Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»
Отчет по рубежному контролю №1
«**Вариант А, 1**»

Выполнил:
Студент группы ИУ5-31Б
Баженов Никита

Проверил:
Гапанюк Ю. Е.

2025 г.

Листинг программы

python

main.py

```python
from typing import List, Any, Callable

from functools import reduce
from dataclasses import dataclass
from random import choice, randint, seed


@dataclass
class Student:
    id_: int
    surname: str
    mark: int     # Числовое поле Оценка. Вместо суммы возьмем среднее
    group_id: int


@dataclass
class Group:
    id_: int
    name: str


@dataclass
class GroupStudents:
    """ Класс для реализации отношения многие ко многим. """
    student_id: int
    group_id: int



def get_groups(count: int) -> List[Group]:
    """ Функция для генерации данных групп. """
    return [
        Group(
            i,
            f"ИУ5-3{i + 1}"
        )
        for i in range(count)
    ]


def get_students(groups: List[Group], in_group: int=5) -> List[Student]:
    """ Функция для генерации данных студентов. """

    def get_random_surname() -> str:
        return choice(["Иванов", "Петров", "Сидоров", "Всеволод", "Тарасов"])

    def get_random_mark() -> int:
        return randint(0, 100)


    return [
        Student(
            i,
            get_random_surname(),
            get_random_mark(),
            i // in_group
        )
        for i in range(len(groups) * in_group)
    ]
```

```python


def get_group_students(groups: List[Group], students: List[Student], count: int) -> List[GroupStudents]:
    """ Функция для генерации связей многие ко многим. """

    def get_random_student_id() -> int:
        return choice(students).id_

    def get_random_group_id() -> int:
        return choice(groups).id_


    return [
        GroupStudents(
            get_random_student_id(),
            get_random_group_id()
        )
        for _ in range(count)
    ]


def print_data(data: List[Any], headers: List[str], title: str, column_width: int=15) -> None:
    """
        Функция для вывода данных в виде таблицы.
        Принимает данные, заголовки столбцов и заголовок таблицы
        """
    total_length = len(headers) * column_width
    columns = len(headers)

    print(f"{title:=^{total_length}}")
    print(("{:<{column_width}}" * columns).format(*headers, column_width=column_width))
    print()
    print("\n".join(
        [
            ("{:<{column_width}}" * columns).format(*i, column_width=column_width) for i in data
        ])
    )
    print()


def first_query(groups: List[Group], students: List[Student]) -> List[Any]:
    """ Реализация первого запроса. """
    result = list()
    for group in groups:
        for student in students:
            if group.id_ == student.group_id:
                result.append((group.name, student.surname, student.mark))

    return result


def second_query(groups: List[Group], students: List[Student]) -> List[Any]:
    """ Реализация второго запроса. """
    counter: dict[int, list] = dict()
    for group in groups:
        counter[group.id_] = [0, 0]

    for student in students:
        counter[student.group_id][0] += student.mark
        counter[student.group_id][1] += 1

    result = list()
```

```python
    for group in groups:
        if counter[group.id_][1] == 0:
            result.append((group.name, 0))
        else:
            result.append((group.name, counter[group.id_][0] / counter[group.id_][1]))

    result.sort(key=lambda x: x[1], reverse=True)

    return result


def third_query(groups: List[Group], students: List[Student], relations: List[GroupStudents], condition: Callable) -> List[Any]:
    """ Реализация третьего запроса. """
    result: dict[int, list] = dict()
    for group in groups:
        if condition(group.name) and (group.id_ not in result):
            result[group.id_] = []

    for relation in relations:
        if relation.group_id in result:
            result[relation.group_id].append(relation.student_id)

    filtered_data = list()
    for group in groups:
        if group.id_ in result:
            for student in students:
                if student.id_ in result[group.id_]:
                    filtered_data.append((group.name, student.surname, student.id_))

    return filtered_data


def main() -> None:
    seed(42)

    groups = get_groups(3)
    students = get_students(groups)

    # Первый запрос
    print_data(
        first_query(groups, students),
        ["Группа", "Фамилия", "Оценка"],
        "Запрос 1",
    )

    # Второй запрос
    print_data(
        second_query(groups, students),
        ["Группа", "Средний балл"],
        "Запрос 2"
    )

    # Третий запрос
    # Отношение многие ко многим никак не зависит от отношения один ко многим
    relations = get_group_students(groups, students, 10)

    print_data(
        third_query(groups, students, relations, lambda name: ("1" in name) or ("2" in name)),
        ["Группа", "Фамилия", "ID"],
        "Запрос 3"
    )

```

```
184
185 if __name__ == "__main__":
186     main()
```

Результат выполнения

```
Запрос A1
==================Запрос 1==================
Группа          Фамилия         Оценка

ИУ5-31          Иванов          3
ИУ5-31          Сидоров         31
ИУ5-31          Петров          17
ИУ5-31          Иванов          86
ИУ5-31          Тарасов         11
ИУ5-32          Тарасов         54
ИУ5-32          Иванов          3
ИУ5-32          Иванов          27
ИУ5-32          Петров          64
ИУ5-32          Тарасов         3
ИУ5-33          Тарасов         25
ИУ5-33          Тарасов         53
ИУ5-33          Петров          57
ИУ5-33          Тарасов         35
ИУ5-33          Иванов          97
```

```
Запрос A2
===========Запрос 2===========
Группа          Средний балл

ИУ5-33          53.4
ИУ5-32          30.2
ИУ5-31          29.6
```

```
Запрос A3
==================Запрос 3==================
Группа          Фамилия         ID

ИУ5-31          Сидоров         1
ИУ5-31          Петров          2
ИУ5-31          Иванов          6
ИУ5-31          Петров          12
ИУ5-32          Тарасов         5
ИУ5-32          Тарасов         9
ИУ5-32          Тарасов         11
ИУ5-32          Петров          12
```