Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «ГУИМЦ»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс«Парадигмы и конструкции языков программирования»»

ОТЧЕТ

ПО РУБЕЖНОМУ КОНТРОЛЮ №1

Вариант предметной области28

Вариант запросов: В

Студент: Агапова А.Д., группа ИУ5Ц-52Б

Преподаватель: Гапанюк Ю. Е.

2024г.

Вариант запросов В. Предметная область 28
1 «Кафедра» и «Студенческая группа» связаны соотношением один-ко-многим.
Выведите список всех групп, относящихся к факультету ИУ (название группы начинается с «ИУ»), и названия их кафедр.
2 «Кафедра» и «Студенческая группа» связаны соотношением один-ко-многим.
Выведите список кафедр с максимальным количеством людей в группе на каждой кафедре, отсортированный по минимальному количеству людей.
3 «Кафедра» и «Студенческая группа» связаны соотношением многие-ко-многим. Выведите список всех связанных групп и кафедр, отсортированный по группам, сортировка по кафедрам произвольная.

Рубежный контроль №2:
Рубежный контроль представляет собой разработку тестов на языке Python.
1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Листинг программы
main.py

```
class Group:
    def __init__(self, id: int, department: str, sem: int, index: int,
faculty_id: int, students: int):
        self._id = id
        self._name = f"{department}-{sem}{index}"
        self._faculty_id = faculty_id
        self._students = students

    @property
    def id(self) -> int:
        return self._id

    @property
    def faculty_id(self) -> int:
        return self._faculty_id

    @property
    def name(self) -> str:
        return self._name

    @property
    def students(self) -> int:
        return self._students


class Faculty:
    def __init__(self, id: int, title: str):
        self._id = id
        self._title = title

    @property
    def id(self) -> int:
```

```
        return self._id

    @property
    def title(self) -> str:
        return self._title


class GroupFaculty:
    def __init__(self, group_id: int, faculty_id: int):
        self._group_id = group_id
        self._faculty_id = faculty_id

    @property
    def faculty_id(self) -> int:
        return self._faculty_id

    @property
    def group_id(self) -> int:
        return self._group_id


def task1(faculties: list[Faculty], groups: list[Group]) -> list[tuple[Group,
Faculty]]:
    return [(g, c)
            for g in groups
            for c in faculties
            if (g.faculty_id == c.id and g.name.startswith("ИУ"))]


def task2(faculties: list[Faculty], groups: list[Group]) -> dict[str, int]:
    data = {}
    for faculty in faculties:
        faculty_students = [g.students
                            for g in groups
                            if (g.faculty_id == faculty.id)]
        if faculty_students:
            data[faculty.title] = max(faculty_students)
    return data


def task3(faculties: list[Faculty], groups: list[Group], groups_faculties:
list[GroupFaculty]) -> list[tuple[Group, Faculty]]:
    data = [(g, f)
            for gf in groups_faculties
            for g in groups
            for f in faculties
            if gf.group_id == g.id and gf.faculty_id == f.id]

    data.sort(key=lambda x: x[0].name)
    return data


def main():
    faculties = [
        Faculty(1, "Информатика и системы управления"),
        Faculty(2, "Инженерный бизнес и менеджмент"),
        Faculty(3, "Робототехника и комплексная автоматизация"),
        Faculty(4, "Фундаментальные науки"),
        Faculty(5, "Энергомашиностроение")
    ]

    groups = [
        Group(1, "ИУ5ц", 5, 1, 1, 28),
        Group(2, "ИУ6ц", 5, 2, 1, 24),
        Group(3, "ИУ7", 3, 3, 1, 29),
        Group(4, "ИБМ7", 3, 4, 2, 27),
```

```
        Group(5, "РК6", 3, 5, 3, 28),
        Group(6, "РК9", 3, 1, 3, 27),
        Group(7, "ФН12", 3, 2, 4, 21),
        Group(8, "ФН12", 3, 3, 4, 26),
        Group(9, "Э9ц", 5, 4, 5, 24)
    ]

    groups_faculties = [
        GroupFaculty(1, 1),
        GroupFaculty(2, 1),
        GroupFaculty(3, 1),
        GroupFaculty(4, 2),
        GroupFaculty(5, 3),
        GroupFaculty(6, 3),
        GroupFaculty(7, 4),
        GroupFaculty(8, 4),
        GroupFaculty(9, 5)
    ]

    # Вывод результатов
    print("Запрос № 1")
    for (g, c) in task1(faculties, groups):
        print(g.name, c.title)
    print()

    print("Запрос № 2")
    data_items = list(task2(faculties, groups).items())
    data_items.sort(key=lambda x: x[1])
    for (faculty, max_students) in data_items:
        print(faculty, max_students)
    print()

    print("Запрос № 3")
    for (group, course) in task3(faculties, groups, groups_faculties):
        print(group.name, course.title)
    print()


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

tests.py

