ИУ5-32Б,

Копылов Николай

***Рубежный контроль 2***

***Вариант 10***

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.

from operator import itemgetter

import math

class Browser:

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class Computer:

def \_\_init\_\_(self, id, name, speed, browser\_id):

self.id = id

self.name = name

self.speed = speed

self.browser\_id = browser\_id

class BrowserComputer:

def \_\_init\_\_(self, browser\_id, comp\_id):

self.browser\_id = browser\_id

self.comp\_id = comp\_id

def first\_task(comp\_list):

return sorted(comp\_list, key=itemgetter(0))

def second\_task(comp\_list):

temp\_dict = {}

for i in comp\_list:

if i[2] in temp\_dict:

temp\_dict[i[2]] += 1

else:

temp\_dict[i[2]] = 1

res\_2 = [(i, temp\_dict[i]) for i in temp\_dict.keys()]

res\_2.sort(key=itemgetter(1), reverse=True)

return res\_2

def third\_task(comp\_list, end\_ch):

return [(i[0], i[2]) for i in comp\_list if i[0].endswith(end\_ch)]

import main

from operator import itemgetter

import unittest

class TestMainMethods(unittest.TestCase):

def setUp(self):

# Инициализация данных

self.browsers = [

main.Browser(1, "Chrome"),

main.Browser(2, "Firefox"),

main.Browser(3, "Safari"),

]

self.computers = [

main.Computer(1, "Dell", 2.5, 1),

main.Computer(2, "HP", 3.0, 2),

main.Computer(3, "MacBook", 2.7, 3),

main.Computer(4, "Lenovo", 2.3, 3),

main.Computer(5, "Acer", 2.9, 1),

main.Computer(6, "Asus", 3.2, 1),

]

self.browser\_computers = [

main.BrowserComputer(1, 1),

main.BrowserComputer(2, 2),

main.BrowserComputer(3, 3),

main.BrowserComputer(3, 4),

main.BrowserComputer(1, 5),

]

# Преобразования для тестирования

self.one\_to\_many = [

('Dell', 2.5, 'Chrome'),

('HP', 3.0, 'Firefox'),

('MacBook', 2.7, 'Safari'),

('Lenovo', 2.3, 'Safari'),

('Acer', 2.9, 'Chrome'),

('Asus', 3.2, 'Chrome'),

]

self.many\_to\_many = [

('Chrome', 1, 1),

('Firefox', 2, 2),

('Safari', 3, 3),

('Safari', 3, 4),

('Chrome', 1, 5),

]

def test\_first\_task\_method(self):

# Проверка первого задания

result = main.first\_task(self.one\_to\_many)

reference = sorted(self.one\_to\_many, key=itemgetter(0))

self.assertEqual(result, reference)

def test\_second\_task\_method(self):

# Проверка второго задания

result = main.second\_task(self.one\_to\_many)

reference = [('Chrome', 3), ('Safari', 2), ('Firefox', 1)]

self.assertEqual(result, reference)

def test\_third\_task\_method\_with\_k(self):

# Проверка третьего задания для символа 'k', который встречается в "MacBook"

result = main.third\_task(self.one\_to\_many, 'k')

reference = [('MacBook', 'Safari')] # Ожидаем, что только MacBook имеет имя, заканчивающееся на 'k'

self.assertEqual(sorted(result), sorted(reference)) # Сравниваем отсортированные списки

def test\_third\_task\_method\_with\_x(self):

# Проверка третьего задания для символа 'x', которого нет в именах

result = main.third\_task(self.one\_to\_many, 'x')

reference = [] # Ожидаем пустой список

self.assertEqual(sorted(result), sorted(reference)) # Сравниваем отсортированные списки

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

Результаты тестирования:  
