Московский государственный технический

Университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по рубежному контролю №2

Выполнил:

студент группы ИУ5-51Б

Афанасьев Д. М.

Проверил: Гапанюк Е.Ю.

2022

**Условия рубежного контроля №2 по курсу БКИТ**

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).

**Измененный код РК1:**

from operator import itemgetter  
class Teacher:  
 def \_\_init\_\_(self, id, name, wage, id\_course):  
 self.id = id  
 self.name = name  
 self.wage = wage  
 self.id\_course = id\_course  
class Course:  
 def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
class Teacher\_and\_course:  
 def \_\_init\_\_(self, id\_teacher, id\_course):  
 self.id\_teacher = id\_teacher  
 self.id\_course = id\_course  
teachers = [  
 Teacher(1, "Александров", 1000, 1),  
 Teacher(2, "Иванов", 1500, 3),  
 Teacher(3, "Барановский", 2000, 2),  
 Teacher(4, "Егоров", 1600, 4),  
 Teacher(5, "Ченский", 1800, 2),  
 Teacher(6, "Соколов", 1300, 4),  
 Teacher(7, "Сороковиков", 1900, 1),  
 Teacher(8, "Смирнов", 1600, 3),  
]  
courses = [  
 Course(1, "физика"),  
 Course(2, "математика"),  
 Course(3, "химия"),  
 Course(4, "информатика")  
 ]  
teacherofcourse = [  
 Teacher\_and\_course(1, 1),  
 Teacher\_and\_course(2, 3),  
 Teacher\_and\_course(3, 2),  
 Teacher\_and\_course(4, 4),  
 Teacher\_and\_course(5, 2),  
 Teacher\_and\_course(6, 4),  
 Teacher\_and\_course(7, 3),  
 Teacher\_and\_course(8, 1),  
 ]  
def one\_to\_many(teachers, courses):  
 return [(t.name, t.wage, c.name)  
 for t in teachers  
 for c in courses  
 if t.id\_course == c.id]  
def many\_to\_many(teachers, courses):  
 many\_to\_many\_tmp = [(c.name, tc.id\_course, tc.id\_teacher)  
 for c in courses  
 for tc in teacherofcourse  
 if c.id == tc.id\_course]  
 return [(t.id, id\_course)  
 for name, id\_course, id\_teacher in many\_to\_many\_tmp  
 for t in teachers  
 if t.id == id\_teacher]  
  
def many\_to\_many(teachers, courses):  
 many\_to\_many\_temp = [(c.name, tc.id\_course, tc.id\_teacher)  
 for c in courses  
 for tc in teacherofcourse  
 if c.id == tc.id\_course]  
 return [(t.name, t.wage, course\_name)  
 for course\_name, id\_course, id\_teacher in many\_to\_many\_temp  
 for t in teachers  
 if t.id == id\_teacher]  
  
def A1(courses, teachers) -> list:  
 res1 = sorted(one\_to\_many(teachers, courses), key=itemgetter(2))  
 return list(res1)  
def A2(courses, teachers) -> list:  
 res2\_unsorted = []  
 for c in courses:  
 c\_courses = list(  
 filter(lambda i: i[2] == c.name, one\_to\_many(teachers, courses)))  
 if len(c\_courses) > 0:  
 c\_sizes = [sal for \_, sal, \_ in c\_courses]  
 c\_sizes\_sum = sum(c\_sizes) / 2  
 res2\_unsorted.append((c.name, c\_sizes\_sum))  
 return sorted(res2\_unsorted, key=itemgetter(1), reverse=True)  
  
def A3(courses, teachers):  
 res3 = {}  
 for c in courses:  
 if 'физика' in c.name:  
 c\_courses = list(  
 filter(lambda i: i[2] == c.name, many\_to\_many(teachers, courses)))  
 c\_courses\_names = [x for x, \_, \_ in c\_courses]  
 res3[c.name] = c\_courses\_names  
 return res3  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('Task А1')  
 print(A1(courses, teachers))  
 print('Task А2')  
 print(A2(courses, teachers))  
 print('Task А3')  
 print(A3(courses, teachers))

**Тестирование:**

|  |
| --- |
| import unittest from changed\_rk1 import \*  class rk\_test(unittest.TestCase):  def setUp(self):  self.courses = [  Course(1, "физика"),  Course(2, "математика"),  Course(3, "химия"),  Course(4, "информатика")  ]  self.teachers = [  Teacher(1, "Александров", 1000, 1),  Teacher(2, "Иванов", 1500, 3),  Teacher(3, "Барановский", 2000, 2),  Teacher(4, "Егоров", 1600, 4),  Teacher(5, "Ченский", 1800, 2),  Teacher(6, "Соколов", 1300, 4),  Teacher(7, "Сороковиков", 1900, 1),  Teacher(8, "Смирнов", 1600, 3),  ]  self.teachers\_and\_courses = [  Teacher\_and\_course(1, 1),  Teacher\_and\_course(2, 3),  Teacher\_and\_course(3, 2),  Teacher\_and\_course(4, 4),  Teacher\_and\_course(5, 2),  Teacher\_and\_course(6, 4),  Teacher\_and\_course(7, 3),  Teacher\_and\_course(8, 1),  ]  def test\_A1(self):  expected\_result = [  ('Егоров', 1600, 'информатика'),  ('Соколов', 1300, 'информатика'),  ('Барановский', 2000, 'математика'),  ('Ченский', 1800, 'математика'),  ('Александров', 1000, 'физика'),  ('Сороковиков', 1900, 'физика'),  ('Иванов', 1500, 'химия'),  ('Смирнов', 1600, 'химия'),  ]  result = A1(self.courses, self.teachers)  self.assertEqual(result, expected\_result)  def test\_A2(self):  expected\_result = [  ('математика', 1900.0),  ('химия', 1550.0),  ('физика', 1450.0),  ('информатика', 1450.0)  ]  result = A2(self.courses, self.teachers)  self.assertEqual(result, expected\_result)  def test\_A3(self):  expected\_result = {  'физика': ['Александров', 'Смирнов'],  }  result = A3(self.courses, self.teachers)  self.assertEqual(result, expected\_result) if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() |

**Результат тестирования:**



