Практическое занятие №16

Студент группы ИС-23 Яцына Даниил

Практическое занятие №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.

Задача №1:

Создайте класс «Матрица», который имеет атрибуты количества строк и столбцов. Добавьте методы для сложения, вычитания и умножения матриц.

Текст программы:

```python
class Matrix:
    columns = 0
    rows = 0


    def __init__(self, col, row, matrix):
        self.columns = col
        self.rows = row
        self.matrix = matrix

    def addiction(self, other):
        if self.rows !=other.rows or self.columns != other.columns:
            raise ValueError(f"Нельзя сложить матрицу {self.matrix} с {other.matrix}")

        result_matrix = []

        for i in range(self.rows):
            new_row = [self.matrix[i][j] + other.matrix[i][j] for j in range(self.columns)]
            result_matrix.append(new_row)
        return Matrix(self.columns, self.rows, result_matrix)

    def subtraction(self, other):
        if self.rows !=other.rows or self.columns != other.columns:
            raise ValueError(f"Нельзя вычесть матрицу {self.matrix} с {other.matrix}")

        result_matrix = []

        for i in range(self.rows):
            new_row = [self.matrix[i][j] - other.matrix[i][j] for j in range(self.columns)]
            result_matrix.append(new_row)
        return Matrix(self.columns, self.rows, result_matrix)

    def multypli(self,other):
        if(self.columns!=other.rows):
            raise ValueError(f"Нельзя перемножить матрицу {self.matrix} с {other.matrix}")
        result_matrix = []

        for i in range(self.rows):
            new_row = []
            for j in range(other.columns):
                new_row.append(sum(self.matrix[i][k] + other.matrix[k][j] for j in range(self.columns) for k in range(self.columns)))
            result_matrix.append(new_row)
        return Matrix(self.columns, self.rows, result_matrix)
```

```
    def show(self):
        matrix_str = " "
        for  row in self.matrix:
            matrix_str += " ".join(str(elem) for elem in row) + "\t"
        print(f"\nMatrix: {matrix_str}")


m1 = Matrix(2,2,[[1,2],[2,2]])
m2 = Matrix(2,2,[[2,2],[1,2]])

m3 = m1.addiction(m2)
m4 = m1.subtraction(m2)
m5 = m1.multypli(m2)

m3.show()
m4.show()
m5.show()
```

Протокол работы программы:
Matrix: 3 4   3 4

Matrix: -1 0   1 0

Matrix: 13 13 15 15

Программа успешно завершена.

Задача №2

Создайте базовый класс "Человек" со свойствами "имя", "возраст" и "пол". От этого класса унаследуйте классы "Мужчина" и "Женщина" и добавьте в них свойства, связанные с социальным положением (например, "семейное положение", "количество детей" и т.д.)

Текст программы:

```
class Human():
    def __init__(self, name, age, sex):
        self.name = name
        self.age = age
        self.sex = sex

class Man(Human):
    def __init__(self, name, age,  status, count_child):
        super().__init__(name,age, "Муж")
        self.status = status
        self.count_child = count_child

    def show(self):
        print(f"\n===\nИмя:{self.name}\nВозраст:{self.age}\nПол:{self.sex}\nСеменойе положение:{self.status}\nКоличество детей: {self.count_child}\n===")


class Woman(Human):
    def __init__(self, name, age, status, count_child):
        super().__init__(name,age,"Женщ")
        self.status = status
        self.count_child = count_child
    def show(self):
        print(f"\n===\nИмя:{self.name}\nВозраст:{self.age}\nПол:{self.sex}\nСеменойе положение:{self.status}\nКоличество детей: {self.count_child}\n===")

man1 = Man("Джон", 37, "Женат", 0)
man1.show()

wom1 = Woman("Соня", 27, "Замужем", "2")
wom1.show()
```

Протокол работы программы:

```
===
Имя : Джон
Возраст : 37
```

```
Пол:Муж
Семенойе положение:Женат
Количество детей: 0
===

===
Имя:Соня
Возраст:27
Пол:Женщ
Семенойе положение:Замужем
Количество детей: 2
===
```

Задача 3:

```python
def save_def(obj, file_path):
    with open(file_path, 'wb') as file:
        pickle.dump(obj, file)

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

save_def(m1, 'matrix1.pkl')
save_def(m2, 'matrix2.pkl')
save_def(m3, 'result_matrix.pkl')


loaded_m1 = load_def('matrix1.pkl')
loaded_m2 = load_def('matrix2.pkl')
loaded_result = load_def('result_matrix.pkl')
```

Вывод: В процессе выполнения практического занятия выработал навыки составление программ с использованием ООП в IDE PyCharm Community. и закрепил усвоенные навыки.
Выполнены разработка кода, отладка, тестирование, оптимизация программного кода. Готовые программные коды выложена на GitHub.