МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №11

по дисциплине: ООП
тема: «Знакомство с языком программирования Python. Базовые структуры данных.»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
пр. Черников Сергей Викторович

Белгород 2024 г.

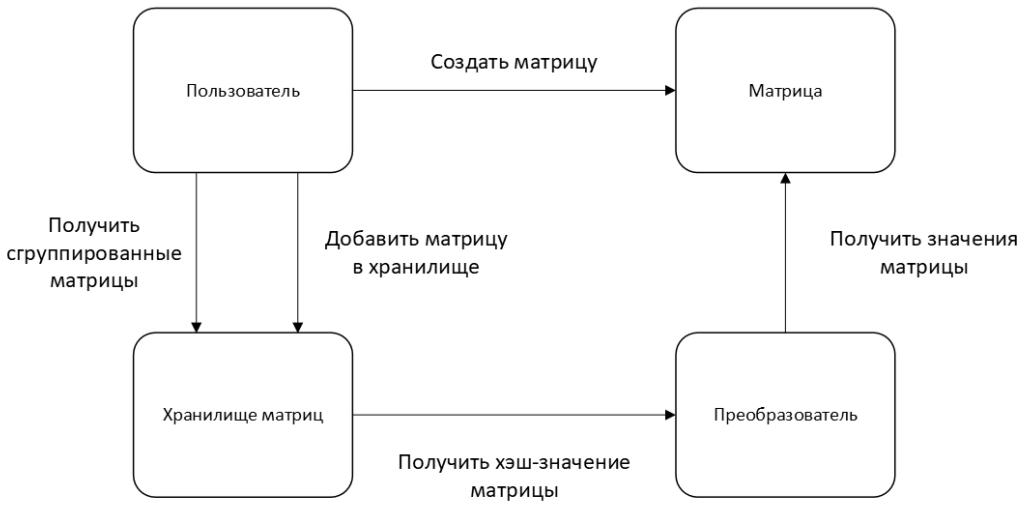
Лабораторная работа №11

«Знакомство с языком программирования Python. Базовые структуры данных.»

Вариант 3

Цель работы: познакомится с базовыми конструкциями языка. Получить навык создания простых приложений. Изучить базовые типы.

В соответствии с вариантом задания требуется выполнить объектную декомпозицию задачи. В качестве одного из обязательных объектов выделить «матрицу». Для реализации соблюдения условия задачи требуется использовать возможности перегрузки операторов. При выводе также требуется выполнить перегрузку соответствующего оператора. На вход подаются данные в форме двумерных «матриц», количество матриц заранее не определено, разделителем между матрицами являются строки. Для каждой матрицы найти все, которые удовлетворяют следующему условию: четность/нечетность соответствующих элементов матриц совпадает. Форма матрицы может быть не полной. Формат вывода требуется соблюсти.




```python
from __future__ import print_function
from abc import ABC, abstractmethod


class Matrix:
    def __init__(self):
        self.matrix = []

    def add_line(self, line: list) -> None:
        self.matrix.append(line)

    def get_height(self):
        return len(self.matrix)

    def get_min_width(self):
        return min(map(lambda x: len(x), self.matrix))

    def get_max_width(self):
```

```
        return max(map(lambda x: len(x), self.matrix))


class GroupMatricesPredicate(ABC):
    @abstractmethod
    def hash(self, matrix):
        pass


class GroupByEvenOddFactor(GroupMatricesPredicate):
    def hash(self, matrix):
        result = []

        for line in matrix.matrix:
            result.append(tuple(map(lambda x: x % 2 == 0, line)))

        return tuple(result)


class GroupMatrices:
    def __init__(self):
        self.matrices = []

    def add_matrix(self, matrix: Matrix):
        self.matrices.append(matrix)

    def group_by_predicate(self, predicate: GroupMatricesPredicate):
        result: dict = {}

        for matrix in self.matrices:
            hash = predicate.hash(matrix)

            if hash in result.keys():
                result[hash].append(matrix)
            else:
                result[hash] = [matrix]

        return result


def main():
    group_matrices = GroupMatrices()

    with open(input("Введите путь к файлу: "), "r") as f:
        tmp_matrix = Matrix()

        for line in f:
            if not line.strip():
                group_matrices.add_matrix(tmp_matrix)
                tmp_matrix = Matrix()
            else:
                tmp_matrix.add_line(list(map(lambda x: int(x), line.strip().split(" "))))

        if tmp_matrix.get_height() != 0:
```

```
            group_matrices.add_matrix(tmp_matrix)

    for group, matrices in group_matrices.group_by_predicate(GroupByEvenOddFactor()).items():
        print("Группа:")
        print(*group, "", sep="\n")

        print("Матрицы:")
        for matrix in matrices:
            print(*matrix.matrix, "", sep="\n")

        print("")


if __name__ == "__main__":
    main()
```

Ссылка на репозиторий

Вывод: в ходе лабораторной работы познакомились с базовыми конструкциями языка. Получили навык создания простых приложений. Изучили базовые типы.