Факультет «Информатика и системы управления»

Кафедра ИУ5 «Система обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Рубежный контроль №2

Выполнил:
студент группы ИУ5-32Б:
Бокатуев М. С.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2023 г.

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).

Код программы:
Файл RK2.py

```
from collections import Counter
from collections import defaultdict

class Pupil:
    """Ученик"""
    def __init__(self, id, fio, rate, form_id):
        self.id = id
        self.fio = fio
        self.rate = rate
        self.form_id = form_id
class Form:
    """Класс"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
class PupilAndForm:
    def __init__(self, pupil_id, form_id):
        self.pupil_id = pupil_id
        self.form_id = form_id

def query_1(forms_data, pupils):
    sorted_pupils = sorted(pupils, key=lambda x: x.fio)
    forms_dict = {forms.id: forms.name for forms in forms_data}
    result = []
    for pupils in sorted_pupils:
        result.append(f"Школьник: {pupils.fio}, Класс:
{forms_dict.get(pupils.form_id, 'Неизвестный класс')}")
    return result

def query_2(forms_data, pupil_and_form_data):
    num_pupils = Counter(d.form_id for d in pupil_and_form_data)
    sorted_forms = sorted(forms_data, key=lambda x: num_pupils[x.id], re-
verse=True)
    result = []
    for form in sorted_forms:
        result.append(f"Класс: {form.name}, Количество учеников:
{num_pupils[form.id]}")
    return result

def query_3(forms_data, pupils, pupil_and_form_data):
    pupil_form_dict = defaultdict(list)
```

```
    forms_dict = {forms.id: forms.name for forms in forms_data}
    result = []
    for cd in pupil_and_form_data:
        pupil_form_dict[cd.pupil_id].append(cd.form_id)
    filtered_pupils = [pupil for pupil in pupils if pupil.fio.endswith("ов")]
    for pupils in filtered_pupils:
        result.append(f"Школьник: {pupils.fio}, Класс(ы): {',
'.join(forms_dict[id] for id in pupil_form_dict.get(pupils.id, []))}")
    return result
```

