ИУ5-34Б Матвеев Илья

**Отчет РК1 по дисциплине  
«Парадигмы и конструкции языков программирования»**

**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**

**main.py**

from operator import itemgetter  
  
  
class Computer:  
 *"""Компьютер"""* def \_\_init\_\_(self, id, name, price, os\_id):  
 self.id = id  
 self.name = name  
 self.price = price  
 self.os\_id = os\_id  
  
  
class OperatingSystem:  
 *"""Операционная система"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class ComputersOS:  
 *"""  
 'Компьютеры с ОС' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, os\_id, comp\_id):  
 self.os\_id = os\_id  
 self.comp\_id = comp\_id  
  
  
def get\_one\_to\_many(oses, computers):  
 *"""Соединение данных один-ко-многим"""* return [(c.name, c.price, os.name)  
 for os in oses  
 for c in computers  
 if c.os\_id == os.id]  
  
  
def get\_many\_to\_many(oses, computers, computers\_os):  
 *"""Соединение данных многие-ко-многим"""* many\_to\_many\_temp = [(os.name, co.os\_id, co.comp\_id)  
 for os in oses  
 for co in computers\_os  
 if os.id == co.os\_id]  
  
 return [(c.name, c.price, os\_name)  
 for os\_name, os\_id, comp\_id in many\_to\_many\_temp  
 for c in computers if c.id == comp\_id]  
  
  
def task\_1(oses, computers):  
 *"""Задание 1"""* selected\_os = [os for os in oses if os.name.startswith('W')]  
 return {os.name: [comp.name for comp in computers if comp.os\_id == os.id] for os in selected\_os}  
  
  
def task\_2(oses, one\_to\_many):  
 *"""Задание 2"""* res\_2\_unsorted = []  
 for os in oses:  
 os\_comps = list(filter(lambda i: i[2] == os.name, one\_to\_many))  
 if os\_comps:  
 os\_prices = [price for \_, price, \_ in os\_comps]  
 os\_max\_price = max(os\_prices)  
 res\_2\_unsorted.append((os.name, os\_max\_price))  
  
 return sorted(res\_2\_unsorted, key=itemgetter(1), reverse=True)  
  
  
def task\_3(oses, computers, computers\_os):  
 *"""Задание 3"""* result\_3 = {os.name: [comp.name for relation in computers\_os for comp in computers if  
 relation.os\_id == os.id and relation.comp\_id == comp.id]  
 for os in oses}  
  
 return {os: result\_3[os] for os in sorted(result\_3.keys())}  
  
  
def main():  
 oses = [  
 OperatingSystem(1, 'Windows'),  
 OperatingSystem(2, 'Linux'),  
 OperatingSystem(3, 'macOS'),  
 OperatingSystem(11, 'Windows Server'),  
 OperatingSystem(22, 'Ubuntu'),  
 ]  
  
 computers = [  
 Computer(1, 'PC1', 1000, 1),  
 Computer(2, 'PC2', 1200, 1),  
 Computer(3, 'PC3', 900, 11),  
 Computer(4, 'PC4', 1500, 3),  
 Computer(5, 'PC5', 1100, 2),  
 ]  
  
 computers\_os = [  
 ComputersOS(1, 1),  
 ComputersOS(1, 2),  
 ComputersOS(2, 3),  
 ComputersOS(2, 5),  
 ComputersOS(3, 4),  
 ComputersOS(11, 1),  
 ComputersOS(22, 3),  
 ]  
  
 one\_to\_many = get\_one\_to\_many(oses, computers)  
 many\_to\_many = get\_many\_to\_many(oses, computers, computers\_os)  
  
 print("\nTask 1")  
 print(task\_1(oses, computers))  
  
 print("\nTask 2")  
 print(task\_2(oses, one\_to\_many))  
  
 print("\nTask 3")  
 print(task\_3(oses, computers, computers\_os))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**test\_computers\_os.py**

import unittest  
from main import Computer, OperatingSystem, ComputersOS, get\_one\_to\_many, get\_many\_to\_many, task\_1, task\_2, task\_3  
  
  
class TestComputers(unittest.TestCase):  
  
 def setUp(self):  
 self.oses = [  
 OperatingSystem(1, 'Windows'),  
 OperatingSystem(2, 'Linux'),  
 OperatingSystem(3, 'macOS'),  
 OperatingSystem(11, 'Windows Server'),  
 OperatingSystem(22, 'Ubuntu'),  
 ]  
  
 self.computers = [  
 Computer(1, 'PC1', 1000, 1),  
 Computer(2, 'PC2', 1200, 1),  
 Computer(3, 'PC3', 900, 11),  
 Computer(4, 'PC4', 1500, 3),  
 Computer(5, 'PC5', 1100, 2),  
 ]  
  
 self.computers\_os = [  
 ComputersOS(1, 1),  
 ComputersOS(1, 2),  
 ComputersOS(2, 3),  
 ComputersOS(2, 5),  
 ComputersOS(3, 4),  
 ComputersOS(11, 1),  
 ComputersOS(22, 3),  
 ]  
  
 def test\_task\_1(self):  
 result = task\_1(self.oses, self.computers)  
 expected = {  
 'Windows': ['PC1', 'PC2'],  
 'Windows Server': ['PC3']  
 }  
 self.assertEqual(result, expected)  
  
 def test\_task\_2(self):  
 one\_to\_many = get\_one\_to\_many(self.oses, self.computers)  
 result = task\_2(self.oses, one\_to\_many)  
 expected = [('macOS', 1500), ('Windows', 1200), ('Linux', 1100), ('Windows Server', 900)]  
 self.assertEqual(result, expected)  
  
 def test\_task\_3(self):  
 result = task\_3(self.oses, self.computers, self.computers\_os)  
 expected = {  
 'Linux': ['PC3', 'PC5'],  
 'Ubuntu': ['PC3'],  
 'Windows': ['PC1', 'PC2'],  
 'Windows Server': ['PC1'],  
 'macOS': ['PC4']  
 }  
 self.assertEqual(result, expected)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**Анализ результатов**

