Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2

Вариант 12А

Выполнил:
студент группы ИУ5-31Б
Ларкин Борис

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

Постановка задачи

Задание РК1:

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Вариант А.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с суммарной зарплатой сотрудников в каждом отделе, отсортированный по суммарной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.

Вар. 12	Язык программирования	Средство разработки

Задание РК2:

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
main.py

```
# используется для сортировки
from operator import itemgetter

# Язык программирования - средство разработки
class Lang:
    """ЯП"""
    """share = market share, %"""

    def __init__(self, id, name, ver, share, ide_id):
        self.id = id
        self.ver = ver
        self.name = name
        self.market_share = share
        self.ide_id = ide_id

class IDE:
    """Средство разработки"""

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

class Lang_IDE:
    """
    'Языки, поддерживаемые IDE' для реализации
    связи многие-ко-многим
    """

    def __init__(self, l_id, ide_id):
        self.l_id = l_id
        self.ide_id = ide_id

# Languages
langs = [
    Lang(1, 'Python', '3.11', 29.48, 2),  # VSCODE
    Lang(2, 'C#', '11', 6.94, 1),  # MVS
    Lang(3, 'C', 'C17', 6.49, 2),  # VSCODE

    Lang(11, 'C++', 'C++20', 6.49, 4),  # IDEA
    Lang(22, 'Java', 'Java SE 21', 17.18, 5),  # Eclipse
    Lang(33, 'GO', '1.21.3', 36.1, 3),  # Komodo
]

# Сотрудники
IDEs = [
    IDE(1, 'Microsoft Visual Studio', 2023),
    IDE(2, 'Visual Studio Code', 2023),
    IDE(3, 'Komodo', 2022),
    IDE(4, 'IntelliJ IDEA', 2023),
    IDE(5, 'Eclipse', 2022),
]

IDEs_langs = [
    Lang_IDE(1, 2),  # Py - VSCODE
    Lang_IDE(2, 1),  # C# - MVS
```

```
    Lang_IDE(3, 2),  # C - VSCODE
    Lang_IDE(1, 1),  # Py - MVS
    Lang_IDE(2, 2),  # C# - VSCODE

    Lang_IDE(11, 4),  # C++  - IDEA
    Lang_IDE(22, 5),  # Java - Eclipse
    Lang_IDE(33, 3),  # GO - Komodo
    Lang_IDE(11, 1),  # C++ - MVS
    Lang_IDE(11, 2),  # C++ - VSCODE
]

def solve_a1(one_to_many):
    res_11 = sorted(one_to_many, key=itemgetter(3))  # Сортировка по названию IDE
    return res_11

def solve_a2(one_to_many):
    res_12_unsorted = []
    # Перебираем все IDE
    for i in IDEs:
        # Список языков, поддерживаемых средой
        ide_langs = list(filter(lambda k: k[3] == i.name, one_to_many))
        # Если хотя бы один язык поддерживается
        if len(ide_langs) > 0:
            # Доли рынка каждого языка IDE
            ide_market_shares = [share for _, _, share, _, _ in ide_langs]
            # Общая доля рынка поддерживаемых языков
            ide_ms_sum = sum(ide_market_shares)
            res_12_unsorted.append((i.name, ide_ms_sum))

    # Сортировка IDE по суммарной доле рынка
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

def solve_a3(many_to_many):
    res_13 = {}
    # Перебираем все IDE
    for i in IDEs:
        if 'Visual Studio' in i.name:  # Если в названии присутствует "Visual
Studio"
            # Список языков IDE
            i_langs = list(filter(lambda k: k[0] == i.name, many_to_many))
            # Только названия языков
            i_langs_names = [x for _, _, x, _, _ in i_langs]
            # Добавляем результат в словарь
            # ключ - IDE, значение - список языков
            res_13[i.name] = i_langs_names

    return res_13

one_to_many = []
many_to_many = []

def create_1m() -> list:
    # Соединение данных один-ко-многим
    result = [(l.name, l.ver, l.market_share, i.name, i.year)
              for l in langs
              for i in IDEs
              if i.id == l.ide_id]
    return result
```

```
def create_mm() -> list:
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(i.name, i.year, il.l_id)
                         for i in IDEs
                         for il in IDEs_langs
                         if i.id == il.ide_id]

    result = [(i_name, i_year, l.name, l.ver, l.market_share)
              for i_name, i_year, l_id in many_to_many_temp
              for l in langs if l.id == l_id]

    return result

def main():
    """Основная функция"""
    one_to_many = create_1m()
    many_to_many = create_mm()

    print('Задание A1')
    print(solve_a1(one_to_many))
    print('\nЗадание A2')
    print(solve_a2(one_to_many))
    print('\nЗадание A3')
    print(solve_a3(many_to_many))

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

tddtests.py

```
import unittest
from main import *
one_to_many = create_1m()
many_to_many = create_mm()
class TestMain(unittest.TestCase):
    def test_A1(self):
        self.assertEqual(solve_a1(one_to_many),
                         [('Java', 'Java SE 21', 17.18, 'Eclipse', 2022), ('C++',
'C++20', 6.49, 'IntelliJ IDEA', 2023),
                          ('GO', '1.21.3', 36.1, 'Komodo', 2022), ('C#', '11',
6.94, 'Microsoft Visual Studio', 2023),
                          ('Python', '3.11', 29.48, 'Visual Studio Code', 2023),
                          ('C', 'C17', 6.49, 'Visual Studio Code', 2023)])

    def test_A2(self):
        self.assertEqual(solve_a2(one_to_many), [('Komodo', 36.1), ('Visual
Studio Code', 35.97), ('Eclipse', 17.18),
                                                 ('Microsoft Visual Studio',
6.94), ('IntelliJ IDEA', 6.49)])

    def test_A3(self):
        self.assertEqual(solve_a3(many_to_many), {'Microsoft Visual Studio':
['C#', 'Python', 'C++'],
                                          'Visual Studio Code': ['Python', 'C', 'C#',
'C++']})

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

Результаты выполнения

Результаты выполнения РК №1:

Задание А1

[('Java', 'Java SE 21', 17.18, 'Eclipse', 2022), ('C++', 'C++20', 6.49, 'IntelliJ IDEA', 2023), ('GO', '1.21.3', 36.1, 'Komodo', 2022), ('C#', '11', 6.94, 'Microsoft Visual Studio', 2023), ('Python', '3.11', 29.48, 'Visual Studio Code', 2023), ('C', 'C17', 6.49, 'Visual Studio Code', 2023)]

Задание А2

[('Komodo', 36.1), ('Visual Studio Code', 35.97), ('Eclipse', 17.18), ('Microsoft Visual Studio', 6.94), ('IntelliJ IDEA', 6.49)]

Задание А3

{'Microsoft Visual Studio': ['C#', 'Python', 'C++'], 'Visual Studio Code': ['Python', 'C', 'C#', 'C++']}

Результаты выполнения РК №2:

Используем эти значения как параметры assertEqual() для каждого из тестов и запустим код на выполнение:

```
----------------------------------------------------------------------
Ran 3 tests in 0.004s

OK
```

Затем изменим, скажем, в ожидаемом результате выполнения solve_a3(many_to_many), C++ на Go: {'Microsoft Visual Studio': ['C#', 'Python', 'Go'], 'Visual Studio Code': ['Python', 'C', 'C#', 'Go']}. Запустим на выполнение.

```
..F
======================================================================
FAIL: test_A3 (__main__.TestMain)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "E:\Git\BKIT2023\RK2\tddtest.py", line 20, in test_A3
    self.assertEqual(solve_a3(many_to_many), {'Microsoft Visual Studio': ['C#',
'Python', 'Go'], 'Visual Studio Code': ['Python', 'C', 'C#', 'Go']})
AssertionError: {'Mic[36 chars]n', 'C++'], 'Visual Studio Code': ['Python', 'C',
 'C#', 'C++']} != {'Mic[36 chars]n', 'Go'], 'Visual Studio Code': ['Python', 'C'
, 'C#', 'Go']}
- {'Microsoft Visual Studio': ['C#', 'Python', 'C++'],
?                                              ^^^

+ {'Microsoft Visual Studio': ['C#', 'Python', 'Go'],
?                                              ^^

-  'Visual Studio Code': ['Python', 'C', 'C#', 'C++']}
?                                              ^^^

+  'Visual Studio Code': ['Python', 'C', 'C#', 'Go']}
?                                              ^^


----------------------------------------------------------------------
Ran 3 tests in 0.006s

FAILED (failures=1)
```