Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего **образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль № 2

по дисциплине «Базовые компоненты интернет-технологий»

16 вариант

Выполнил:
студент группы ИУ5-34Б
Сергеев Н.С

Проверил:
Гапанюк Ю.Е.

2021 г.

Полученное задание:

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:

Файл rk2.py

```
from operator import itemgetter

class CD:
    """CD-диск"""

    def __init__(self, id, name_cd, cost, cd_lib_id):
        self.id = id
        self.name_cd = name_cd # название
        self.cost = cost # стоимость
        self.cd_lib_id = cd_lib_id

class CD_Library:
    """CD-Библиотека"""

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

class CD_v_library:
    """Диск в библиотеке для реализации связи многие-ко-многим"""

    def __init__(self, cd_id, library_id):
        self.cd_id = cd_id
        self.library_id = library_id

# библиотека
library = [
    CD_Library(1, 'Музыкальные клипы'),
    CD_Library(2, 'Игры'),
    CD_Library(3, 'Музыка'),
    CD_Library(4, 'Аквапарк'),
    CD_Library(5, 'Фильмы'),
    CD_Library(6, 'Мультфильмы'),
]

# CD
CDs = [
    CD(1, 'Альбом группы Кино', 500, 3),
    CD(2, 'Аватар', 2000, 5),
    CD(3, 'Том и Джерри', 2500, 6),
    CD(4, 'Видео с аквапарка', 0, 4),
    CD(5, 'Музыкальные клипы 90-х', 3000, 1),
    CD(6, 'Соник', 200, 2),
```

```
]

CD_and_Library = [
    CD_v_library(3, 1),
    CD_v_library(5, 2),
    CD_v_library(6, 3),
    CD_v_library(4, 4),
    CD_v_library(1, 5),
    CD_v_library(2, 6),
    CD_v_library(6, 6),
    CD_v_library(2, 3),
    CD_v_library(3, 5),
    CD_v_library(1, 1),
]


# Соединение данных один-ко-многим
one_to_many = [(k.name_cd, k.cost, m.name)
               for m in library
               for k in CDs
               if k.cd_lib_id == m.id]

    # Соединение данных многие-ко-многим
many_to_many_temp = [(m.name, km.cd_id, km.library_id)
                     for m in library
                     for km in CD_and_Library
                     if m.id == km.cd_id]

many_to_many = [(k.name_cd, k.cost, CD_Library_name)
                for CD_Library_name, library_id, cd_id in many_to_many_temp
                for k in CDs if k.id == cd_id]

def number1(one_to_many):
    result1 = []
    for name_cd, _, CD_Library_name in one_to_many:
        if 'A' in CD_Library_name[0]:
            result1.append((name_cd, CD_Library_name))
    return result1


def number2(one_to_many):
    res_12_unsorted = []
    # Перебираем все библиотеки
    for d in library :
        # Список CD в библиотеке
        d_libr = list(filter(lambda i: i[2] == d.name, one_to_many))
        # Если библиотека не пустая
        if len(d_libr) > 0:
            res_12_unsorted.append((d.name, max(d_libr, key=lambda x: x[1])[1]))
    # Сортировка по максимальной стоимости
    result2 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return result2

def number3(many_to_many):
```

```
    res13 = []
    for name_cd, cost, CD_Library_name in many_to_many:
        res13.append((name_cd, CD_Library_name))
    result3 = list(sorted(res13, key=itemgetter(1)))
    return result3

if __name__ == '__main__':
    print('Задание Г1', number1(one_to_many), sep = '\n')
    print('Задание Г2', number2(one_to_many), sep = '\n')
    print('Задание Г3', number3(many_to_many), sep = '\n')
```

Файл test.py

```
import unittest
from rk2 import library, CDs, CD_and_Library, number1, number2, number3

one_to_many_test = [(k.name_cd, k.cost, m.name)
                for m in library
                for k in CDs
                if k.cd_lib_id == m.id]

many_to_many_temp = [(m.name, km.cd_id, km.library_id)
                    for m in library
                    for km in CD_and_Library
                    if m.id == km.cd_id]

many_to_many_test = [(k.name_cd, k.cost, CD_Library_name)
                for CD_Library_name, library_id, cd_id in many_to_many_temp
                for k in CDs if k.id == cd_id]

class Tests(unittest.TestCase):

    def test1(self):
        self.assertEqual(number1(one_to_many_test), [('Видео с аквапарка', 'Аквапарк')])

    def test2(self):
        self.assertEqual(number2(one_to_many_test), [('Музыкальные клипы', 3000),
        ('Мультфильмы', 2500), ('Фильмы', 2000), ('Музыка', 500), ('Игры', 200),
('Аквапарк', 0)])

    def test3(self):
        self.assertEqual(number3(many_to_many_test), [('Видео с аквапарка', 'Аквапарк'),
        ('Соник', 'Игры'), ('Том и Джерри', 'Игры'), ('Альбом группы Кино', 'Музыка'),
        ('Музыкальные клипы 90-х', 'Музыка'), ('Музыкальные клипы 90-х', 'Музыкальные
клипы'),
        ('Альбом группы Кино', 'Музыкальные клипы'), ('Том и Джерри', 'Мультфильмы'),
        ('Соник', 'Мультфильмы'), ('Аватар', 'Фильмы')])

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

Результат выполнения

```
PS D:\лабы\рк2> & D:/Python/python.exe d:/лабы/рк2/test.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
PS D:\лабы\рк2>
```