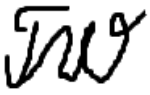
Московский государственный технический университет
им. Н.Э. Баумана

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

Дисциплина «Парадигмы и конструкции языков программирования»

Отчет по РК2



Выполнил:
студент группы ИУ5-36Б
Германенко И. М.
Преподаватель:
Гапанюк Ю.Е.

Москва, 2025

Описание задания:

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 кода:

RK1_refactored.py:

```python
# RK1_refactored.py

from operator import itemgetter
from typing import List, Tuple, Dict

class Student:
    def __init__(self, id: int, fio: str, scholarship: float, group_id: int):
        self.id = id
        self.fio = fio
        self.scholarship = scholarship
        self.group_id = group_id

class Group:
    def __init__(self, id: int, name: str):
        self.id = id
        self.name = name

class StudentGroup:
    def __init__(self, student_id: int, group_id: int):
        self.student_id = student_id
        self.group_id = group_id


def get_one_to_many(students: List[Student], groups: List[Group]) ->
List[Tuple[str, float, str]]:
    return [
        (s.fio, s.scholarship, g.name)
        for g in groups
        for s in students
        if s.group_id == g.id
    ]


def get_many_to_many(students: List[Student], groups: List[Group],
students_groups: List[StudentGroup]) -> List[Tuple[str, float, str]]:
    many_to_many_temp = [
        (g.name, sg.student_id)
        for g in groups
        for sg in students_groups
        if g.id == sg.group_id
    ]
```

```
    return [
        (s.fio, s.scholarship, group_name)
        for group_name, student_id in many_to_many_temp
        for s in students
        if s.id == student_id
    ]


def task_d1(one_to_many: List[Tuple[str, float, str]]) -> List[Tuple[str, str]]:
    """Студенты с фамилией на 'ов' и их группы (один-ко-многим)"""
    return [(fio, group) for fio, _, group in one_to_many if fio.endswith('ов')]


def task_d2(one_to_many: List[Tuple[str, float, str]], groups: List[Group]) ->
List[Tuple[str, float]]:
    """Средняя стипендия по группам, отсортировано по стипендии (один-ко-
многим)"""
    result = []
    for g in groups:
        group_students = [sch for fio, sch, grp in one_to_many if grp == g.name]
        if group_students:
            avg = sum(group_students) / len(group_students)
            result.append((g.name, round(avg, 2)))
    return sorted(result, key=itemgetter(1))


def task_d3(many_to_many: List[Tuple[str, float, str]], groups: List[Group]) ->
Dict[str, List[str]]:
    """Группы на 'А' и студенты в них (многие-ко-многим)"""
    result = {}
    for g in groups:
        if g.name.startswith('А'):
            students_in_group = [fio for fio, _, grp in many_to_many if grp ==
g.name]
            result[g.name] = students_in_group
    return result
```

test_rk1:

```
# test_rk1.py

import unittest
from RK1_refactored import (
    Student, Group, StudentGroup,
    get_one_to_many, get_many_to_many,
    task_d1, task_d2, task_d3
)


class TestRK1(unittest.TestCase):
```

```python
    def setUp(self):
        """Подготовка данных для всех тестов"""
        self.groups = [
            Group(1, 'ИУ5-31Б'),
            Group(2, 'ИУ5-32Б'),
            Group(3, 'АК4-01Б'),
            Group(11, 'АЭ7-11М'),
        ]
        self.students = [
            Student(1, 'Петров', 2500, 1),
            Student(2, 'Сидоров', 3000, 2),
            Student(3, 'Иванов', 3500, 2),
            Student(4, 'Смирнов', 2500, 3),
            Student(5, 'Кузнецов', 4000, 3),
        ]
        self.students_groups = [
            StudentGroup(1, 1),
            StudentGroup(2, 2),
            StudentGroup(3, 2),
            StudentGroup(4, 3),
            StudentGroup(5, 3),
            StudentGroup(1, 11),
            StudentGroup(2, 11),
            StudentGroup(4, 11),
        ]

        self.one_to_many = get_one_to_many(self.students, self.groups)
        self.many_to_many = get_many_to_many(self.students, self.groups,
self.students_groups)

    def test_task_d1(self):
        """Тест задания Д1: фамилии на 'ов'"""
        result = task_d1(self.one_to_many)
        expected = [('Петров', 'ИУ5-31Б'), ('Сидоров', 'ИУ5-32Б'), ('Иванов',
'ИУ5-32Б'), ('Смирнов', 'АК4-01Б')]
        self.assertEqual(result, expected)

    def test_task_d2(self):
        """Тест задания Д2: средняя стипендия по группам"""
        result = task_d2(self.one_to_many, self.groups)
        expected = [
            ('ИУ5-31Б', 2500.0),
            ('ИУ5-32Б', 3250.0),
            ('АК4-01Б', 3250.0),
        ]
        self.assertEqual(result, expected)

    def test_task_d3(self):
        """Тест задания Д3: группы на 'А' и их студенты"""
        result = task_d3(self.many_to_many, self.groups)
        expected = {
```

```
            'АК4-01Б': ['Смирнов', 'Кузнецов'],
            'АЭ7-11М': ['Петров', 'Сидоров', 'Смирнов']
        }
        self.assertEqual(result, expected)


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

Скриншоты: