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