Студентка группы ИС-22 Ким Я.С.

**Практическое занятие №16**

**Тема:** Составление программ с использованием ООП в IDE PyCharm Community.

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

Блок 1

**Постановка задачи:** Создайте класс «Банк», который имеет атрибуты суммы денег и процентной ставки. Добавьте методы для вычисления процентных начислений и снятия денег.

**Текст программы:**

```
# Создайте класс «Банк», который имеет атрибуты суммы денег и процентной
ставки.
# Добавьте методы для вычисления процентных начислений и снятия денег.

class Bank:
    def __init__(self, balance, interest_rate):
        self.balance = balance
        self.interest_rate = interest_rate

    def calculate_interest(self, time_period):
        interest = self.balance * (self.interest_rate / 100) * time_period
        return interest

    def withdraw(self, amount):
        if amount > self.balance:
            raise ValueError("Недостаточно средств на балансе.")
        self.balance -= amount


bal = float(input("Введите текущий баланс: "))
int_r = float(input("Введите текущую ставку: "))
my_bank = Bank(bal, int_r)
print(f"Текущий баланс: {my_bank.balance:.2f} руб.")


time_p = int(input("Введите период вложения: "))
interest_earned = my_bank.calculate_interest(time_p)
print(f"Начисленные проценты за {time_p} года: {interest_earned:.2f} руб.")

try:
    wd = float(input("Введите сумму для снятия: "))
    my_bank.withdraw(wd)
    print(f"Новый баланс: {my_bank.balance:.2f} руб.")
except ValueError as e:
    print(e)
```

**Протокол программы:**

Введите текущий баланс: 5000

Введите текущую ставку: 5.5

Текущий баланс: 5000.00 руб.

Введите период вложения: 3

Начисленные проценты за 3 года: 825.00 руб.

Введите сумму для снятия: 3000

Новый баланс: 2000.00 руб.

Process finished with exit code 0

## Блок 2

**Постановка задачи:** 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:**

```
# 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


my_dog = Dog("Немецкая", 5, "Овчарка")
my_cat = Cat("Вислоухая", 7, "Шотландская")


print("Моя собака вида: " + my_dog.species + ", возрастом:", my_dog.age,
"лет, " + "породы: " + my_dog.breed + ".")
print("Моя кошка вида: " + my_cat.species + ", возрастом:", my_cat.age, "лет,
" + "породы: " + my_cat.breed + ".")
```

**Протокол программы:**

Моя собака вида: Немецкая, возрастом: 5 лет, породы: Овчарка.

Моя кошка вида: Вислоухая, возрастом: 7 лет, породы: Шотландская.

Process finished with exit code 0

Блок 3

**Постановка задачи:** 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.

**Текст программы:**

```
# 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.
import pickle


class Bank:
    def __init__(self, balance, interest_rate, time_period, amount):
        self.balance = balance
        self.interest_rate = interest_rate
        self.time_period = time_period
        self.amount = amount

    def calculate_interest(self):
        interest = self.balance * (self.interest_rate / 100) *
self.time_period
        return interest

    def withdraw(self):
        if self.amount > self.balance:
            raise ValueError("Недостаточно средств на балансе.")
        self.balance -= self.amount
        return self.balance


def save_def(they_bank, filename):
    with open(filename, 'wb') as f:
        pickle.dump(they_bank, f)


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


bank1 = Bank(5000, 5.3, 4, 2000)
bank2 = Bank(6000, 5.5, 5, 3000)
bank3 = Bank(5000, 5.7, 6, 4000)
they_bank = [bank1, bank2, bank3]

save_def(they_bank, 'they_bank.pkl')
loaded_they_bank = load_def('they_bank.pkl')
for banks in loaded_they_bank:
    print(f"Начисленные проценты: {banks.calculate_interest():.2f} руб.")
    print("Новый баланс после снятия:", banks.withdraw())
    print()
```

**Протокол программы:**

Начисленные проценты: 1060.00 руб.

Новый баланс после снятия: 3000

Начисленные проценты: 1650.00 руб.

Новый баланс после снятия: 3000

Начисленные проценты: 1710.00 руб.

Новый баланс после снятия: 1000

Process finished with exit code 0

**Вывод:** в процессе выполнения практического занятия выработала навыки составления программ с ООП в IDE PyCharm Community. Выполнены разработка кода, откладка, тестирование программного кода. Готовые программные коды выложены на GitHub