**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: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Ларкин Борис |  | 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 А1')

    print(solve\_a1(one\_to\_many))

    print('\nЗадание А2')

    print(solve\_a2(one\_to\_many))

    print('\nЗадание А3')

    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:



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.

