Студент группы ИС-23 Федосеев Тихон

Практическая работа №16

Тема: составление программ с использованием ООП в IDE PyCharm Community.

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

Тип алгоритма: Последовательный

Постановка задачи:

Задача 1.

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

```python
#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):
        """
        Вычисляет начисленные проценты за указанный период времени.

        Args:
            time_period (int): Период времени в годах.

        Returns:
            float: Начисленные проценты.
        """
        interest = self.balance * (self.interest_rate / 100) * time_period
        return interest

    def withdraw(self, amount):
        """
        Снимает указанную сумму денег с баланса.

        Args:
            amount (float): Сумма для снятия.

        Raises:
            ValueError: Если сумма для снятия превышает текущий баланс.
        """
        if amount > self.balance:
            raise ValueError("Недостаточно средств на балансе.")
        self.balance -= amount


# Пример использования
my_bank = Bank(10000, 5.0)
print(f"Текущий баланс: {my_bank.balance:.2f} руб.")

interest_earned = my_bank.calculate_interest(3)
print(f"Начисленные проценты за 3 года: {interest_earned:.2f} руб.")
```

```
try:
    my_bank.withdraw(3000)
    print(f"Новый баланс: {my_bank.balance:.2f} руб.")
except ValueError as e:
    print(e)
```

Протокол работы программы:

Текущий баланс: 10000.00 руб.
Начисленные проценты за 3 года: 1500.00 руб.
Новый баланс: 7000.00 руб.

Process finished with exit code 0

Задача 2.
Создайте базовый класс "Фигура" со свойствами "ширина" и "высота". От этого класса унаследуйте классы "Прямоугольник" и "Квадрат". Для класса "Квадрат" переопределите методы, связанные с вычислением площади и периметра.

```
#Создайте базовый класс "Фигура" со свойствами "ширина" и "высота". От этого
#класса унаследуйте классы "Прямоугольник" и "Квадрат". Для класса "Квадрат"
#переопределите методы, связанные с вычислением площади и периметра.

class Figure:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        pass

    def perimeter(self):
        pass


class Rectangle(Figure):
    def area(self):
        return self.width * self.height

    def perimeter(self):
        return 2 * (self.width + self.height)


class Square(Figure):
    def __init__(self, side):
        super().__init__(side, side)

    def area(self):
        return self.width ** 2

    def perimeter(self):
        return 4 * self.width


# Создаем объекты
rect = Rectangle(5, 3)
square = Square(4)

# Вычисляем площади и периметры
print(f"Площадь прямоугольника: {rect.area()}")
print(f"Периметр прямоугольника: {rect.perimeter()}")
print(f"Площадь квадрата: {square.area()}")
print(f"Периметр квадрата: {square.perimeter()}")
```

Протокол работы программы:

```
Площадь прямоугольника: 15
Периметр прямоугольника: 16
Площадь квадрата: 16
Периметр квадрата: 16

Process finished with exit code 0
```

Задача 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.

```python
#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, money, rate):
        self.money = money
        self.rate = rate

    def calculate_interest(self):
        return self.money * self.rate / 100

    def withdraw(self, amount):
        if amount > self.money:
            raise ValueError("Insufficient funds")
        self.money -= amount

    def deposit(self, amount):
        self.money += amount

def save_def(bank_instances, filename):
    with open(filename, 'wb') as file:
        pickle.dump(bank_instances, file)

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

# Example usage
bank1 = Bank(10000, 5)
bank2 = Bank(20000, 3.5)
bank3 = Bank(15000, 4.2)

bank_instances = [bank1, bank2, bank3]

save_def(bank_instances, 'bank_data.pkl')

loaded_instances = load_def('bank_data.pkl')
for instance in loaded_instances:
    print(f"Money: {instance.money}, Rate: {instance.rate}")
```

Протокол работы программы:

```
Money: 10000, Rate: 5
Money: 20000, Rate: 3.5
Money: 15000, Rate: 4.2

Process finished with exit code 0
```

Вывод: в процессе выполнения практического занятия закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с использованием ООП в IDE PyCharm Community. Были использованы языковые конструкции def, class и другие. Выполнены разработка кода, отладка, тестирование, оптимизация программного кода. Готовые программные коды выложены на GitHub.