Талаев А.П.

ИУ5-35Б

РК 2

Задание:

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).

Выполнение:

```
import unittest
class Student:
    def __init__(self, student_id, last_name, points, group_id):
        self.student_id = student_id
        self.last_name = last_name
        self.points = points
        self.group_id = group_id
class Group:
    def __init__(self, group_id, name):
        self.group_id = group_id
        self.name = name
class StudentsGroups:
    def __init__(self, student_id, group_id):
        self.student_id = student_id
        self.group_id = group_id
groups = [
    Group(1, "A Группа"),
    Group(2, "B Группа"),
    Group(3, "C Группа"),
    Group(4, "D Группа"),
    Group(5, "A+ Группа")
]
students = [
    Student(1, "Иванов", 85, 1),
    Student(2, "Петров", 90, 1),
    Student(3, "Сидоров", 75, 2),
    Student(4, "Алексеев", 88, 3),
    Student(5, "Андреев", 92, 3),
    Student(7, "Андреев", 100, 5),
    Student(6, "Федоров", 80, 4)
]
students_groups = [
    StudentsGroups(1, 1),
    StudentsGroups(2, 1),
    StudentsGroups(3, 2),
    StudentsGroups(4, 3),
    StudentsGroups(5, 3),
    StudentsGroups(6, 4),
    StudentsGroups(7, 5)
]
def get_students_with_lastname_ending_with_ov(students, groups):
    result = []
    for student in students:
        # Проверяем, заканчивается ли фамилия на 'ов'
        if student.last_name.endswith('ов'):
            group_name = next((group.name for group in groups if group.group_id == student.group_id), None)
            if group_name:
                result.append((student.last_name, group_name))
    return result



def get_average_points_by_group(students, groups):
    group_points = {}
    group_counts = {}

    for student in students:
        group_id = student.group_id
        if group_id not in group_points:
            group_points[group_id] = 0
            group_counts[group_id] = 0
        group_points[group_id] += student.points
        group_counts[group_id] += 1

    average_points = []
    for group in groups:
        if group.group_id in group_points:  # Изменено на group_id
            avg = group_points[group.group_id] / group_counts[group.group_id]
            average_points.append((group.name, avg))

    # Сортируем по имени группы
    average_points.sort(key=lambda x: x[0])
    return average_points

def get_students_in_groups_starting_with_A(groups, students_groups, students):
    result = {}
    for group in groups:
        if group.name.startswith("A"):
            enrolled_students = [
                student.last_name for sg in students_groups
                for student in students
                if sg.group_id == group.group_id and sg.student_id ==
student.student_id
            ]
            result[group.name] = enrolled_students
    return result
class TestStudentFunctions(unittest.TestCase):

    def setUp(self):
        self.groups = [
            Group(1, "A Группа"),
            Group(2, "B Группа"),
            Group(3, "C Группа"),
            Group(4, "D Группа"),
            Group(5, "A+ Группа")
        ]

        self.students = [
            Student(1, "Иванов", 85, 1),
            Student(2, "Петров", 90, 1),
            Student(3, "Сидоров", 75, 2),
            Student(4, "Алексеев", 88, 3),
            Student(5, "Андреев", 92, 3),
            Student(7, "Андреев", 100, 5),
            Student(6, "Федоров", 80, 4)
        ]

        self.students_groups = [
            StudentsGroups(1, 1),
            StudentsGroups(2, 1),
            StudentsGroups(3, 2),
            StudentsGroups(4, 3),
            StudentsGroups(5, 3),
            StudentsGroups(6, 4),
            StudentsGroups(7, 5)
        ]

    def test_get_students_with_lastname_ending_with_ov(self):
        expected_result = [
            ('Иванов', 'A Группа'),
            ('Петров', 'A Группа'),
            ('Сидоров', 'B Группа'),
            ('Федоров', 'D Группа')
        ]
        result = get_students_with_lastname_ending_with_ov(self.students,
self.groups)
        self.assertEqual(result, expected_result)

    def test_get_average_points_by_group(self):
        expected_result = [
            ('A Группа', 87.5),
            ('A+ Группа', 100.0),
            ('B Группа', 75.0),
            ('C Группа', 90.0),
            ('D Группа', 80.0)
        ]
        result = get_average_points_by_group(self.students, self.groups)
        self.assertEqual(result, expected_result)

    def test_get_students_in_groups_starting_with_A(self):
        expected_result = {
            'A Группа': ['Иванов', 'Петров'],
            'A+ Группа': ['Андреев']
        }
        result = get_students_in_groups_starting_with_A(self.groups,
self.students_groups, self.students)
        self.assertEqual(result, expected_result)

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

Результаты:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```