Студентка группы ИС-23 Никифорова З.А.

Практическое задание № 16

Наименование: составление программ с классами в IDE PyCharm Community

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

Задача 1. Создайте класс «Счетчик», который имеет атрибут текущего значения и методы для инкремента и декремента значения.

```
# 20. Создайте класс «Счетчик», который имеет атрибут текущего значения и методы для инкремента и декремента значения.

class Counter:

    def __init__(self, count):
        self.count = count

    def upcount(self, plus):
        self.count += plus
        print(f'{self.count-plus} + {plus} = {self.count}')

    def discount(self, minus):
        self.count -= minus
        print(f'{self.count+minus} - {minus} = {self.count}')


c1 = Counter(5)
c1.upcount(2)
c1.discount(3)
c1.upcount(-8)
```

Протокол работы программы:

5 + 2 = 7

7 - 3 = 4

4 + -8 = -4

Задача 2. Создание базового класса "Работник" и его наследование для создания классов "Менеджер" и "Инженер". В классе "Работник" будут общие методы, такие как "работать" и "получать зарплату", а классы-наследники будут иметь свои уникальные методы и свойства, такие как "управлять командой" и "проектировать системы".

```
# 20. Создание базового класса "Работник" и его наследование для создания классов "Менеджер" и "Инженер". В классе "Работник" будут общие методы, такие как "работать" и "получать зарплату", а классы-наследники будут иметь свои уникальные методы и свойства, такие как "управлять командой" и "проектировать системы".

class Worker:

    def __init__(self, name, surname):
        self.name = name
```

```
        self.surname = surname

    def work(self):
        print(f"{self.name} {self.surname} выполняет свою работу.")

    def get_salary(self):
        print(f"{self.name} {self.surname} получает зарплату.")


class Manager(Worker):

    def __init__(self, name, surname, department):
        super().__init__(name, surname)
        self.department = department

    def commanding(self):
        print(f"{self.name} {self.surname} руководит командой в отделе
{self.department}.")


class Engineer(Worker):


    def __init__(self, name, surname, specialization):
        super().__init__(name, surname)
        self.specialization = specialization

    def system_projecting(self):
        print(f"{self.name} {self.surname} проектирует системы в области
{self.specialization}.")


worker1 = Worker("Иван", "Петров")
worker1.work()
worker1.get_salary()

manager1 = Manager("Василий", "Сидоров", "Маркетинг")
manager1.work()
manager1.get_salary()
manager1.commanding()

engineer1 = Engineer("Ольга", "Кузнецова", "Программное обеспечение")
engineer1.work()
engineer1.get_salary()
engineer1.system_projecting()
```

Протокол работы программы:

Иван Петров выполняет свою работу.
Иван Петров получает зарплату.
Василий Сидоров выполняет свою работу.
Василий Сидоров получает зарплату.
Василий Сидоров руководит командой в отделе Маркетинг.
Ольга Кузнецова выполняет свою работу.
Ольга Кузнецова получает зарплату.
Ольга Кузнецова проектирует системы в области Программное обеспечение.

Задача 3. Для задачи из блока 1 создать две функции, save_def u 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.

```
import pickle
class Counter:
    def __init__(self, count):
        self.count = count
    def upcount(self, plus):
        self.count += plus
        print(f'{self.count-plus} + {plus} = {self.count}')
    def discount(self, minus):
        self.count -= minus
        print(f'{self.count+minus} - {minus} = {self.count}')
    def save_def(self, filename):
        with open(filename, "wb") as file:
            pickle.dump(self, file)
    def load_def(self, filename):
        with open(filename, "rb") as file:
            loaded_object = pickle.load(file)
            self.count = loaded_object.count

c1 = Counter(5)
c2 = Counter(6)
c3 = Counter(7)

c1.save_def("c1.pickle")
c2.save_def("c2.pickle")
c3.save_def("c3.pickle")

loaded_object1 = Counter("")
loaded_object1.load_def("c1.pickle")
print(f"Загруженный объект 1: {loaded_object1.count}")

loaded_object2 = Counter("")
loaded_object2.load_def("c2.pickle")
print(f"Загруженный объект 2: {loaded_object2.count}")

loaded_object3 = Counter("")
loaded_object3.load_def("c3.pickle")
print(f"Загруженный объект 3: {loaded_object3.count}")
```

Протокол работы программы:

Загруженный объект 1: 5
Загруженный объект 2: 6
Загруженный объект 3: 7

Создание файлов c1.pickle, c2.pickle, c3.pickle

Вывод: в процессе выполнения практического занятие, я выработала навыки составления программ с классами в IDE PyCharm Community. Выполнены разработка кода, отладка, тестирование, оптимизация, программного кода. Готовые программные коды выложены на GitHub.