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 биспользованием ООП в IDE PyCharm Community.

Задание 1.

Создайте класс "Матрица", который имеет атрибуты количество строк и столбцов. Добавьте методы для сложения, вычитания и умножения матриц.

Текст программы.

```
import random

class Matrix:
    def __init__(self, rows, cols, data):
        self.rows = rows
        self.cols = cols
        self.data = data

    def add(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Matrix dimensions do not match for addition")
        result = [[0 for _ in range(self.cols)] for _ in range(self.rows)]
        for i in range(self.rows):
            for j in range(self.cols):
                result[i][j] = self.data[i][j] + other.data[i][j]
        return result

    def sub(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Матрицы несовместимы для вычитания.")
        result = [[0 for _ in range(self.cols)] for _ in range(self.rows)]
        for i in range(self.rows):
            for j in range(self.cols):
                result[i][j] = self.data[i][j] - other.data[i][j]
        return result

    def mul(self, other):
        if self.cols != other.rows:
            raise ValueError("Матрицы несовместимы для умножения.")
        result = [[0 for _ in range(self.cols)] for _ in range(other.rows)]
        for i in range(self.cols):
            for j in range(other.rows):
                for k in range(self.rows):
                    result[i][j] += self.data[i][k] * other.data[k][j]
        return result

matrix1 = [[random.randint(-5, 5) for i in range(2)] for i in range(2)]
matrix2 = [[random.randint(-5, 5) for i in range(2)] for i in range(2)]

print(f"Матрица 1: {matrix1}")
print(f"Матрица 2: {matrix2}")

matrix1 = Matrix(len(matrix1[0]), len(matrix1), matrix1)
matrix2 = Matrix(len(matrix2[0]), len(matrix2), matrix2)
res_add = matrix1.add(matrix2)
print(f"Сумма матриц: {res_add}")
res_sub = matrix1.sub(matrix2)
print(f"Разность матриц: {res_sub}")
res_mul = matrix1.mul(matrix2)
print(f"Произведение матриц: {res_mul}")
```

Протокол работы программы.

Матрица 1: [[1, 4], [-5, 2]]
Матрица 2: [[-3, 0], [0, -1]]
Сумма матриц: [[-2, 4], [-5, 1]]
Разность матриц: [[4, 4], [-5, 3]]
Произведение матриц: [[-3, -4], [15, -2]]

Задание 2.

Постановка задачи.

Создание базового класса "Фигура" и его наследование для создания классов "Квадрат", "Прямоугольник", "Круг". Класс "Фигура" будет иметь общие методы, такие как вычисление площади и периметра, а классы-наследники будут иметь спецефичные свойства и методы.

Тип алгоритма.

Линейный.

Текст программы.

```
import math
class Figure:
    def __init__(self):
        self.area = 0
        self.perimeter = 0

    def calculate_area(self):
        pass
    def calculate_perimeter(self):
        pass

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

    def calculate_area(self):
        self.area = self.side ** 2
    def calculate_perimeter(self):
        self.perimeter = 4 * self.side

class Rectangle(Figure):
    def __init__(self, length, width):
        super().__init__()
        self.length = length
        self.width = width

    def calculate_area(self):
        self.area = self.length * self.width
    def calculate_perimeter(self):
        self.perimeter = 2 * (self.length + self.width)


class Circle(Figure):
    def __init__(self, radius):
        super().__init__()
        self.radius = radius

    def calculate_area(self):
        self.area = math.pi * self.radius ** 2
    def calculate_perimeter(self):
        self.perimeter = 2 * math.pi * self.radius
```

```
square = Square(5)
square.calculate_area()
square.calculate_perimeter()
print("Площадь квадрата:", square.area)
print("Периметр квадрата:", square.perimeter)

rectangle = Rectangle(10, 5)
rectangle.calculate_area()
rectangle.calculate_perimeter()
print("Площадь прямоугольника:", rectangle.area)
print("Периметр прямоугольника:", rectangle.perimeter)

circle = Circle(3)
circle.calculate_area()
circle.calculate_perimeter()
print("Площадь круга:", circle.area)
print("Периметр круга:", circle.perimeter)
```

Протокол работы программы.

Площадь квадрата: 25
Периметр квадрата: 20
Площадь прямоугольника: 50
Периметр прямоугольника: 30
Площадь круга: 28.274333882308138
Периметр круга: 18.84955592153876

Process finished with exit code 0

Задание 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.

Тип алгоритма.

Линейный.

```
import pickle
import random

class Matrix:
    def __init__(self, rows, cols, data):
        self.rows = rows
        self.cols = cols
        self.data = data

    def add(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Matrix dimensions do not match for addition")
        result = [[0 for _ in range(self.cols)] for _ in range(self.rows)]
        for i in range(self.rows):
            for j in range(self.cols):
                result[i][j] = self.data[i][j] + other.data[i][j]
        return result

    def sub(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Матрицы несовместимы для вычитания.")
        result = [[0 for _ in range(self.cols)] for _ in range(self.rows)]
        for i in range(self.rows):
            for j in range(self.cols):
                result[i][j] = self.data[i][j] - other.data[i][j]
        return result
```

```
    def mul(self, other):
        if self.cols != other.rows:
            raise ValueError("Матрицы несовместимы для умножения.")
        result = [[0 for _ in range(self.cols)] for _ in range(other.rows)]
        for i in range(self.cols):
            for j in range(other.rows):
                for k in range(self.rows):
                    result[i][j] += self.data[i][k] * other.data[k][j]
        return result

def save_def(matri, file):
    with open(file, 'wb') as f:
        pickle.dump(matri, f)


def load_def(file):
    with open(file, 'rb') as f:
        matr = pickle.load(f)

    return matr

matrix1 = [[random.randint(-5, 5) for i in range(2)] for i in range(2)]
matrix2 = [[random.randint(-5, 5) for i in range(2)] for i in range(2)]
matrix3 = [[random.randint(-5, 5) for i in range(2)] for i in range(2)]

print(f"Матрица 1: {matrix1}")
print(f"Матрица 2: {matrix2}")
print(f"Матрица 3: {matrix3}")

matrix1 = Matrix(len(matrix1[0]), len(matrix1), matrix1)
matrix2 = Matrix(len(matrix2[0]), len(matrix2), matrix2)
matrix3 = Matrix(len(matrix3[0]), len(matrix3), matrix3)

matrix_info = [matrix1, matrix2, matrix3]
for mat in matrix_info:
    save_def(mat, 'matrixes.pkl')
    matrixes = load_def('matrixes.pkl')
    print(f"Сумма матриц: {matrixes.add(matrixes)}")
    print(f"Разность матриц: {matrixes.sub(matrixes)}")
    print(f"Произведение матриц: {matrixes.mul(matrixes)}")
```

Протокол работы программы.

Матрица 1: [[-2, 1], [2, 5]]
Матрица 2: [[2, -1], [-3, -2]]
Матрица 3: [[-4, -4], [-1, -3]]
Сумма матриц: [[-4, 2], [4, 10]]
Разность матриц: [[0, 0], [0, 0]]
Произведение матриц: [[6, 3], [6, 27]]
Сумма матриц: [[4, -2], [-6, -4]]
Разность матриц: [[0, 0], [0, 0]]
Произведение матриц: [[7, 0], [0, 7]]
Сумма матриц: [[-8, -8], [-2, -6]]
Разность матриц: [[0, 0], [0, 0]]
Произведение матриц: [[20, 28], [7, 13]]

Process finished with exit code 0
Вывод: Закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрел навыки составления программ с использованием ООП в IDE PyCharm Community.