Практическое занятие №16

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:

Создать класс «Банк», который имеет атрибуты суммы денег и процентной ставки. Добавить методы для вычисления процентных начислений и снятия денег.

Тип алгоритма: линейный

Текст программы:

```
# Вариант 15

# Создайте класс «Банк», который имеет атрибуты суммы денег и процентной
ставки.

# Добавьте методы для вычисления процентных начислений и снятия денег (на
год)




class Bank:

    def __init__(self, sum, percent):

        self.sum = sum

        self.percent = percent



    def calculation(self):

        return self.sum * (self.percent/100)



    def withdraw(self):

        return f"Процентные начисления >> {self.calculation()}, вы сможете
снять {self.calculation() + self.sum} "



money1 = Bank(1000, 12)
```

```
money2 = Bank(2000, 7)

money3 = Bank(670, 20)




print(money1.__dict__)

print(money2.__dict__)

print(money3.__dict__)


print(money1.calculation())

print(money2.calculation())

print(money3.calculation())


print(money1.withdraw())

print(money2.withdraw())

print(money3.withdraw())
```

Протокол работы программы:

{'sum': 1000, 'percent': 12}

{'sum': 2000, 'percent': 7}

{'sum': 670, 'percent': 20}

120.0

140.0

134.0

Процентные начисления >> 120.0, вы сможете снять 1120.0

Процентные начисления >> 140.0, вы сможете снять 2140.0

Процентные начисления >> 134.0, вы сможете снять 804.0

Process finished with exit code 0

Постановка задачи №2:

Создать класс "Животное", который содержит информацию о виде и возрасте животного. Создать классы "Собака" и "Кошка", которые наследуются от класса "Животное" и содержат информацию о породе.

Тип алгоритма: линейный

Текст программы:

```
# Вариант 15
# 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 Animal:
    def __init__(self, species, age):
        self.species = species
        self.age = age

class Dog(Animal):
    def __init__(self, species, age, breed):
        super().__init__(species, age)
        self.breed = breed

class Cat(Animal):
    def __init__(self, species, age, breed):
        super().__init__(species, age)
        self.breed = breed


dog = Dog('Собака', 1, 'Такса')
cat = Cat('Кошка', 3, 'Мейн-кун')
print(dog.__dict__)
print(cat.__dict__)
```

Протокол работы программы:

{'species': 'Собака', 'age': 1, 'breed': 'Такса'}

{'species': 'Кошка', 'age': 3, 'breed': 'Мейн-кун'}

Process finished with exit code 0

Постановка задачи №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, циклический

Текст программы:

```
# Вариант 15
# 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.


import pickle
class Bank:
    def __init__(self, sum, percent):
        self.sum = sum
        self.percent = percent

def save_bank(money, filename):
    with open(filename, 'wb') as f:
        pickle.dump(money, f)

def load_bank(filename):
    with open(filename, 'rb') as f:
        return pickle.load(f)

money1 = Bank(1000, 12)
money2 = Bank(2000, 7)
money3 = Bank(670, 20)

money = [money1, money2, money3]
save_bank(money, 'Bank.bin')

# Загружаем информацию о студентах из файла
loaded_money = load_bank('Bank.bin')

# Выводим информацию о загруженных студентах
for money in loaded_money:
    print(f'Сумма: {money.sum}, Процент: {money.percent}')
```

Протокол работы программы:

Сумма: 1000, Процент: 12

Сумма: 2000, Процент: 7

Сумма: 670, Процент: 20

Process finished with exit code 0

Вывод: В процессе выполнения практического занятия закрепила усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ, работы с ООП в IDE PyCharm Community.
Выполнены разработка кода, отладка, тестирование, оптимизация программного кода.

Готовые программные коды выложены на GitHub.