt
    def two(self):
        self.d =  (self.radius * 2)
```

```
        return self.d
    def three(self):
        self.dlina = (self.d * 3.14)
        return self.dlina

CircleOne = Circle(3)
print ('Радиус =', CircleOne.r())
print ('Площадь =', CircleOne.one())
print ('Диаметр =', CircleOne.two())
print ('Длина окружности =', CircleOne.three())
```

2)

```
# Блок 2, вариант 18
# Создание базового класса "Транспортное средство" и его наследование для создания
# классов "Автомобиль" и "Мотоцикл". В классе "Транспортное средство" будут
# общие свойства, такие как максимальная скорость и количество колес, а классы-
# наследники будут иметь свои уникальные свойства и методы.

class Transport:
    def __init__(self, max_speed, wheels):
        self.max_speed = max_speed
        self.wheels = wheels

class Car(Transport):
    def __init__(self, max_speed, wheels, brand, model):
        super().__init__(max_speed, wheels)
        self.brand = brand
        self.model = model
        self.fuel = 0

    def drive(self, distance):
        fuel_consumption = distance / 10.0
        if fuel_consumption <= self.fuel:
            self.fuel -= fuel_consumption
            return f"{self.brand} {self.model} проехал {distance}km"
        else:
            return "Недостаточно топлива"

    def add_fuel(self, amount):
        self.fuel += amount
```

```
class Motorcycle(Transport):
    def __init__(self, max_speed, wheels, brand, model, helmet):
        super().__init__(max_speed, wheels)
        self.brand = brand
        self.model = model
        self.helmet = helmet

    def wear_helmet(self):
        return f"{self.brand} {self.model} носит {self.helmet}"

# Пример использования классов
car = Car(200, 4, "Toyota", "Camry")
print(car.drive(50)) # Недостаточно топлива
car.add_fuel(10)
print(car.drive(50)) # Toyota Camry проехал 50km

motorcycle = Motorcycle(150, 2, "Honda", "CBR", "полную защиту")
print(motorcycle.wear_helmet()) # Honda CBR носит полную защиту
```

Протокол работы программы:

1)

Радиус = 3
Площадь = 28.26
Диаметр = 6
Длина окружности = 18.84

Process finished with exit code 0

2)

Недостаточно топлива
Toyota Camry проехал 50km
Honda CBR носит полную защиту

Process finished with exit code 0

Вывод: в процессе выполнения практического занятия я закрепила усвоенные знания, понятия, алгоритмы,
основные принципы составления программ, приобрела навыки составление
программ с ООП в IDE PyCharm Community.
Выполнены разработка кода, отладка, тестирование, оптимизация программного кода.
Готовые программные коды выложены на GitHub.