МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по рубежному контролю №1
Вариант Г21

Выполнил:
Студент группы ИУ5-35Б
Перфильев С. М.
Преподаватель:
Гапанюк Ю. Е.

Подпись, дата

_____/_____

Москва 2025

Задание

В рамках рубежного контроля №1 по курсу «Парадигмы и конструкции языков программирования» требовалось разработать программу на языке Python.

Цель работы

1. Создать два класса данных, соответствующие предметной области «**Оператор**» (**Класс 1**) и «**Язык программирования**» (**Класс 2**) (Вариант №21).
2. Реализовать связь «один-ко-многим» между классами.
3. Создать дополнительный класс для реализации связи «многие-ко-многим».
4. Создать списки объектов классов, содержащие тестовые данные (3-5 записей).
5. Реализовать три запроса в соответствии с **Вариантом Г**, адаптировав их под выбранную предметную область.
6. При реализации запросов использовать функциональные возможности Python (list/dict comprehensions, функции высших порядков).

Запросы

Запрос 1
«Оператор» и «Язык программирования» связаны соотношением «один-ко-многим».
Вывести список всех **языков программирования**, название которых начинается с буквы «А», и список **операторов**, относящихся к каждому из этих языков.

Запрос 2
«Оператор» и «Язык программирования» связаны соотношением «один-ко-многим».
Вывести список **языков программирования** с максимальным значением количественного признака их операторов (например, частоты использования) для каждого языка, отсортированный по максимальному значению этого признака.

Запрос 3
«Оператор» и «Язык программирования» связаны соотношением «многие-ко-многим».
Вывести список всех связанных **операторов** и **языков программирования**, отсортированный по языкам программирования.

Листинг кода

```
from operator import itemgetter


class Operator:

    def __init__(self, id, name, usage, lang_id):
        self.id = id
        self.name = name
        self.usage = usage
        self.lang_id = lang_id


class Lang:
    def __init__(self, id, name):
        self.id = id
        self.name = name


class OperLang:

    def __init__(self, lang_id, oper_id):
        self.lang_id = lang_id
        self.oper_id = oper_id



langs = [
    Lang(1, 'Ассемблер'),
    Lang(2, 'Алгол'),
    Lang(3, 'Питон'),
    Lang(4, 'Java'),
    Lang(5, 'Ада'),
]

opers = [
    Operator(1, 'MOV',   100, 1),
    Operator(2, 'ADD',    80, 1),
```

```
    Operator(3, 'IF',    120, 3),
    Operator(4, 'FOR',   110, 3),
    Operator(5, 'BEGIN',  90, 2),
    Operator(6, 'END',    70, 2),
    Operator(7, 'LOOP',   60, 5),
]


opers_langs = [
    OperLang(1, 1),
    OperLang(1, 2),
    OperLang(2, 5),
    OperLang(2, 6),
    OperLang(3, 3),
    OperLang(3, 4),
    OperLang(3, 6),
    OperLang(5, 7),
]


def main():

    one_to_many = [
        (op.name, op.usage, lang.name)
        for lang in langs
        for op in opers
        if op.lang_id == lang.id
    ]

    many_to_many_temp = [
        (lang.name, ol.lang_id, ol.oper_id)
        for lang in langs
        for ol in opers_langs
        if lang.id == ol.lang_id
    ]

    many_to_many = [
        (op.name, op.usage, lang_name)
```

```
        for lang_name, lang_id, oper_id in many_to_many_temp
        for op in opers if op.id == oper_id
    ]

    print('Задание Г1')
    res_g1 = {}
    for lang in langs:
        if lang.name.startswith('A'):
            lang_ops = list(filter(lambda x: x[2] == lang.name, one_to_many))
            op_names = [name for name, _, _ in lang_ops]
            res_g1[lang.name] = op_names

    print(res_g1)

    print('\nЗадание Г2')
    res_g2_unsorted = []
    for lang in langs:
        lang_ops = list(filter(lambda x: x[2] == lang.name, one_to_many))
        if lang_ops:
            usages = [usage for _, usage, _ in lang_ops]
            max_usage = max(usages)
            res_g2_unsorted.append((lang.name, max_usage))

    res_g2 = sorted(res_g2_unsorted, key=itemgetter(1), reverse=True)
    print(res_g2)

    print('\nЗадание Г3')
    res_g3 = sorted(many_to_many, key=itemgetter(2))
    print(res_g3)


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

Вывод программы

```
Задание Г1
{'Ассемблер': ['MOV', 'ADD'], 'Алгол': ['BEGIN', 'END'], 'Ада': ['LOOP']}

Задание Г2
[('Питон', 120), ('Ассемблер', 100), ('Алгол', 90), ('Ада', 60)]

Задание Г3
[('LOOP', 60, 'Ада'), ('BEGIN', 90, 'Алгол'), ('END', 70, 'Алгол'), ('MOV', 100, 'Ассемблер'), ('ADD', 80, 'Ассемблер'), ('IF', 120, 'Питон'), ('FOR', 110, 'Питон'), (
'END', 70, 'Питон')]
```