Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»

Отчет по Рубежному контролю №2
Вариант В 15

Выполнил:
студент группы ИУ5-23Б:

Османов З. Ш.
Подпись и дата:

Проверил:
преподаватель каф.

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).

Вариант В.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведитесписок всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведитесписок отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированныйпо сотрудникам, сортировка по отделам произвольная.
4. **Таблица 1.** Варианты предметной области

№ варианта	Класс 1	Класс 2
15	Программа	Компьютер

Текст программы

main,py

```
      from operator import itemgetter
class Program:
    def __init__(self, program_id, name, release_year, computer_id):
        self.id = program_id  # ID записи программы
        self.name = name  # Название программы
        self.release_year = release_year  # Год выпуска программы
        self.computer_id = computer_id  # ID компьютера, на котором установлена
программа

class Computer:
    def __init__(self, computer_id, model, owner):
        self.id = computer_id  # ID записи о компьютере
        self.model = model  # Модель компьютера
        self.owner = owner  # Владелец компьютера

class InstalledProgram:
    def __init__(self, program_id, computer_id):
        self.program_id = program_id  # ID записи программы
        self.computer_id = computer_id  # ID записи компьютера
```

```python
computers = [
    Computer(1, "Asus", "Aleksey"),
    Computer(2, "MacBook Pro", "Sasha"),
    Computer(3, "Lenovo", "Michael"),
    Computer(4, "Dell XPS", "Alice"),
    Computer(5, "MacBook Air", "Bob"),
    Computer(6, "HP ", "Charlie"),
]

programs = [
    Program(1, "Photoshop", 1988, 1),
    Program(2, "Visual Studio", 1997, 2),
    Program(3, "Zoom", 2012, 3),
    Program(4, "Slack", 2013, 1),
    Program(5, "PyCharm", 2010, 2),
    Program(6, "Microsoft Word", 1983, 4),
    Program(7, "Blender", 1998, 5),
    Program(8, "AutoCAD", 1982, 6),
    Program(9, "Notepad++", 2003, 3),
    Program(10, "Firefox", 2002, 4)
]


installed_programs = [
    InstalledProgram(1, 1),
    InstalledProgram(2, 2),
    InstalledProgram(3, 3),
    InstalledProgram(4, 1),
    InstalledProgram(5, 2),
    InstalledProgram(2, 3),
    InstalledProgram(5, 1),
    InstalledProgram(6, 4),
    InstalledProgram(7, 5),
    InstalledProgram(8, 6),
    InstalledProgram(9, 3),
    InstalledProgram(10, 4),
    InstalledProgram(1, 5),
    InstalledProgram(6, 6)
]

# Соединение один-ко-многим
one_to_many = [(p.name, p.release_year, c.model, c.owner)
               for c in computers
               for p in programs
               if p.computer_id == c.id]
print(one_to_many)

# Соединение данных многие-ко-многим
many_to_many_temp = [(c.model, ip.computer_id, ip.program_id)
    for c in computers
    for ip in installed_programs
    if c.id == ip.computer_id]
```

```python
many_to_many = [(p.name, p.release_year, computer_model)
    for computer_model, computer_id, program_id in many_to_many_temp
    for p in programs if p.id == program_id]

# Задание 1: Компьютеры, владельцы которых начинаются с "A", и установленные
программы
def first_task(one_to_many):
    result_1_temp = [(program_name, computer_model)
            for program_name, _, computer_model, owner in one_to_many
            if owner[0] == 'A']
    result_1 = [([program_name for program_name, Computer in result_1_temp if
Computer == computer_model],
                         computer_model) for computer_model in sorted(set([Computer
for _, Computer in result_1_temp]))]
    return result_1
print('Задание 1')
for program_name, computer_model in first_task(one_to_many):
    print(f"Программа: {program_name}, Компьютер: {computer_model}")


# Задание 2: Найти программы с минимальным годом выпуска для каждого компьютера,
отсортировать по году
def second_task(one_to_many):
    result_2_unsorted = []
    for computer in set([computer for _, _, computer,_ in one_to_many]):
        c_programs = [program_relise for _, program_relise, computer_model, _
in  one_to_many
                      if computer == computer_model]
        if len(c_programs) > 0:
            min_year = min(c_programs)
            result_2_unsorted.append((min_year, computer))
    result_2 = sorted(result_2_unsorted, key = itemgetter(0))
    return result_2

print('\nЗадание 2')
for computer_model, min_year in second_task(one_to_many):
    print(f"Компьютер: {computer_model}, Минимальный год выпуска программы:
{min_year}")


# Задание 3: Все компьютеры и установленные на них программы (многие-ко-многим)
# Сортируем по названию программ
def third_task(many_to_many):
    result_3 = sorted(many_to_many, key=itemgetter(0))
    return result_3

print('\nЗадание 3')
for program_name, release_year, computer_model in third_task(many_to_many):
    print(f"Программа: {program_name}, Год выпуска: {release_year}, Компьютер:
{computer_model}")
```

