Евлантьева Алина ИС-25

Отчет о практическом задании

Практическое задание №16. Вариант 9

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 использованием ООП в IDE PyCharm Community.

Задание 1

Постановка задачи

#Создайте класс "Калькулятор" с методами "сложение", "вычитание", "умножение" и "деление".
#Каждый метод должен принимать два аргумента и возвращать результат операции

Текст программы:

```
#Создайте класс "Калькулятор" с методами "сложение", "вычитание", "умножение" и "деление".
#Каждый метод должен принимать два аргумента и возвращать результат операции

class Calc:
    def Addition(self, a, b):
        return a + b
    def Subtraction(self, a, b):        return a - b

    def Multiplication(self, a, b):
        return a * b
    def Division(self, a, b):
        try:
            return a / b
        except ZeroDivisionError:
            print('Деление на 0 невозможно')
            exit()

Calculate = Calc()
```

```
a = 5
b = 5
result = Calculate.Addition(a, b)
print("Сложение", result)
result = Calculate.Subtraction(a, b)
print("Вычитание", result)
result = Calculate.Multiplication(a, b)
print("Умножение", result)
result = Calculate.Division(a, b)
print("Деление", result)
```

Протокол программы:

Сложение 10

Вычитание 0

Умножение 25

Деление 1.0

Process finished with exit code 0

Задание 2

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

Текст программы:

```
#Создание базового класса "Работник" и его наследование для создания классов
# "Менеджер" и "Инженер". В классе "Работник" будут общие методы, такие как
# "работать" и "получить зарплату", а классы-наследники будут иметь свои уникальные методы
# и свойства такие как "управлять командой" и "проектировть системы".
class Worker:
    def __init__(self, name, doljnost, salary):
        self.name = name
```

```
        self.doljnost = doljnost
        self.salary = salary

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

    def getawage(self):
        print(f"{self.name} получил зарплату в размере {self.salary} рублей")


class Manager(Worker):
    def __init__(self, name, salary):
        super().__init__(name, "Менеджер", salary)

    def manage_team(self):
        print(f"{self.name} управляет своей командой")


class Engineer(Worker):
    def __init__(self, name, salary):
        super().__init__(name, "Инженер", salary)

    def design_systems(self):
        print(f"{self.name} проектирует системы")


менеджер = Manager("Иван", 100000)
менеджер.work()
менеджер.getawage()
менеджер.manage_team()

инженер = Engineer("Петр", 80000)
инженер.work()
инженер.getawage()
инженер.design_systems()
```

Протокол программы:

Иван выполняет свою работу
Иван получил зарплату в размере 100000 рублей
Иван управляет своей командой
Петр выполняет свою работу
Петр получил зарплату в размере 80000 рублей
Петр проектирует системы

Process finished with exit code 0

Задание 3

Для задачи из блока 1 создать две функции, save_def и load_def,
которые позволяют сохранять информацию из экземпляров класса (3шт) в файл и загружать ее обратно.
Использовать модуль pickle для сериализации и десериализации обьектов 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 (3шт) в файл и загружать ее обратно.
# Использовать модуль pickle для сериализации и десериализации обьектов Python в бинарном формате
import pickle
class Calc:
    def Addition(self, a, b):
        return a + b
    def Subtraction(self, a, b):
        return a - b
    def Multiplication(self, a, b):
        return a * b
    def Division(self, a, b):
        try:
            return a / b
        except ZeroDivisionError:
            print('Деление на 0 невозможно')
            exit()

Calculate = Calc()
a = 5
b = 5

def save_def(obj, filename):
    with open(filename, 'wb') as f:
        pickle.dump(obj, f)
def load_def(filename):
    with open(filename, 'rb') as f:
        return pickle.load(f)
save_def(Calculate, 'calc_obj.pkl')

# Загрузка объектов из файла
Calc_new = load_def('calc_obj.pkl')
```

```
result = Calc_new.Addition(a, b)
print("Сложение", result)
result = Calc_new.Subtraction(a, b)
print("Вычитание", result)
result = Calc_new.Multiplication(a, b)
print("Умножение", result)
result = Calc_new.Division(a, b)
print("Деление", result)
```

Протокол программы:

Сложение 10
Вычитание 0
Умножение 25
Деление 1.0

Process finished with exit code 0

Вывод: в процессе выполнения практического задания №16 я выработала навыки составления программ с использованием ООП в IDE PyCharm Community. Выполнены: разработка кода, откладка, тестирование, оптимизация программного кода. Готовые программные коды выложены на GitHub.