**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»

Отчёт по РК 2

Выполнил: Проверил:

студент группы ИУ5-31Б преподаватель каф. ИУ5

Климов Никита Гапанюк Юрий

Сергеевич Евгеньевич

Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2021 г.

Задание:

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).

Текст программы:

main.py

from operator import itemgetter  
from typing import List, Tuple, Any  
  
  
class microprocessor:  
 *"""Микропроцессор"""* def \_\_init\_\_(self, id, title, sal, comp\_id):  
 self.id = id  
 self.title = title  
 self.sal = sal  
 self.comp\_id = comp\_id  
  
  
class Computer:  
 *"""Компьютер"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class ComputerMicroprocessor:  
 *"""  
 '' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, comp\_id, mic\_id):  
 self.comp\_id = comp\_id  
 self.mic\_id = mic\_id  
  
# Компьютеры  
Computers = [  
 Computer(1, 'MSI'),  
 Computer(2, 'ASUS'),  
 Computer(3, 'Lenovo'),  
 Computer(4, 'Imac'),  
 Computer(5, 'Xiaomi'),  
]  
  
# Микропроцессоры  
Microprocessors = [  
 microprocessor(1, 'AMD A8', 253, 3),  
 microprocessor(2, 'AMD Ryzen 5', 222, 2),  
 microprocessor(3, 'Intel Core i5', 434, 5),  
 microprocessor(4, 'AMD Athlon', 370, 2),  
 microprocessor(5, 'Intel Core i7', 556, 3),  
]  
  
Microprocessor\_Computer = [  
 ComputerMicroprocessor(1, 3),  
 ComputerMicroprocessor(2, 2),  
 ComputerMicroprocessor(3, 5),  
 ComputerMicroprocessor(4, 2),  
 ComputerMicroprocessor(5, 3),  
  
 ComputerMicroprocessor(1, 2),  
 ComputerMicroprocessor(2, 4),  
 ComputerMicroprocessor(3, 1),  
 ComputerMicroprocessor(4, 4),  
 ComputerMicroprocessor(5, 1),  
]  
def res\_11(one\_to\_many):  
 res11 = []  
 for title, \_, Computer in one\_to\_many:  
 if title[0] == "A":  
 res11.append((title, Computer))  
 return res11  
def res\_12(one\_to\_many):  
 res12 = [[one\_to\_many[0][2], one\_to\_many[0][1]]]  
 for title, sal, name\_ in one\_to\_many:  
 if name\_ == res12[len(res12) - 1][0]:  
 if sal < res12[len(res12) - 1][1]:  
 res12[len(res12) - 1][1] = sal  
 else:  
 res12.append([title, sal])  
 return sorted(res12, key=itemgetter(1))  
def res\_13(many\_to\_many):  
 res13 = []  
 for title, sal, Computer in many\_to\_many:  
 res13.append((title, Computer))  
 # print (res\_13)  
 return sorted(res13, key=itemgetter(0))  
  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
one\_to\_many = [(m.title, m.sal, c.name)  
 for c in Computers  
 for m in Microprocessors  
 if m.comp\_id == c.id]  
  
# Соединение данных многие-ко-многим  
many\_to\_many\_temp = [(c.name, cm.comp\_id, cm.mic\_id)  
 for c in Computers  
 for cm in Microprocessor\_Computer  
 if c.id == cm.comp\_id]  
  
many\_to\_many = [(c.title, c.sal, Computer\_name)  
 for Computer\_name, Computer\_id, microprocessor\_id in many\_to\_many\_temp  
 for c in Microprocessors if c.id == microprocessor\_id]  
  
print('Задание B1:')  
  
print(res\_11(one\_to\_many), "\n")  
  
print('Задание В2')  
  
print(res\_12(one\_to\_many), "\n")  
  
  
print('Задание В3')  
  
print(res\_13(many\_to\_many), "\n")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

tests.py

from main import microprocessor, Computer, ComputerMicroprocessor, res\_11, res\_12, res\_13  
from unittest import TestCase  
  
class Test(TestCase):  
 def selUp(self)->None:  
 self.Computers = [  
 Computer(1, 'MSI'),  
 Computer(2, 'ASUS'),  
 Computer(3, 'Lenovo'),  
 Computer(4, 'Imac'),  
 Computer(5, 'Xiaomi'),  
 ]  
 self.Microprocessors = [  
 microprocessor(1, 'AMD A8', 253, 3),  
 microprocessor(2, 'AMD Ryzen 5', 222, 2),  
 microprocessor(3, 'Intel Core i5', 434, 5),  
 microprocessor(4, 'AMD Athlon', 370, 2),  
 microprocessor(5, 'Intel Core i7', 556, 3),  
 ]  
 self.Microprocessor\_Computer = [  
 ComputerMicroprocessor(1, 3),  
 ComputerMicroprocessor(2, 2),  
 ComputerMicroprocessor(3, 5),  
 ComputerMicroprocessor(4, 2),  
 ComputerMicroprocessor(5, 3),  
  
 ComputerMicroprocessor(1, 2),  
 ComputerMicroprocessor(2, 4),  
 ComputerMicroprocessor(3, 1),  
 ComputerMicroprocessor(4, 4),  
 ComputerMicroprocessor(5, 1),  
 ]  
 self.one\_to\_many = [(m.title, m.sal, c.name)  
 for c in self.Computers  
 for m in self.Microprocessors  
 if m.comp\_id == c.id]  
 self.many\_to\_many\_temp = [(c.name, cm.comp\_id, cm.mic\_id)  
 for c in self.Computers  
 for cm in self.Microprocessor\_Computer  
 if c.id == cm.comp\_id]  
 self.many\_to\_many = [(c.title, c.sal, Computer\_name)  
 for Computer\_name, Computer\_id, microprocessor\_id in self.many\_to\_many\_temp  
 for c in self.Microprocessors if c.id == microprocessor\_id]  
  
 def test1(self):  
 result = res\_11(self.one\_to\_many)  
 desired = [('AMD Ryzen 5', 'ASUS'), ('AMD Athlon', 'ASUS'), ('AMD A8', 'Lenovo')]  
 self.assertEqual(result, desired)  
  
 def test2(self):  
 result = res\_12(self.one\_to\_many)  
 desired = [['ASUS', 222], ['AMD A8', 253], ['Intel Core i5', 434], ['Intel Core i7', 556]]  
 self.assertEqual(result, desired)  
 def test3(self):  
 result = res\_13(self.many\_to\_many)  
 desired = [('AMD A8', 'Lenovo'), ('AMD A8', 'Xiaomi'), ('AMD Athlon', 'ASUS'), ('AMD Athlon', 'Imac'), ('AMD Ryzen 5', 'MSI'), ('AMD Ryzen 5', 'ASUS'), ('AMD Ryzen 5', 'Imac'), ('Intel Core i5', 'MSI'), ('Intel Core i5', 'Xiaomi'), ('Intel Core i7', 'Lenovo')]  
 self.assertEqual(result, desired)

Результаты работы программы:

