Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по РК №2

Вариант запросов Д

Вариант предметной области 1

Выполнил:
студент группы ИУ5-35Б
Абалуев Антон

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

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:

```
class Student:
    """Студент"""

    def __init__(self, id, surname, average_rating, group_id):
        self.id = id
        self.surname = surname
        self.average_rating = average_rating
        self.group_id = group_id


class Group:
    """Группа"""

    def __init__(self, id, group_name):
        self.id = id
        self.group_name = group_name


class StudentGroup:
    """
    'Студенты группы' для реализации связи многие-ко-многим
    """

    def __init__(self, group_id, student_id):
        self.group_id = group_id
        self.student_id = student_id


def join_one_to_many(groups, students):
    return [(y.surname, y.average_rating, x.group_name)
            for x in groups
            for y in students
            if y.group_id == x.id]


def join_many_to_many(groups, student_group, students):
    many_to_many_temp = [(x.group_name, y.group_id, y.student_id)
                         for x in groups
                         for y in student_group
                         if x.id == y.group_id]

    return [(x.surname, x.average_rating, group_name)
            for group_name, group_id, student_id in many_to_many_temp
            for x in students if x.id == student_id]


def filter_students_by_condition(data, condition):
    return [(student_surname, student_group_id)
            for student_surname, student_average_rating, student_group_id in
data
            if condition(student_surname)]
```

```python
def calculate_group_average_ratings(groups, one_to_many_data):
    group_average_ratings = {}
    for g in groups:
        g_students = list(filter(lambda i: i[2] == g.group_name,
one_to_many_data))
        if len(g_students) > 0:
            g_students_average_ratings = [x for _, x, _ in g_students]
            group_average_ratings[g.group_name] =
(sum(g_students_average_ratings) / len(g_students_average_ratings))
    return sorted(group_average_ratings.items(), key=lambda item: item[1])


def get_students_surnames_by_condition(groups, many_to_many_data, condition):
    result = {}
    for g in groups:
        if condition(g.group_name):
            g_students = list(filter(lambda i: i[2] == g.group_name,
many_to_many_data))
            g_students_surnames = [x for x, _, _ in g_students]
            result[g.group_name] = g_students_surnames
    return result

students = [Student(1, 'Ivanov', 3.4, 1), Student(2, 'Baranov', 4.2, 2)]
groups = [Group(1, 'IU5-31B'), Group(2, 'IU5-32B')]
student_group = [
    StudentGroup(1, 1),
    StudentGroup(2, 2),
]

# Вызов функций
print("Task 1 Result:", filter_students_by_condition(join_one_to_many(groups,
students),  lambda surname: surname.endswith('ov')))
print("Task 2 Result:", calculate_group_average_ratings(groups,
join_one_to_many(groups, students)))
print("Task 3 Result:", get_students_surnames_by_condition(groups,
join_many_to_many(groups, student_group, students), lambda group_name:
group_name[0] == 'I'))

import unittest

class TestFilterStudentsByLastName(unittest.TestCase):
    def test_filter_by_lastname(self):
        students = [Student(1, 'Ivanov', 3.4, 1), Student(2, 'Baranov', 4.2,
2)]
        groups = [Group(1, 'IU5-31B'), Group(2, 'IU5-32B')]
        result = filter_students_by_condition(join_one_to_many(groups,
students),  lambda surname: surname.endswith('ov'))
        self.assertEqual(result, [('Ivanov', 'IU5-31B'), ('Baranov', 'IU5-
32B')])

class TestCalculateGroupAverageRatings(unittest.TestCase):
    def test_average_ratings(self):
        students = [Student(1, 'Ivanov', 3.4, 1), Student(2, 'Baranov', 4.2,
2)]
        groups = [Group(1, 'IU5-31B'), Group(2, 'IU5-32B')]
        result = calculate_group_average_ratings(groups,
join_one_to_many(groups, students))
        self.assertEqual(result, [('IU5-31B', 3.4), ('IU5-32B', 4.2)])

class TestGetStudentsSurnamesByCondition(unittest.TestCase):
    def test_get_students_surnames(self):
        students = [Student(1, 'Ivanov', 3.4, 1), Student(2, 'Baranov', 4.2,
2)]
        groups = [Group(1, 'IU5-31B'), Group(2, 'IU5-32B')]
```

```
        student_group = [
            StudentGroup(1, 1),
            StudentGroup(2, 2),
        ]
        result = get_students_surnames_by_condition(groups,
join_many_to_many(groups, student_group, students), lambda group_name:
group_name[0] == 'I')
        self.assertEqual(result,  {'IU5-31B': ['Ivanov'], 'IU5-32B':
['Baranov']})

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

Вывод:

```
Task 1 Result: [('Ivanov', 'IU5-31B'), ('Baranov', 'IU5-32B')]
Task 2 Result: [('IU5-31B', 3.4), ('IU5-32B', 4.2)]
Task 3 Result: {'IU5-31B': ['Ivanov'], 'IU5-32B': ['Baranov']}
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```