**Московский государственный технический университет им. Н.Э. Баумана**

**Факультет ИУ Кафедра ИУ5**

**Курс «Основы информатики» Отчет по Рубежному контролю №2**

Выполнил студент группы ИУ5-33Б: Козлов А. А.

Подпись и дата:

Проверил преподаватель каф.: Гапанюк Ю. Е.

Подпись и дата:

Москва, 2024 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**

import unittest

from operator import itemgetter

class Computer():

    def \_\_init\_\_(self, id, name, cost, class\_id):

        self.id = id

        self.name = name

        self.cost = cost

        self.class\_id = class\_id

class DisClass():

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

class CompDC():

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

        self.class\_id = class\_id

        self.comp\_id = comp\_id

comps = [

    Computer(1, 'A123', 100, 1),

    Computer(2, 'B123', 200, 2),

    Computer(3, 'A233', 150, 3),

    Computer(4, 'D123', 200, 4),

    Computer(5, 'E123', 300, 5),]

dis\_classes = [

    DisClass(1, '312л'),

    DisClass(2, '313л'),

    DisClass(3, '314л'),

    DisClass(4, '315л'),

    DisClass(5, '316л'),]

comp\_dc = [

    CompDC(1, 1),

    CompDC(2, 2),

    CompDC(3, 3),

    CompDC(3, 4),

    CompDC(3, 5),

    CompDC(4, 1),

    CompDC(4, 2),

    CompDC(4, 3),

    CompDC(5, 4),

    CompDC(5, 5),]

one\_to\_many = [(c.name, c.cost, dc.name)

                for dc in dis\_classes

                for c in comps

                if c.class\_id == dc.id]

many\_to\_many\_temp = [(dc.name, co.class\_id, co.comp\_id)

                for dc in dis\_classes

                for co in comp\_dc

                if dc.id == co.class\_id]

many\_to\_many = [(c.name, c.cost, class\_name)

                for class\_name, class\_id, comp\_id in many\_to\_many\_temp

                for c in comps if c.id == comp\_id]

'''

print('Задание B1')

«дисплейный класс» и «компьютер» связаны соотношением один-ко-многим. Выведите список всех компьютеров, у которых название начинается с буквы «А», и названия их дисплейных классов.

'''

def firstEx():

    resultB1 = sorted(x for x in one\_to\_many if x[0].startswith('A'))

    return[f'Название: {i[0]}, класс: {i[2]}' for i in resultB1]

'''

print('\nЗадание B2')

«дисплейный класс» и «компьютер» связаны соотношением один-ко-многим. Выведите список классов с минимальной ценой за компьютер в каждом дисплейном классе, отсортированный по минимальной цене.

'''

def secondEx():

    resultB2 = {}

    for class\_name in set(x[2] for x in one\_to\_many):

        min\_cost = min(x[1] for x in one\_to\_many if x[2] == class\_name)

        resultB2[class\_name] =  min\_cost

    resultB2 = sorted(resultB2.items(), key=itemgetter(1))

    return[f'Класс: {i[0]}, минимальная стоимость: {i[1]}' for i in resultB2]

'''

print('\nЗадание B3')

«дисплейный класс» и «компьютер» связаны соотношением многие-ко-многим. Выведите список всех связанных компьютеров и дисплейных классов, отсортированный по компьютерам, сортировка по дисплейным классам произвольная.

'''

def thirdEx():

    resultB3 = sorted(many\_to\_many, key=itemgetter(0))

    return[f'Название: {i[0]}, класс: {i[2]}' for i in resultB3]

class TestMyFunctions(unittest.TestCase):

    def test\_firstEx(self):

        expected\_result = ['Название: A123, класс: 312л', 'Название: A233, класс: 314л']

        self.assertEqual(firstEx(), expected\_result)

    def test\_secondEx(self):

        expected\_result = ['Класс: 312л, минимальная стоимость: 100',

            'Класс: 314л, минимальная стоимость: 150',

            'Класс: 315л, минимальная стоимость: 200',

            'Класс: 313л, минимальная стоимость: 200',

            'Класс: 316л, минимальная стоимость: 300']

        self.assertEqual(secondEx(), expected\_result)

    def test\_thirdEx(self):

        expected\_result = ['Название: A123, класс: 312л',

            'Название: A123, класс: 315л',

            'Название: A233, класс: 314л',

            'Название: A233, класс: 315л',

            'Название: B123, класс: 313л',

            'Название: B123, класс: 315л',

            'Название: D123, класс: 314л',

            'Название: D123, класс: 316л',

            'Название: E123, класс: 314л',

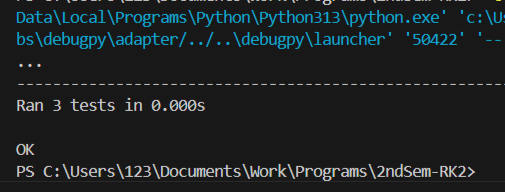
            'Название: E123, класс: 316л']

        self.assertEqual(thirdEx(), expected\_result)

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

    unittest.main()

**Анализ результатов**

****