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 РК №2
Вариант запросов: В
Вариант предметной области: 2

Выполнил:
студент группы ИУ5-33Б
Абрамов Александр

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

Вариант запросов В.

1. «Школьник» и «Класс» связаны соотношением один-ко-многим. Выведите список всех школьников, фамилия которых начинается на букву «А».

2. «Школьник» и «Класс» связаны соотношением один-ко-многим. Выведите список классов с минимальным средним баллом в каждом классе, отсортированный по минимальному среднему баллу.

3. «Школьник» и «Класс» связаны соотношением многие-ко-многим. Выведите список всех связанных школьников и классов, отсортированный по школьникам, сортировка классов произвольная.

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).

Текст программы

Файл main.py

```python
class Student:
    def __init__(self, student_id: int, name: str, class_id: int, avg_rating: float):
        self._student_id = student_id
        self._name = name
        self._class_id = class_id
        self._avg_rating = avg_rating

    @property
    def student_id(self) -> int:
        return self._student_id

    @property
    def name(self) -> str:
        return self._name

    @property
    def class_id(self) -> int:
        return self._class_id

    @property
    def avg_rating(self) -> float:
        return self._avg_rating


class Class:
    def __init__(self, class_id: int, name: str):
        self._class_id = class_id
        self._name = name

    @property
    def class_id(self) -> int:
```

```python
        return self._class_id

    @property
    def name(self) -> str:
        return self._name


def query1(student_class: list):
    # Задание B1
    data = []
    for student, cls in student_class:
        if student.name.startswith('A'):
            data.append((student.name, cls.name))
    return data


def query2(student_class: list):
    # Задание B2
    data = []
    class_min_ratings = {}
    for student, cls in student_class:
        if cls.name in class_min_ratings:
            if student.avg_rating < class_min_ratings[cls.name]:
                class_min_ratings[cls.name] = student.avg_rating
        else:
            class_min_ratings[cls.name] = student.avg_rating

    sorted_classes = sorted(class_min_ratings.items(), key=lambda x: x[1])
    for group, min_rating in sorted_classes:
        data.append((group, min_rating))
    return data


def query3(student_class: list):
    # Задание B3
    data = []
    student_class.sort(key=lambda x: x[0].name)
    for student, cls in student_class:
        data.append((student.name, cls.name))
    return data


def generate_data():
    # Создаем объекты класса Class
    classes = [
        Class(1, "2Б"),
        Class(2, "10А"),
        Class(3, "5В")
    ]

    # Создаем объекты класса Student
    students = [
        Student(1, "Лупарев", 1, 4.9),
        Student(2, "Гукасян", 1, 3.8),
        Student(3, "Абрамов", 2, 4.5),
        Student(4, "Иноземцев", 2, 2.7),
        Student(5, "Барсукова", 3, 5.0)
    ]

    # Создаем список "Школьники и классы" для связи один-ко-многим
    student_class = [
        (students[0], classes[0]),
        (students[1], classes[0]),
```

```
        (students[2], classes[1]),
        (students[3], classes[1]),
        (students[4], classes[2])
    ]
    return classes, students, student_class


def execute_queries(student_class: list):
    print("Задание B1")
    for stud, cls in query1(student_class):
        print(f"{stud} - {cls}")
    print()

    print("Задание B2")
    for group, rating in query2(student_class):
        print(f"{group} - Минимальный рейтинг: {rating}")
    print()

    print("Задание B3")
    for stud, cls in query3(student_class):
        print(f"{stud} - {cls}")
    print()


def main():
    # Генерация данных
    classes, students, student_class = generate_data()

    # Запуск запросов
    execute_queries(student_class)


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

Файл TDDtests.py

```
import unittest
from main import *


# Тестирование класса "Студент"
class TestStudent(unittest.TestCase):

    def test_student_creation(self):
        student = Student(1, "Беляев", 2, 4.5)
        self.assertEqual(student.student_id, 1)
        self.assertEqual(student.name, "Беляев")
        self.assertEqual(student.class_id, 2)
        self.assertEqual(student.avg_rating, 4.5)


# Тестирование класса "Класс"
class TestClass(unittest.TestCase):

    def test_computer_classroom_creation(self):
        cls = Class(1, "1Б")
        self.assertEqual(cls.class_id, 1)
        self.assertEqual(cls.name, "1Б")


# Тестирование выполнения запросов
```

```python
class TestQueryExecution(unittest.TestCase):
    def setUp(self):
        self.classes, self.students, self.student_class = generate_data()

    # Тестирование запроса №1
    def test_query1(self):
        result = query1(self.student_class)
        self.assertEqual(result, [("Абрамов", "10А")])

    # Тестирование запроса №2
    def test_query2(self):
        result = query2(self.student_class)
        self.assertEqual(result, [("10А", 2.7), ("2Б", 3.8), ("5В", 5.0)])

    # Тестирование запроса №3
    def test_query3(self):
        result = query3(self.student_class)
        self.assertEqual(result,
                         [("Абрамов", "10А"), ("Барсукова", "5В"), ("Гукасян", "2Б"),
                          ("Иноземцев", "10А"), ("Лупарев", "2Б")])


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

Результат выполнения программы