Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль №2

«Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-33Б
Комаров Дмитрий

Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись и дата:

Москва, 2022 г.

Задача.

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).

1. **Рефакторинг текста :**

```
# используется для сортировки
from operator import itemgetter



class Microprocessor:
    """Микропроцессор"""
    def __init__(self, id, name, threads, comp_id):
        self.id = id
        self.name = name
        self.threads = threads
        self.comp_id = comp_id

class Computer:
    """Компьютер"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class MicropComp:
    """
    'Сотрудники отдела' для реализации
    связи многие-ко-многим
    """
    def __init__(self, comp_id, proc_id):
        self.comp_id = comp_id
        self.proc_id = proc_id

# Компьютеры
computers = [
    Computer(1, 'MacBook'),
    Computer(2, 'ThinkPad'),
```

```
    Computer(3, 'MateBookX'),

    Computer(11, 'Latitude'),
    Computer(22, 'ThinkPad'),
    Computer(33, 'Inspiron'),
]

# Микропроцессоры
microprocessor = [
    Microprocessor(1, 'Intel Core i5-12500k', 4, 1),
    Microprocessor(2, 'Intel Core i9-9000', 16, 2),
    Microprocessor(3, 'AMD Ryzen 5', 4, 3),
    Microprocessor(4, 'AMD Ryzen 3', 32, 3),
    Microprocessor(5, 'AMD Ryzen 8', 12, 1),
]

microp_comp = [
    MicropComp(1,1),
    MicropComp(2,2),
    MicropComp(3,3),
    MicropComp(3,4),
    MicropComp(3,5),
    MicropComp(11,1),
    MicropComp(22,2),
    MicropComp(33,3),
    MicropComp(33,4),
    MicropComp(33,5),
]

def task1(computers, microprocessor):

    for proc in microprocessor:
        res = list(filter(lambda i: i[0][0] == 'I',
one_to_many(computers, microprocessor)))
    return res

def task2(computers, microprocessor):
    res_min = []
    for c in computers:
        c_proc_threads = [(c_name, proc_threads) for proc_name,
proc_threads, c_name in one_to_many(computers, microprocessor) if
c_name == c.name]
    #Если в компьютере есть процессор
```

```
        if len(c_proc_threads) > 0:
            res_min.append(min(c_proc_threads, key =
itemgetter(1)))
    res_2 = sorted(res_min, key = itemgetter(1))
    return res_2

def task3(computers, microp_comp, microprocessor):
    res3 = sorted(many_to_many(computers, microp_comp,
microprocessor), key = itemgetter(2))
    return res3

def one_to_many(computers, microprocessor):
    return [(proc.name, proc.threads, c.name)
        for c in computers
        for proc in microprocessor
        if proc.comp_id==c.id]


def many_to_many_temp(computers, microp_comp):
    return [(c.name, procc.comp_id, procc.proc_id)
        for c in computers
        for procc in microp_comp
        if c.id == procc.comp_id]

def many_to_many(computers, microp_comp, microprocessor):
    return [(proc.name, proc.threads, procc_name)
        for procc_name, c_id, proc_id in
many_to_many_temp(computers, microp_comp)
        for proc in microprocessor if proc.id==proc_id]

def main():
    """Основная функция"""

    # Соединение данных многие-ко-многим
    print('Заданик 1')
    task1(computers, microprocessor)
    print('\nЗадание 2')
    task2(computers, microprocessor)
    print("\nЗадание 3")
    task3(computers, microp_comp, microprocessor)

if __name__ == '__main__':
    main()
```

2. Тест программы

```
import unittest
import rk1

class Test_field(unittest.TestCase):
    def setUp(self):
        self.test1 = [('Intel Core i5-12500k', 4, 'MacBook'),
('Intel Core i9-9000', 16, 'ThinkPad')]
        self.test2 = [('MacBook', 4), ('MateBookX', 4),
('ThinkPad', 16), ('ThinkPad', 16)]
        self.test3 = [('AMD Ryzen 5', 4, 'Inspiron'), ('AMD Ryzen
3', 32, 'Inspiron'), ('AMD Ryzen 8', 12, 'Inspiron'), ('Intel
Core i5-12500k', 4, 'Latitude'), ('Intel Core i5-12500k', 4,
'MacBook'), ('AMD Ryzen 5', 4, 'MateBookX'), ('AMD Ryzen 3', 32,
'MateBookX'), ('AMD Ryzen 8', 12, 'MateBookX'), ('Intel Core
i9-9000', 16, 'ThinkPad'), ('Intel Core i9-9000', 16,
'ThinkPad')]

    def test1_rk(self):
        self.assertEqual(rk1.task1(rk1.computers,
rk1.microprocessor), self.test1)

    def test2_rk(self):
        self.assertEqual(rk1.task2(rk1.computers,
rk1.microprocessor), self.test2)

    def test3_rk(self):
        self.assertEqual(rk1.task3(rk1.computers,
rk1.microp_comp, rk1.microprocessor), self.test3)

if __name__ == '__main__':
    unittest.main()
```

Результат:

```
userdead@DESKTOP-439N993:/mnt/c/Users/dimka/BKIT_Labs/PK2$ python3 test_rk2.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```