ИУ5-32Б,

Колесников Никита

Рубежный контроль 2

Условия:

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

```
# используется для сортировки
from operator import itemgetter


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


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


class OSComp:
    """
    'Операционная система компьютера' для реализации
    связи многие-ко-многим
    """
```

```
    def __init__(self, os_id, comp_id):
        self.os_id = os_id
        self.comp_id = comp_id

# Компьютеры
comps = [
    Comp(1, 'Apple MacBook Air'),
    Comp(2, 'Dell XPS 13'),
    Comp(3, 'Acer Swift 3'),
]
# Операционные системы
OSs = [
    OS(1, 'Windows', 11, 1),
    OS(2, 'macOS', 14, 2),
    OS(3, 'Android', 14, 3),
    OS(4, 'iOS', 17, 3),
    OS(5, 'Linux', 23.10, 3),
]
comps_OSs = [
    OSComp(1,1),
    OSComp(2,2),
    OSComp(5,3),
    OSComp(3,1),
    OSComp(4,3),
]

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

    # Соединение данных один-ко-многим
    one_to_many = [(os.name, os.version, c.name)
                   for c in comps
```

```
            for os in  OSs
            if os.comp_id == c.id]

# Соединение данных многие-ко-многим
many_to_many_temp = [(c.name, cos.comp_id, cos.os_id)
                for c in comps
                for cos in comps_OSs
                if c.id == cos.comp_id]

many_to_many = [(os.name, os.version, comp_name)
            for comp_name, comp_id, os_id in many_to_many_temp
            for os in OSs if os.id == os_id]

print('Задание Б1')
res_1 = sorted(one_to_many, key=itemgetter(2))
print(res_1)

print('\nЗадание Б2')
res_2 = []
temp_dict=dict()
for i in one_to_many:
   if i[2] in temp_dict:
      temp_dict[i[2]]+=1
   else:
      temp_dict[i[2]]=1
for i in temp_dict.keys():
   res_2.append((i,temp_dict[i]))
res_2.sort(key=itemgetter(1), reverse=True)
print(res_2)

print('\nЗадание Б3')
res_3=[]
```

```
    for i in many_to_many:
        if i[0][-1]=='S':
            res_3.append((i[0], i[2]))
    print(res_3)


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

unit_tests.py

```
import main
from operator import itemgetter
import unittest


class TestMainMethods(unittest.TestCase):
    def test_first_task_method(self):
     test_list = [('second', 'second', 'second'), ('first', 'first', 'first'), ('third', 'third', 'third')]
     result = main.first_task(test_list)
     reference = sorted(test_list, key=itemgetter(0))
     self.assertEqual(result, reference)
    def test_second_task_method(self):
        test_list = [('A. Mercer', 120000, 'Resource department'), ('R. Gosling', 110000, 'Archive
        department'),
        ('E. Yeger', 80000, 'Resource department'), ('C. Nolan', 130000, 'Logistic department')]
        result = main.second_task(test_list)
        reference = [('Resource department', 2), ('Archive department', 1), ('Logistic department', 1)]
        self.assertEqual(result, reference)
    def test_third_method(self):
        test_list = [('A. Mercer', 120000, 'Resource department'), ('R. Gosling', 110000, 'Archive
        department'),
        ('E. Yeger', 80000, 'Resource department'), ('C. Nolan', 130000, 'Logistic department')]
        result = main.third_task(test_list, 'r')
```

```
reference = [('A. Mercer', 'Resource department'), ('E. Yeger', 'Resource department')]
self.assertEqual(result, reference)
```