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 №2

«**Модульное тестирование**»

Выполнил:

Студент группы ИУ5-31Б

Баринов Егор

Проверил:

Гапанюк Ю. Е.

2025 г.

Листинг программы

refactored_rk1_varA2.py

refactored_rk1_varA2.py

```python
from operator import itemgetter

class Student:
    def __init__(self, id, last_name, score, class_id):
        self.id = id
        self.last_name = last_name
        self.score = score
        self.class_id = class_id

class SchoolClass:
    def __init__(self, id, name):
        self.id = id
        self.name = name

class StudentClass:
    def __init__(self, class_id, student_id):
        self.class_id = class_id
        self.student_id = student_id

class DataService:
    def __init__(self):
        self.classes = [
            SchoolClass(1, 'математический класс'),
            SchoolClass(2, 'физический класс'),
            SchoolClass(3, 'химический класс'),
            SchoolClass(11, 'математический кружок'),
            SchoolClass(22, 'физический кружок'),
            SchoolClass(33, 'химический кружок'),
        ]
        self.students = [
            Student(1, 'Иванов', 85, 1),
            Student(2, 'Петров', 92, 2),
            Student(3, 'Сидоров', 78, 3),
            Student(4, 'Козлов', 95, 1),
            Student(5, 'Смирнов', 88, 2),
        ]
        self.students_classes = [
            StudentClass(1, 1),
            StudentClass(2, 2),
            StudentClass(3, 3),
            StudentClass(1, 4),
            StudentClass(2, 5),
            StudentClass(11, 1),
            StudentClass(22, 2),
            StudentClass(33, 3),
            StudentClass(11, 4),
            StudentClass(22, 5),
        ]

```

```python
class StudentClassService:
    def __init__(self, data_service):
        self.data_service = data_service

    def get_one_to_many(self):
        return [
            (s.last_name, s.score, c.name)
            for c in self.data_service.classes
            for s in self.data_service.students
            if s.class_id == c.id
        ]

    def get_many_to_many(self):
        many_to_many_temp = [
            (c.name, sc.class_id, sc.student_id)
            for c in self.data_service.classes
            for sc in self.data_service.students_classes
            if c.id == sc.class_id
        ]
        return [
            (s.last_name, s.score, class_name)
            for class_name, _, student_id in many_to_many_temp
            for s in self.data_service.students
            if s.id == student_id
        ]

    def task_a1(self):
        one_to_many = self.get_one_to_many()
        return sorted(one_to_many, key=itemgetter(2, 0))

    def task_a2(self):
        one_to_many = self.get_one_to_many()
        class_total_scores = []
        for c in self.data_service.classes:
            c_students = [item for item in one_to_many if item[2] == c.name]
            if c_students:
                total_score = sum(score for _, score, _ in c_students)
                class_total_scores.append((c.name, total_score))
        return sorted(class_total_scores, key=itemgetter(1), reverse=True)

    def task_a3(self):
        many_to_many = self.get_many_to_many()
        result = {}
        for c in self.data_service.classes:
            if 'класс' in c.name.lower():
                c_students = [item for item in many_to_many if item[2] == c.name]
                students_names = [student_name for student_name, _, _ in c_students]
                result[c.name] = students_names
        return result

```

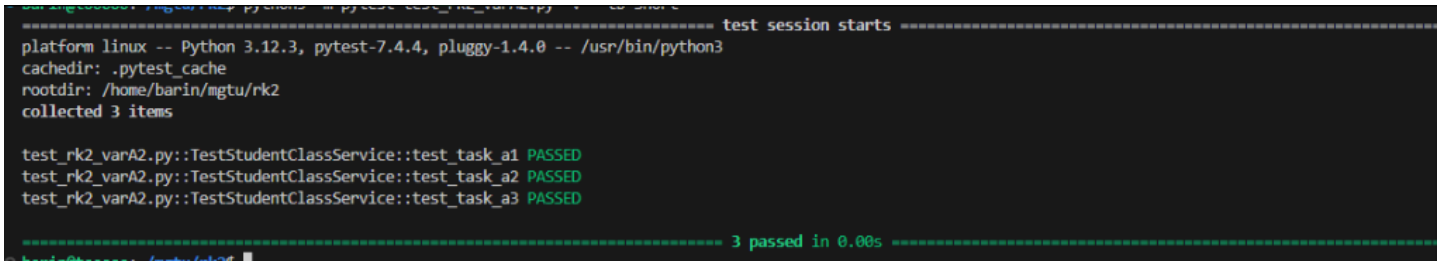
```python
def print_results(service):
    print('Задание A1')
    print('Список всех связанных школьников и классов, отсортированный по классам:')
    for last_name, score, class_name in service.task_a1():
        print(f'Школьник: {last_name}, Баллы: {score}, Класс: {class_name}')

    print('\n' + '=' * 50)
    print('Задание A2')
    print('Список классов с суммарными баллами школьников, отсортированный по суммарным баллам:')
    for class_name, total_score in service.task_a2():
        print(f'Класс: {class_name}, Суммарные баллы: {total_score}')

    print('\n' + '=' * 50)
    print('Задание A3')
    print('Список всех классов, у которых в названии присутствует слово "класс", и список школьников в них:')
    for class_name, students_list in service.task_a3().items():
        print(f'Класс: {class_name}')
        print(f'  Школьники: {", ".join(students_list)}')

def main():
    data_service = DataService()
    student_class_service = StudentClassService(data_service)
    print_results(student_class_service)

if __name__ == '__main__':
    main()

```

test_rk2_varA2.py

test_rk2_varA2.py

```python
import pytest
from refactored_rk1_varA2 import DataService, StudentClassService

class TestStudentClassService:
    @pytest.fixture
    def service(self):
        return StudentClassService(DataService())

    def test_task_a1(self, service):
        result = service.task_a1()
        assert len(result) == 5
        class_names = [class_name for _, _, class_name in result]
        assert class_names == sorted(class_names)
        assert result[0][2] == 'математический класс'
        assert result[0][0] == 'Иванов'
        assert result[0][1] == 85
        math_class_items = [item for item in result if item[2] == 'математический класс']
        math_students = [student for student, _, _ in math_class_items]
        assert math_students == ['Иванов', 'Козлов']

    def test_task_a2(self, service):
        result = service.task_a2()
        assert len(result) == 3
        total_scores = [score for _, score in result]
        assert total_scores == sorted(total_scores, reverse=True)
        result_dict = dict(result)
        assert result_dict['математический класс'] == 180
        assert result_dict['физический класс'] == 180
        assert result_dict['химический класс'] == 78

    def test_task_a3(self, service):
        result = service.task_a3()
        assert isinstance(result, dict)
        assert len(result) == 3
        assert 'математический класс' in result
        assert 'физический класс' in result
        assert 'химический класс' in result
        assert result['математический класс'] == ['Иванов', 'Козлов']
        assert result['физический класс'] == ['Петров', 'Смирнов']
        assert result['химический класс'] == ['Сидоров']

if __name__ == '__main__':
    pytest.main([__file__, '-v'])

```

Результат выполнения:


test session starts
platform linux -- Python 3.12.3, pytest-7.4.4, pluggy-1.4.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/barin/mgtu/rk2
collected 3 items
test_rk2_varA2.py::TestStudentClassService::test_task_a1 PASSED
test_rk2_varA2.py::TestStudentClassService::test_task_a2 PASSED
test_rk2_varA2.py::TestStudentClassService::test_task_a3 PASSED
3 passed in 0.00s