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 calc\_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.calc\_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(f"Моя собака вида: {my\_dog.species}, возрастом: {my\_dog.age} лет, породы: {my\_dog.breed}.")  
print(f"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 calc\_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.calc\_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 c ООП в IDE PyCharm Community. Выполнены разработка кода, откладка, тестирование программного кода. Готовые программные коды выложены на GitHub