Tests.py

```
import unittest
import RK2
from operator import itemgetter


class TestMainMethods(unittest.TestCase):

    def test_first_task(self):
        test_list = [
            ('Photoshop', 1988, 'Asus', 'Aleksey'),
            ('Firefox', 2002, 'Toshiba', 'Andrey'),
            ('Visual Studio', 1997, 'Toshiba', 'Andrey'),
            ('Slack', 2013, 'Huawei', 'Michael'),
            ('Blender', 1998, 'MacBook Air', 'Bob'),
            ('AutoCAD', 1982, 'HP', 'Charlie'),
            ('Notepad++', 2003, 'Dell XPS', 'Alice'),
            ('Edge', 2015, 'Dell XPS', 'Alice'),
            ('Google Chrome', 2008, 'Xiaomi', 'Zagid'),
            ('Zoom', 2019, 'Lenovo', 'Ivan'),
            ('Internet Explorer', 1995, 'Thinkpad', 'Gasan'),
            ('Vivaldi', 2016, 'Lenovo', 'Ivan'),
            ('Opera', 1995, 'HP', 'Charlie'),

        ]
        result = RK2.first_task(test_list)
        reference =[
            (['Photoshop'], 'Asus'),                        #Aleksey
            (['Notepad++', 'Edge'], 'Dell XPS'),            #Alice
            (['Firefox', 'Visual Studio'], 'Toshiba'),      #Andrey
        ]
        self.assertEqual(result, reference)

    def test_second_task(self):
        test_list = [
            ('Photoshop', 1988, 'Asus', 'Aleksey'),
            ('Firefox', 2002, 'Toshiba', 'Andrey'),
            ('Visual Studio', 1997, 'Toshiba', 'Andrey'),
            ('Slack', 2013, 'Huawei', 'Michael'),
            ('Blender', 1998, 'MacBook Air', 'Bob'),
            ('AutoCAD', 1982, 'HP', 'Charlie'),
            ('Notepad++', 2003, 'Dell XPS', 'Alice'),
            ('Edge', 2015, 'Dell XPS', 'Alice'),
            ('Google Chrome', 2008, 'Xiaomi', 'Zagid'),
            ('Zoom', 2019, 'Lenovo', 'Ivan'),
            ('Internet Explorer', 1995, 'Thinkpad', 'Gasan'),
            ('Vivaldi', 2016, 'Lenovo', 'Ivan'),
            ('Opera', 1995, 'HP', 'Charlie'),
        ]
        result = RK2.second_task(test_list)

        reference =[
            ( 1982, 'HP'),
```

```
            ( 1988, 'Asus'),
            ( 1995, 'Thinkpad'),
            ( 1997, 'Toshiba'),
            ( 1998, 'MacBook Air'),
            ( 2003, 'Dell XPS'),
            ( 2008, 'Xiaomi'),
            ( 2013, 'Huawei'),
            ( 2016, 'Lenovo'),
        ]
        self.assertEqual(result, reference)

    def test_third_task(self):
        test_list = [
            ('Photoshop', 1988, 'Asus'),
            ('AutoCAD', 1982, 'Asus'),
            ('Firefox', 2002, 'Toshiba'),
            ('Visual Studio', 1997, 'Toshiba'),
            ('Slack', 2013, 'Huawei'),
            ('Notepad++', 2003, 'HP'),
            ('Edge', 2015, 'Huawei'),
            ('Blender', 1998, 'MacBook Air'),
            ('AutoCAD', 1982, 'HP'),
            ('Notepad++', 2003, 'Dell XPS'),
            ('Edge', 2015, 'Dell XPS'),
        ]
        reference = [
            ('AutoCAD', 1982, 'Asus'),
            ('AutoCAD', 1982, 'HP'),
            ('Blender', 1998, 'MacBook Air'),
            ('Edge', 2015, 'Huawei'),
            ('Edge', 2015, 'Dell XPS'),
            ('Firefox', 2002, 'Toshiba'),
            ('Notepad++', 2003, 'HP'),
            ('Notepad++', 2003, 'Dell XPS'),
            ('Photoshop', 1988, 'Asus'),
            ('Slack', 2013, 'Huawei'),
            ('Visual Studio', 1997, 'Toshiba'),
        ]
        result = RK2.third_task(test_list)
        self.assertEqual(result, reference)


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