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»

Рубежный контроль №2

Выполнил:
студент группы ИУ5-33Б:
Лебедева С.К.

Руководитель:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2022 г.

Постановка задачи:

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:

PythonApplication6.py (изменённый код РК1)

```
from operator import itemgetter

class OperatingSystem:
    """Операционная система"""
    def __init__(self, id, name, version, bit_depth, comp_id):
        self.id = id
        self.name = name
        self.version = version
        self.bit_depth = bit_depth
        self.comp_id = comp_id

class Computer:
    """Компьютер"""
    def __init__(self, id, processor):
        self.id = id
        self.processor = processor
        #self.manufacturer = manufacturer
class OSComp:
    """Операционные системы в компьютерах"""
    def __init__(self, comp_id, os_id):
        self.comp_id = comp_id
        self.os_id = os_id

#Компьютеры
Computers = [
    Computer(1, 'Intel Core i7-1265'),
    Computer(2, 'AMD Ryzen 5 7600'),
    Computer(3, 'Intel Core i5-1235'),
    #Для связи M<->M
    Computer(11, 'Intel Core i5-12500'),
    Computer(22, 'Intel Core i3-10300 '),
    Computer(33, 'AMD Ryzen 3 2500')
]

#Операционные системы
OperatingSystems = [
    OperatingSystem(1, 'Microsoft Windows', '8', 16, 1),
    OperatingSystem(2, 'Microsoft Windows', '11', 64, 2),
    OperatingSystem(3, 'Linux', 'Oracle', 16, 3),
```

```
    OperatingSystem(4, 'Linux', 'Miracle', 32, 3),
    OperatingSystem(5, 'macOS', 'Mac Ventura', 64, 1)
]

#Операционные системы в компьютерах
OSComps = [
    OSComp(1, 1),
    OSComp(2, 2),
    OSComp(2, 3),
    OSComp(3, 4),
    OSComp(3, 5),

    OSComp(11, 3),
    OSComp(11, 2),
    OSComp(22, 4),
    OSComp(22, 1),
    OSComp(33, 5),
]


one_to_many = [(os.name, os.version, os.bit_depth, c.processor)
    for c in Computers
    for os in OperatingSystems
    if os.comp_id == c.id]

    # Соединение данных M<->M
many_to_many_temp = [(c.processor, osc.comp_id, osc.os_id)
    for c in Computers
    for osc in OSComps
    if c.id == osc.comp_id]

many_to_many = [(os.name, os.version, os.bit_depth, osc_name)
    for osc_name, c_id, os_id in many_to_many_temp
    for os in OperatingSystems if os.id == os_id]


def task1(one_to_many):
    """
    «Компьютер» 1<->M «Операционная система».
    Вывести список всех ОС, начинающихся с буквы «L»
    и названия процессоров компьютеров, на которых
    они установлены.
    """
    for os in OperatingSystems:
        res_1 = list(filter(lambda i: i[0][0]  == 'L', one_to_many))
    return res_1

def task2(one_to_many):
    """
    «Компьютер» 1<->M «Операционная система».
    Вывести список названий процессоров компьютеров
    с минимальной разрядностью ОС этих компьютеров,
    отсортированный по минимальной разрядности ОС.
    """
```

```
    res_min_unsorted = []

    #Перебираем все компьютеры
    for c in Computers:
        c_os_bit_depth = [(c_processor, os_bit_depth) for os_name, os_version,
os_bit_depth, c_processor in one_to_many if c_processor == c.processor]
        #Если на компьютере есть ОС
        if len(c_os_bit_depth) > 0:
            res_min_unsorted.append(min(c_os_bit_depth, key = itemgetter(1)))
    res_2 = sorted(res_min_unsorted, key = itemgetter(1))
    return res_2

def task3(many_to_many):
    """
    «Компьютер» M<->M «Операционная система».
    Вывести список всех связанных операционных систем
    и компьютеров, отсортированный по разрядности ОС,
    сортировка по компьютерам произвольная.
    """
    res_3 = sorted(many_to_many, key = itemgetter(2))
    return res_3

def main():
    """Основная функция"""
    print('Задание B1')
    r1 = task1(one_to_many)
    print(r1)
    print('\nЗадание B2')
    r2 = task2(one_to_many)
    for r in r2:
        print(r)
    print('\nЗадание B3')
    r3 = task3(many_to_many)
    for r in r3:
        print(r)


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

test.py

```
import unittest
from PythonApplication6 import *

class TestStringMethods(unittest.TestCase):
    def setUp(self):
        self.one_to_many = [(os.name, os.version, os.bit_depth, c.processor)
        for c in Computers
        for os in OperatingSystems
        if os.comp_id == c.id]
        self.many_to_many_temp = [(c.processor, osc.comp_id, osc.os_id)
```

```
        for c in Computers
        for osc in OSComps
        if c.id == osc.comp_id]
        self.many_to_many = [(os.name, os.version, os.bit_depth, osc_name)
        for osc_name, c_id, os_id in many_to_many_temp
        for os in OperatingSystems if os.id == os_id]

    def test_task1(self):
        answer = [('Linux', 'Oracle', 16, 'Intel Core i5-1235'),
                  ('Linux', 'Miracle', 32, 'Intel Core i5-1235')]
        res = task1(one_to_many)
        self.assertEqual(res, answer)

    def test_task2(self):
        answer = [('Intel Core i7-1265', 16),
                  ('Intel Core i5-1235', 16),
                  ('AMD Ryzen 5 7600', 64)]
        res = task2(one_to_many)
        self.assertEqual(res, answer)

    def test_task3(self):
        answer = [('Microsoft Windows', '8', 16, 'Intel Core i7-1265'),
                ('Linux', 'Oracle', 16, 'AMD Ryzen 5 7600'),
                ('Linux', 'Oracle', 16, 'Intel Core i5-12500'),
                ('Microsoft Windows', '8', 16, 'Intel Core i3-10300 '),
                ('Linux', 'Miracle', 32, 'Intel Core i5-1235'),
                ('Linux', 'Miracle', 32, 'Intel Core i3-10300 '),
                ('Microsoft Windows', '11', 64, 'AMD Ryzen 5 7600'),
                ('macOS', 'Mac Ventura', 64, 'Intel Core i5-1235'),
                ('Microsoft Windows', '11', 64, 'Intel Core i5-12500'),
                ('macOS', 'Mac Ventura', 64, 'AMD Ryzen 3 2500')]
        res = task3(many_to_many)
        self.assertEqual(res, answer)

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

Результаты выполнения: