Группа: ИУ5-32Б
Студент: Трундаев Дмитрий

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:

```
# rk2.py
from operator import itemgetter

class Provider:
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Part:
    def __init__(self, id, name, cost, provider_id):
        self.id = id
        self.name = name
        self.cost = cost
        self.provider_id = provider_id


class Provided_parts:
    def __init__(self, provider_id, part_id):
        self.provider_id = provider_id
        self.part_id = part_id

providers = {
    Provider(1, "Nvidia"),
    Provider(2, "Asus"),
    Provider(3, "Corsair"),
    Provider(4, "Kingston"),
    Provider(5, "AMD")
}

parts = {
    Part(1, "motherboard", 200, 2),
    Part(2, "graphics card", 374, 1),
    Part(3, "CPU", 125, 5),
    Part(4, "memory card", 40, 4),
    Part(5, "RAM", 75, 4),
    Part(6, "PSU", 66, 3)
}

pr_parts = {
    Provided_parts(2, 1),
    Provided_parts(3, 5),
    Provided_parts(2, 2),
    Provided_parts(5, 4),
    Provided_parts(5, 3)
}

def first_task(pr_list):
    result = sorted(pr_list, key=itemgetter(0))
    return result

def second_task(pr_list):
    result = []
    temp_dict = dict()
    for i in pr_list:
        if i[2] in temp_dict:
            temp_dict[i[2]] += 1
        else:
            temp_dict[i[2]] = 1

    for i in temp_dict.keys():
        result.append((i, temp_dict[i]))

    result.sort(key=itemgetter(1), reverse=True)
    return result

def third_task(pr_list, max_cost):
    result = [(i[0], i[2]) for i in pr_list if i[1] < max_cost]
    return result

def main():
    one_to_many = [(part.name, part.cost, provider.name)
                   for provider in providers
                   for part in parts
                   if part.provider_id == provider.id]

    many_to_many_temp = [(provider.name, pp.provider_id, pp.part_id)
                         for provider in providers
                         for pp in pr_parts
                         if pp.provider_id == provider.id]

    many_to_many = [(part.name, part.cost, provider_name)
                   for provider_name, provider_id, part_id in many_to_many_temp
                   for part in parts if part.id == part_id]

    print('Задание Б1')
    print(first_task(one_to_many))

    print("\n 'Задание Б2")
    print(second_task(one_to_many))

    print("\n 'Задание Б3")
    print(third_task(many_to_many, 200))

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

```
# unit_tests.py

import unittest
import rk2

class TestTaskFunctions(unittest.TestCase):

   def test_task1(self):
      in_data = (('RAM', 75, 'Kingston'), ('memory card', 40, 'Kingston'), ('CPU', 125, 'AMD'), ('motherboard', 
200, 'Asus'), ('PSU', 66, 'Corsair'), ('graphics card', 374, 'Nvidia'))
      correct_out = [('CPU', 125, 'AMD'), ('PSU', 66, 'Corsair'), ('RAM', 75, 'Kingston'), ('graphics card', 374, 
'Nvidia'), ('memory card', 40, 'Kingston'), ('motherboard', 200, 'Asus')]
      out = rk2.first_task(in_data)
      self.assertEqual(out, correct_out)

   def test_task2(self):
      in_data = (('RAM', 75, 'Kingston'), ('memory card', 40, 'Kingston'), ('CPU', 125, 'AMD'), ('motherboard', 
200, 'Asus'), ('PSU', 66, 'Corsair'), ('graphics card', 374, 'Nvidia'))
      correct_out = [('Kingston', 2), ('AMD', 1), ('Asus', 1), ('Corsair', 1), ('Nvidia', 1)]
      out = rk2.second_task(in_data)
      self.assertEqual(out, correct_out)

   def test_task3(self):
      in_data = (('memory card', 40, 'AMD'), ('CPU', 125, 'AMD'), ('motherboard', 200, 'Asus'), ('graphics card', 
374, 'Asus'), ('RAM', 75, 'Corsair'))
      correct_out = [('memory card', 'AMD'), ('CPU', 'AMD'), ('RAM', 'Corsair')]
      out = rk2.third_task(in_data, 200)
      self.assertEqual(out, correct_out)


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

Результаты выполнения:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```