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:

студент группы ИУ5-31Б

Климов Никита

Сергеевич

Подпись:_____

Дата:_____

Проверил:

преподаватель каф. ИУ5

Гапанюк Юрий

Евгеньевич

Подпись:_____

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 B2')
```

```
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: