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: Создайте класс «Календарь», который имеет атрибуты год, месяц и день. Добавьте методы для определения дня недели, проверки на високосный год и определения количества дней в месяце.

Текст программы:

```
# Вариант 4
# Создайте класс «Календарь», который имеет атрибуты
год, месяц и день. Добавьте
# методы для определения дня недели, проверки на
високосный год и определения
# количества дней в месяце.


class Kalendar:
    def __init__(self, year, months, day):
        self.year = year
        self.months = months
        self.day = day

    def week_definition(self):
        days = {1: "Понедельник", 2: "Вторник", 3:
"Среда", 4: "Четверг", 5: "Пятница", 6: "Суббота",
7: "Воскресенье"}
        if self.day > 7:
            now_day = self.day // 7
            return f"Сегодняшний день недели:
{days[now_day]}"
        return f"Сегодняшний день недели:
{days[self.day]}"
```

```
    def year_v_definition(self):
        if self.year % 4 == 0:
            return self.year, "Является високосным
годом"
        else:
            return self.year, "Не является високосным
годом"

    def number_days_in_months(self):
        # Функция для определения количества дней в
месяце
        self.months_days = {"Январь": 31, "Февраль":
28, "Апрель": 30, "Март": 31, "Май": 31,
                            "Июнь": 30, "Июль": 31,
"Август": 31, "Сентябрь": 30, "Октябрь": 31,
"Декабрь": 31}
        if self.year % 4 == 0:
            self.months_days["Февраль"] += 1
            return "Количество дней в месяце: " +
str(self.months_days[self.months])
        else:  # Обратное if.
            return "Количество дней в месяце: " +
str(self.months_days[self.months])


obj_1 = Kalendar(2000, "Январь", 20)
print(obj_1.number_days_in_months())
print(obj_1.week_definition())
print(obj_1.year_v_definition())

print()

obj_2 = Kalendar(2008, "Февраль", 6)
print(obj_2.number_days_in_months())
print(obj_2.week_definition())
print(obj_2.year_v_definition())
```

Протокол работы программы:
Количество дней в месяце: 31
Сегодняшний день недели: Вторник
(2000, 'Является високосным годом')

Количество дней в месяце: 29
Сегодняшний день недели: Суббота
(2008, 'Является високосным годом')

Process finished with exit code 0

Постановка задачи 2: Создайте класс "Животное", который содержит информацию о виде и возрасте животного. Создайте классы "Собака" и "Кошка", которые наследуются от класса "Животное" и содержат информацию о породе

Текст программы:

```
# Вариант 4
# Создайте класс "Животное", который содержит информацию о виде и возрасте
# животного. Создайте классы "Собака" и "Кошка", которые наследуются от класса
# "Животное" и содержат информацию о породе.


class Animals:
    def __init__(self, vid, age):
        self.vid = vid
        self.age = age


class Dog(Animals):
    def __init__(self, vid, age, poroda):
        super().__init__(vid, age)
        self.poroda = poroda

    def print(self):
```

```
        return f"Вид животного: {self.vid}\nВозраст:
{self.age}\nПорода животного: {self.poroda}"


class Cat(Animals):
    def __init__(self, vid, age, poroda):
        super().__init__(vid, age)
        self.poroda = poroda

    def print(self):
        return f"Вид животного: {self.vid}\nВозраст:
{self.age}\nПорода животного: {self.poroda}"


animal_one = Dog("Собака", 7, "Немецкая овчарка")
print(animal_one.print())

print()

animal_two = Cat("Кошка", 12, 'Сиамская')
print(animal_two.print())
```

Протокол работы программы:
Вид животного: Собака
Возраст: 7
Порода животного: Немецкая овчарка

Вид животного: Кошка
Возраст: 12
Порода животного: Сиамская

Process finished with exit code 0