Файл tests.py

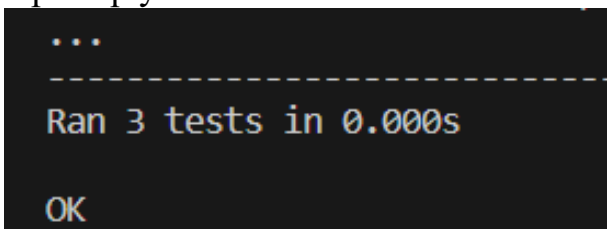
```
import unittest
from RK2 import Form, Pupil, PupilAndForm, query_1, query_2, query_3

forms_data = [
    Form(1, '9А'),
    Form(2, '9Б'),
    Form(3, '10А'),
    Form(4, '10Б'),
    Form(5, '11А'),
    Form(6, '11Б'),

    Form(7, '(другой)9А'),
    Form(8, '(другой)9Б'),
    Form(9, '(другой)10А'),
    Form(10, '(другой)10Б'),
    Form(11, '(другой)11А'),
    Form(12, '(другой)11Б'),
]

pupils = [
    Pupil(1, 'Попова', 5, 1),
    Pupil(2, 'Соколов', 4, 1),
    Pupil(3, 'Алексеев', 4, 1),

    Pupil(4, 'Орлов', 3, 2),
    Pupil(5, 'Кузьмин', 4, 2),
    Pupil(6, 'Егорова', 5, 2),

    Pupil(7, 'Белов', 4, 3),
    Pupil(8, 'Козлова', 5, 3),
    Pupil(9, 'Жукова', 5, 3),

    Pupil(10, 'Титов', 3, 4),
    Pupil(11, 'Смирнова', 4, 4),
    Pupil(12, 'Ефимов', 4, 4),

    Pupil(13, 'Соловьев', 3, 5),
    Pupil(14, 'Осипова', 5, 5),
    Pupil(15, 'Комаров', 5, 5),
```

```
    Pupil(16, 'Крылов', 4, 6),
    Pupil(17, 'Кузнецов', 4, 6),
    Pupil(18, 'Попов', 4, 6),
]

pupil_and_form_data = [
    PupilAndForm(1, 1),
    PupilAndForm(1, 7),
    PupilAndForm(2, 1),
    PupilAndForm(2, 7),
    PupilAndForm(3, 1),
    PupilAndForm(4, 2),
    PupilAndForm(4, 8),
    PupilAndForm(5, 2),
    PupilAndForm(5, 8),
    PupilAndForm(6, 2),
    PupilAndForm(7, 3),
    PupilAndForm(7, 9),
    PupilAndForm(8, 3),
    PupilAndForm(9, 3),
    PupilAndForm(10, 4),
    PupilAndForm(11, 4),
    PupilAndForm(12, 4),
    PupilAndForm(12, 10),
    PupilAndForm(13, 5),
    PupilAndForm(14, 5),
    PupilAndForm(15, 5),
    PupilAndForm(15, 11),
    PupilAndForm(16, 6),
    PupilAndForm(17, 6),
    PupilAndForm(18, 6),
    PupilAndForm(18, 12),
]

class TestCarParkQueries(unittest.TestCase):
    def test_query_1(self):
        result = query_1(forms_data, pupils)
        expected = [
            "Школьник: Алексеев, Класс: 9А",
            "Школьник: Белов, Класс: 10А",
            "Школьник: Егорова, Класс: 9Б",
            "Школьник: Ефимов, Класс: 10Б",
            "Школьник: Жукова, Класс: 10А",
            "Школьник: Козлова, Класс: 10А",
            "Школьник: Комаров, Класс: 11А",
            "Школьник: Крылов, Класс: 11Б",
            "Школьник: Кузнецов, Класс: 11Б",
            "Школьник: Кузьмин, Класс: 9Б",
            "Школьник: Орлов, Класс: 9Б",
```

```
            "Школьник: Осипова, Класс: 11А",
            "Школьник: Попов, Класс: 11Б",
            "Школьник: Попова, Класс: 9А",
            "Школьник: Смирнова, Класс: 10Б",
            "Школьник: Соколов, Класс: 9А",
            "Школьник: Соловьев, Класс: 11А",
            "Школьник: Титов, Класс: 10Б"
        ]
        self.assertEqual(result, expected)

    def test_query_2(self):
        result = query_2(forms_data, pupil_and_form_data)
        expected = [
            "Класс: 9А, Количество учеников: 3",
            "Класс: 9Б, Количество учеников: 3",
            "Класс: 10А, Количество учеников: 3",
            "Класс: 10Б, Количество учеников: 3",
            "Класс: 11А, Количество учеников: 3",
            "Класс: 11Б, Количество учеников: 3",
            "Класс: (другой)9А, Количество учеников: 2",
            "Класс: (другой)9Б, Количество учеников: 2",
            "Класс: (другой)10А, Количество учеников: 1",
            "Класс: (другой)10Б, Количество учеников: 1",
            "Класс: (другой)11А, Количество учеников: 1",
            "Класс: (другой)11Б, Количество учеников: 1"
        ]
        self.assertEqual(result, expected)

    def test_query_3(self):
        result = query_3(forms_data, pupils, pupil_and_form_data)
        expected = [
            "Школьник: Соколов, Класс(ы): 9А, (другой)9А",
            "Школьник: Орлов, Класс(ы): 9Б, (другой)9Б",
            "Школьник: Белов, Класс(ы): 10А, (другой)10А",
            "Школьник: Титов, Класс(ы): 10Б",
            "Школьник: Ефимов, Класс(ы): 10Б, (другой)10Б",
            "Школьник: Комаров, Класс(ы): 11А, (другой)11А",
            "Школьник: Крылов, Класс(ы): 11Б",
            "Школьник: Кузнецов, Класс(ы): 11Б",
            "Школьник: Попов, Класс(ы): 11Б, (другой)11Б"
        ]
        self.assertEqual(result, expected)

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

Результат:

Пример успешного выполнения теста



Пример неудачного прохождения теста