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 №2
“Разработка тестов на языке Python”

Выполнил:
Студент группы ИУ5-34Б
Савинский А. Ю.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Код main.py

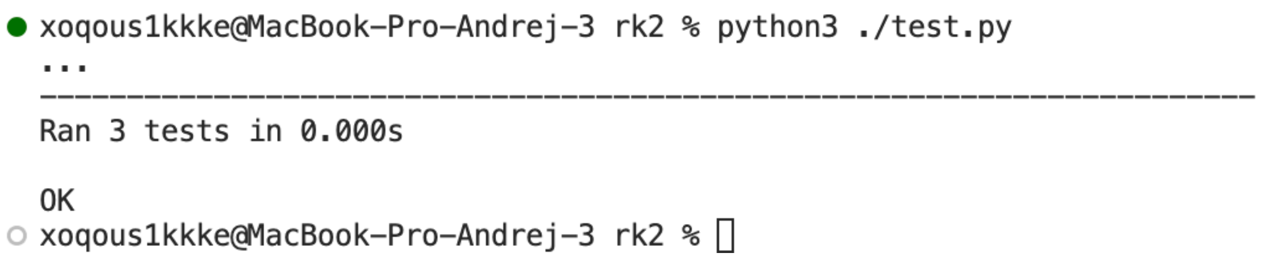
```
from typing import List, Tuple


class University:
    def __init__(self, id_university: int, name: str):
        self.id_university = id_university
        self.name = name


class Faculty:
    def __init__(self, id_faculty: int, name: str, students: int, id_university: int):
        self.id_faculty = id_faculty
        self.name = name
        self.students = students
        self.id_university = id_university


class FacultyUniversity:
    def __init__(self, id_faculty: int, id_university: int):
        self.id_faculty = id_faculty
        self.id_university = id_university


universities = [
    University(1, "МГТУ им. Н. Э. Баумана"),
    University(2, "ВШЭ"),
    University(3, "МФТИ"),
]

faculties = [
    Faculty(1, "Информатика и системы управления", 1800, 1),
    Faculty(2, "Компьютерные науки", 1500, 2),
    Faculty(3, "Прикладная математика и информатика", 1600, 3),
    Faculty(4, "Физика", 1400, 3),
    Faculty(5, "Радиотехника", 900, 1),
    Faculty(6, "Инженерный бизнес и менеджмент", 1200, 1),
]

faculty_universities = [
    FacultyUniversity(4, 3),
    FacultyUniversity(4, 2),
    FacultyUniversity(5, 1),
    FacultyUniversity(5, 3),
    FacultyUniversity(1, 1),
    FacultyUniversity(2, 2),
    FacultyUniversity(3, 3),
    FacultyUniversity(6, 1),
]


def get_faculties_starting_with(letter: str,
                                faculties: List[Faculty],
                                universities: List[University]) -> List[Tuple[str,
str]]:
    result = []
    for f in faculties:
        if f.name.startswith(letter):
            for u in universities:
                if f.id_university == u.id_university:
                    result.append((f.name, u.name))
    return result


def get_universities_with_min_students(faculties: List[Faculty],
                                       universities: List[University]) ->
List[Tuple[str, int]]:
    result = []
    for u in universities:
        students_list = []
        for f in faculties:
            if f.id_university == u.id_university:
                students_list.append(f.students)
        if students_list:
            result.append((u.name, min(students_list)))
```

```python
    return sorted(result, key=lambda x: x[1])


def get_faculties_and_universities_many_to_many(faculties: List[Faculty],
                                                universities: List[University],
                                                faculty_universities:
List[FacultyUniversity]) -> List[Tuple[str, str]]:
    result = []
    for fu in faculty_universities:
        for f in faculties:
            for u in universities:
                if fu.id_faculty == f.id_faculty and fu.id_university ==
u.id_university:
                    result.append((f.name, u.name))
    return sorted(result, key=lambda x: x[0])


def main():
    q1 = get_faculties_starting_with("И", faculties, universities)
    q2 = get_universities_with_min_students(faculties, universities)
    q3 = get_faculties_and_universities_many_to_many(faculties, universities,
faculty_universities)

    print("Задание 1: Факультеты на «И» и их университеты")
    for fac, uni in q1:
        print(f"- {fac} — {uni}")

    print("\nЗадание 2: Университеты с минимальным числом студентов на факультете")
    for uni, mn in q2:
        print(f"- {uni}: {mn}")

    print("\nЗадание 3: Многие-ко-многим (факультет—университет)")
    for fac, uni in q3:
        print(f"- {fac} — {uni}")


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

Код test.py

```python
import unittest
from main import (
    University, Faculty, FacultyUniversity,
    get_faculties_starting_with,
    get_universities_with_min_students,
    get_faculties_and_universities_many_to_many
)


class TestFacultyUniversityLogic(unittest.TestCase):
    def setUp(self):
        self.universities = [
            University(1, "МГТУ им. Н. Э. Баумана"),
            University(2, "ВШЭ"),
            University(3, "МФТИ"),
        ]

        self.faculties = [
            Faculty(1, "Информатика и системы управления", 1800, 1),
            Faculty(2, "Компьютерные науки", 1500, 2),
            Faculty(3, "Прикладная математика и информатика", 1600, 3),
            Faculty(4, "Физика", 1400, 3),
            Faculty(5, "Радиотехника", 900, 1),
            Faculty(6, "Инженерный бизнес и менеджмент", 1200, 1),
        ]

        self.faculty_universities = [
            FacultyUniversity(4, 3),
            FacultyUniversity(4, 2),
            FacultyUniversity(5, 1),
            FacultyUniversity(5, 3),
            FacultyUniversity(1, 1),
            FacultyUniversity(2, 2),
            FacultyUniversity(3, 3),
            FacultyUniversity(6, 1),
        ]

    def test_faculties_starting_with_I(self):
        result = get_faculties_starting_with("И", self.faculties, self.universities)
        expected = [
            ("Информатика и системы управления", "МГТУ им. Н. Э. Баумана"),
            ("Инженерный бизнес и менеджмент", "МГТУ им. Н. Э. Баумана"),
        ]
        self.assertEqual(result, expected)

    def test_min_students_by_university(self):
        result = get_universities_with_min_students(self.faculties, self.universities)
        expected = [
            ("МГТУ им. Н. Э. Баумана", 900),
            ("МФТИ", 1400),
            ("ВШЭ", 1500),
        ]
        self.assertEqual(result, expected)

    def test_many_to_many_relationship(self):
        result = get_faculties_and_universities_many_to_many(
            self.faculties, self.universities, self.faculty_universities
        )
        expected = [
            ("Инженерный бизнес и менеджмент", "МГТУ им. Н. Э. Баумана"),
            ("Информатика и системы управления", "МГТУ им. Н. Э. Баумана"),
            ("Компьютерные науки", "ВШЭ"),
            ("Прикладная математика и информатика", "МФТИ"),
            ("Радиотехника", "МГТУ им. Н. Э. Баумана"),
            ("Радиотехника", "МФТИ"),
            ("Физика", "МФТИ"),
            ("Физика", "ВШЭ"),
        ]
        self.assertEqual(result, expected)


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

Результат работы программы



```
● xoqous1kkke@MacBook-Pro-Andrej-3 rk2 % python3 ./test.py
  ...
  ----------------------------------------------------------------------
  Ran 3 tests in 0.000s

  OK
○ xoqous1kkke@MacBook-Pro-Andrej-3 rk2 % 
```