Шараев Вячеслав

ИУ5-34Б

Отчет по РК2 по дисциплине

“Парадигмы и конструкции языков программирования”

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:

Рефакторинг main (main_test.py)

```
from operator import itemgetter


class CD_disk:
    def init(self, id, name, size, library_id):
        self.id = id
        self.name = name
        self.size = size
        self.library_id = library_id


class CD_library:
    def init(self, id, name):
        self.id = id
        self.name = name


class Disks_in_libraries:
    def init(self, library_id, disk_id):
        self.library_id = library_id
        self.disk_id = disk_id



disks = [
    CD_disk(1, "The Silence of the Lambs", 512, 1),
    CD_disk(2, "Men in Black", 1024, 2),
    CD_disk(3, "Home Alone", 256, 2),
    CD_disk(4, "Avatar", 2048, 3),
    CD_disk(5, "Interstellar", 4096, 3)

]

libraries = [
    CD_library(1, "Thriller"),
    CD_library(2, "Comedy"),
    CD_library(3, "Fiction"),
    CD_library(4, "Drama")
]

disks_in_libraries = [
    Disks_in_libraries(1, 1),
    Disks_in_libraries(2, 2),
    Disks_in_libraries(2, 3),
    Disks_in_libraries(3, 4),
    Disks_in_libraries(3, 5),

    Disks_in_libraries(4, 4)
]



one_to_many = [(d.name, d.size, l.name)
                   for l in libraries
```

```
                for d in disks
                if d.library_id == l.id]

many_to_many_temp = [(l.name, dl.library_id, dl.disk_id)
                    for l in libraries
                    for dl in disks_in_libraries
                    if l.id==dl.library_id]

many_to_many = [(d.name, d.size, lib_name)
    for lib_name, library_id, disk_id in many_to_many_temp
    for d in disks
    if d.id==disk_id]

def task1(one_to_many):
  return [item
      for item in one_to_many
      if item[2].startswith('C')]

def task2(one_to_many, libraries):
  res2_unsorted = []
  for l in libraries:
    lib_disks = list(filter(lambda i: i[2]==l.name, one_to_many))
    if len(lib_disks) > 0:
      d_sizes = [size for _,size,_ in lib_disks]
      l_max_sizes = max(d_sizes)
      res2_unsorted.append((l.name, l_max_sizes))
  return sorted(res2_unsorted, key=itemgetter(1), reverse=True)

def task3(many_to_many, libraries):
  res3 = {}
  for l in libraries:
    lib_disks = list(filter(lambda i: i[2] == l.name, many_to_many))
```

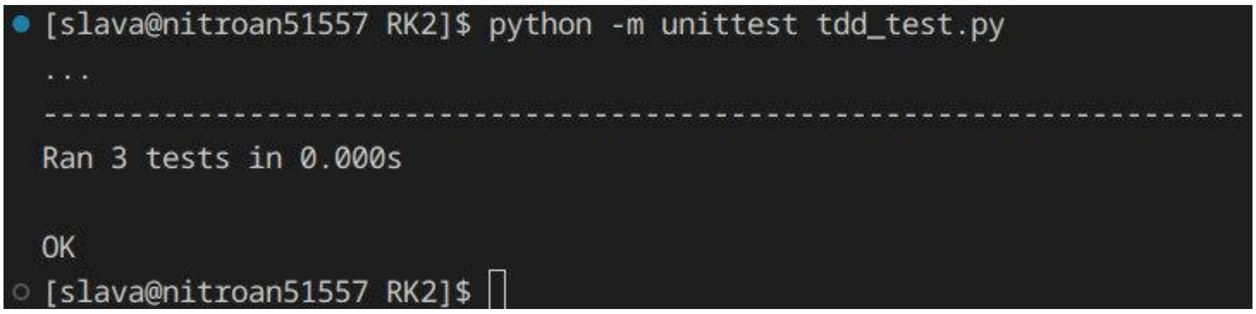
```
        l_disk_name = [name for name, _, _ in lib_disks]
        res3[l.name] = l_disk_name
    return res3

if name == "main":
    print("Задание Г1:", task1(one_to_many))
    print("Задание Г2:", task2(one_to_many, libraries))
    print("Задание Г3:", task3(many_to_many, libraries))
```

tdd_test.py

```
import unittest
from main_test import (
    task1,
    task2,
    task3,
    libraries
)

class TestCDLibrary(unittest.TestCase):
    def setUp(self):
        self.one_to_many = [
            ("The Silence of the Lambs", 512, "Thriller"),
            ("Men in Black", 1024, "Comedy"),
            ("Home Alone", 256, "Comedy"),
            ("Avatar", 2048, "Fiction"),
            ("Interstellar", 4096, "Fiction")
        ]

        self.many_to_many = [
            ("The Silence of the Lambs", 512, "Thriller"),
            ("Men in Black", 1024, "Comedy"),
            ("Home Alone", 256, "Comedy"),
            ("Avatar", 2048, "Fiction"),
            ("Interstellar", 4096, "Fiction"),
            ("Avatar", 2048, "Drama")
        ]

    def test_task1(self):
        result = task1(self.one_to_many)
        expected = [
            ("Men in Black", 1024, "Comedy"),
            ("Home Alone", 256, "Comedy")
        ]
        self.assertEqual(result, expected)

    def test_task2(self):
        result = task2(self.one_to_many, libraries)
        expected = [
            ("Fiction", 4096),
            ("Comedy", 1024),
            ("Thriller", 512)
        ]
        self.assertEqual(result, expected)

    def test_task3(self):
        result = task3(self.many_to_many, libraries)
        expected = {
            "Thriller": ["The Silence of the Lambs"],
            "Comedy": ["Men in Black", "Home Alone"],
            "Fiction": ["Avatar", "Interstellar"],
            "Drama": ["Avatar"]
        }
```

```
        if result != expected:
            print("\nResult:", result)
            print("\nExpected:", expected)

        self.assertDictEqual(result, expected)

if name == "main":
    unittest.main()
```

Результаты выполнения программы:



```
● [slava@nitroan51557 RK2]$ python -m unittest tdd_test.py
  ...
  ----------------------------------------------------------------------
  Ran 3 tests in 0.000s

  OK
○ [slava@nitroan51557 RK2]$ 
```