**Московский государственный технический университет им. Н.Э. Баумана**

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

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2

Вариант А 29

Выполнил:
студент группы ИУ5 - 32Б:
Васильев Н. Д.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2025 г.

## Задание:

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).

## Код программы:

### Файл main.py

```
from operator import itemgetter


class Department:
    """Кафедра"""

    def __init__(self, id, name, faculty_id):
        self.id = id
        self.name = name
        self.faculty_id = faculty_id


class Faculty:
    """Факультет"""

    def __init__(self, id, name, budget):
        self.id = id
        self.name = name
        self.budget = budget


class FacultyDepartment:
    """Связь многие-ко-многим"""

    def __init__(self, faculty_id, department_id):
        self.faculty_id = faculty_id
        self.department_id = department_id


# --- Данные (глобальные, чтобы их видели и функции, и тесты при импорте) ---
faculties = [
    Faculty(1, 'Факультет компьютерных наук', 5000000),
    Faculty(2, 'Факультет экономики', 4500000),
    Faculty(3, 'Факультет лингвистики', 4000000),
    Faculty(4, 'Факультет математики', 4800000),
```

```
]

departments = [
    Department(1, 'Кафедра математического анализа', 4),
    Department(2, 'Кафедра вычислительной техники', 1),
    Department(3, 'Кафедра иностранных языков', 3),
    Department(4, 'Кафедра физики', 4),
    Department(5, 'Кафедра информационных систем', 1),
    Department(6, 'Кафедра программной инженерии', 1),
]

faculties_departments = [
    FacultyDepartment(1, 2),
    FacultyDepartment(1, 5),
    FacultyDepartment(1, 6),
    FacultyDepartment(2, 1),
    FacultyDepartment(2, 3),
    FacultyDepartment(3, 3),
    FacultyDepartment(4, 1),
    FacultyDepartment(4, 4),
    FacultyDepartment(4, 5),
]


# --- Функции с логикой (для тестирования) ---

def get_one_to_many(facs, deps):
    """Соединение данных один-ко-многим"""
    return [
        (d.name, d.faculty_id, f.name, f.budget)
        for f in facs
        for d in deps
        if d.faculty_id == f.id
    ]


def get_many_to_many(facs, deps, links):
    """Соединение данных многие-ко-многим"""
    many_to_many_temp = [
        (f.name, fd.faculty_id, fd.department_id)
        for f in facs
        for fd in links
        if f.id == fd.faculty_id
    ]

    return [
        (d.name, faculty_name)
        for faculty_name, faculty_id, department_id in many_to_many_temp
```

```
        for d in deps if d.id == department_id
    ]


def task_a1(one_to_many):
    """Решение задания А1"""
    return sorted(one_to_many, key=itemgetter(2))


def task_a2(facs, one_to_many):
    """Решение задания А2"""
    res = []
    for f in facs:
        f_deps = list(filter(lambda i: i[2] == f.name, one_to_many))
        if f_deps:
            avg_budget = f.budget / len(f_deps)
            res.append((f.name, avg_budget))

    return sorted(res, key=itemgetter(1), reverse=True)


def task_a3(many_to_many, keyword):
    """Решение задания А3"""
    res = {}
    # Ищем уникальные кафедры
    all_departments = set([d_name for d_name, _ in many_to_many])

    for d_name in all_departments:
        if keyword in d_name.lower():
            d_faculties = list(filter(lambda i: i[0] == d_name, many_to_many))
            if d_faculties:
                faculty_names = [faculty for _, faculty in d_faculties]
                res[d_name] = faculty_names
    return res


# --- Основной запуск программы ---

def main():
    one_to_many = get_one_to_many(faculties, departments)
    many_to_many = get_many_to_many(faculties, departments, faculties_departments)

    print("Задание А1")
    for i in task_a1(one_to_many):
        print(i)

    print("\nЗадание А2")
    for i in task_a2(faculties, one_to_many):
```

```
        print(i)

    print("\nЗадание А3")
    print(task_a3(many_to_many, 'кафедра'))


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

## Файл tests.py

```
import unittest
from main import *


class TestRK2(unittest.TestCase):
    def setUp(self):
        """Подготовка тестовых данных перед каждым тестом"""
        self.faculties = [
            Faculty(1, 'F1', 100),
            Faculty(2, 'F2', 200),
        ]
        self.departments = [
            Department(1, 'D1-test', 1),
            Department(2, 'D2', 1),
            Department(3, 'D3-test', 2),
        ]
        self.fac_deps = [
            FacultyDepartment(1, 1),
            FacultyDepartment(1, 2),
            FacultyDepartment(2, 3),
        ]

        self.one_to_many = get_one_to_many(self.faculties, self.departments)
        self.many_to_many = get_many_to_many(self.faculties, self.departments,
self.fac_deps)

    def test_task_a1(self):
        """Тест А1: Проверка связи один-ко-многим и сортировки"""
        result = task_a1(self.one_to_many)
        # Ожидаем, что D1 привязан к F1, D2 к F1, D3 к F2
        # И сортировка по имени факультета (F1, F1, F2)
        self.assertEqual(result[0][2], 'F1')
        self.assertEqual(result[2][2], 'F2')
        self.assertEqual(len(result), 3)

    def test_task_a2(self):
        """Тест А2: Проверка расчета среднего бюджета"""
        # У F1 две кафедры (D1, D2), бюджет 100. Среднее = 50.
        # У F2 одна кафедра (D3), бюджет 200. Среднее = 200.
        # Сортировка по убыванию: сначала F2, потом F1.
        result = task_a2(self.faculties, self.one_to_many)

        self.assertEqual(result[0][0], 'F2')
        self.assertEqual(result[0][1], 200.0)

        self.assertEqual(result[1][0], 'F1')
        self.assertEqual(result[1][1], 50.0)

    def test_task_a3(self):
        """Тест А3: Поиск кафедр по названию"""
        # Ищем кафедры, где есть 'test'
        result = task_a3(self.many_to_many, 'test')

        # Должны найтись D1-test и D3-test
        self.assertIn('D1-test', result)
        self.assertIn('D3-test', result)
        self.assertNotIn('D2', result)


if __name__ == '__main__':
```

```
    unittest.main()
```

## Скриншот работы:

```
(.venv) PS C:\Users\Николай\Desktop\Python_Course_3Sem\RK2-3Sem> python -m unittest tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```