```
import unittest
from main import Group, Faculty, GroupFaculty, task1, task2, task3


class TestTasks(unittest.TestCase):

    def setUp(self):
        self.faculties = [
            Faculty(1, "Информатика и системы управления"),
            Faculty(2, "Инженерный бизнес и менеджмент"),
            Faculty(3, "Робототехника и комплексная автоматизация"),
            Faculty(4, "Фундаментальные науки"),
            Faculty(5, "Энергомашиностроение")
        ]

        self.groups = [
            Group(1, "ИУ5ц", 5, 1, 1, 28),
            Group(2, "ИУ6ц", 5, 2, 1, 24),
            Group(3, "ИУ7", 3, 3, 1, 29),
            Group(4, "ИБМ7", 3, 4, 2, 27),
            Group(5, "РК6", 3, 5, 3, 28),
            Group(6, "РК9", 3, 1, 3, 27),
            Group(7, "ФН12", 3, 2, 4, 21),
            Group(8, "ФН12", 3, 3, 4, 26),
```

```
            Group(9, "Э9ц", 5, 4, 5, 24)
        ]

        self.groups_faculties = [
            GroupFaculty(1, 1),
            GroupFaculty(2, 1),
            GroupFaculty(3, 1),
            GroupFaculty(4, 2),
            GroupFaculty(5, 3),
            GroupFaculty(6, 3),
            GroupFaculty(7, 4),
            GroupFaculty(8, 4),
            GroupFaculty(9, 5)
        ]

    def test_task1(self):
        result = task1(self.faculties, self.groups)
        self.assertEqual(len(result), 3)
        self.assertEqual(result[0][0].name, "ИУ5ц-51")
        self.assertEqual(result[0][1].title, "Информатика и системы
управления")

    def test_task2(self):
        result = task2(self.faculties, self.groups)
        self.assertEqual(result["Информатика и системы управления"], 29)
        self.assertEqual(result["Инженерный бизнес и менеджмент"], 27)
        self.assertEqual(result["Робототехника и комплексная автоматизация"],
28)
        self.assertEqual(result["Фундаментальные науки"], 26)
        self.assertEqual(result["Энергомашиностроение"], 24)

    def test_task3(self):
        result = task3(self.faculties, self.groups, self.groups_faculties)
        self.assertEqual(len(result), 9)
        self.assertEqual(result[0][0].name, "ИБМ7-34")
        self.assertEqual(result[0][1].title, "Инженерный бизнес и
менеджмент")
        self.assertEqual(result[8][0].name, "Э9ц-54")
        self.assertEqual(result[8][1].title, "Энергомашиностроение")


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

Результат работы программы

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.19045.5131]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\AnnaA>cd Desktop

C:\Users\AnnaA\Desktop>cd Парадигмы программирования

C:\Users\AnnaA\Desktop\Парадигмы программирования>cd RK_2

C:\Users\AnnaA\Desktop\Парадигмы программирования\RK_2>python -m unittest tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK

C:\Users\AnnaA\Desktop\Парадигмы программирования\RK_2>
```