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: Проверил:

студент группы ИУ5-35Б преподаватель каф. ИУ5

Абалуев Антон 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)]  
  
  
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:  
