Отчёт по рубежному контролю №1 по курсу «Базовые компоненты интернет-технологий»

Вариант 1.

Руководитель   
Гапанюк Ю.Е.

Выполнил  
студент группы ИУ5-31Б   
Абуховский Иван  
23.10.2021

МГТУ им. Н.Э. Баумана   
2021 г.

**Задание**

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

1. Класс «Студент», содержащий поля:
   * ID записи о студенте;
   * Фамилия студента;
   * Количество долгов;
   * ID записи о группе. (для реализации связи один-ко-многим)
2. Класс «Группа», содержащий поля:
   * ID записи о группе;
   * Наименование группы.
3. (Для реализации связи многие-ко-многим) Класс «Студенты группы», содержащий поля:
   * ID записи о студенте;
   * ID записи о группе.

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

3) Необходимо разработать запросы в соответствии с вариантом. Запросы сформулированы в терминах классов «Студент» и «Группа», которые используются в примере. Нужно перенести эти требования в выданный вариант предметной области.

**Предметная область: Вариант 1 - Студент-Группа**

**Запрос: Вариант А.**

1. «Группа» и «Студент» связаны соотношением один-ко-многим. Выведите список всех связанных студентов и групп, отсортированный по группам, сортировка по студентам произвольная.
2. «Группа» и «Студент» связаны соотношением один-ко-многим. Выведите список групп с суммарным количеством долгов студентов в каждой группе, отсортированный по суммарной количеству долгов.
3. «Группа» и «Студент» связаны соотношением многие-ко-многим. Выведите список всех групп, у которых в названии присутствует индекс «Б», и список учащихся в них студентов.

**Текст программы**

from operator import itemgetter

class Student:

"""Студент"""

def \_\_init\_\_(self, id, surname, dolgi, group\_id):

self.id = id

self.surname = surname

self.dolgi = dolgi

self.group\_id = group\_id

class Group:

"""Группа"""

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class StudentGroup:

"""

'Студенты группы' для реализации

связи многие-ко-многим

"""

def \_\_init\_\_(self, group\_id, student\_id):

self.group\_id = group\_id

self.student\_id = student\_id

# Группы

groups = [

Group(1, 'ИУ5-31Б'),

Group(2, 'ИУ6-12Б'),

Group(3, 'ИУ7-53Б'),

Group(4, 'ИУ8-34Б'),

Group(5, 'ИУ9-15Б'),

Group(11, 'МТ5-11'),

Group(22, 'РК5-32'),

Group(33, 'СМ5-13'),

Group(44, 'Э5-54'),

Group(55, 'ОЭП5-35'),

]

# Студенты

students = [

Student(1, 'Абуховский', 25, 1),

Student(2, 'Черноморец', 47, 2),

Student(3, 'Шагиахметов', 39, 3),

Student(4, 'Стельмах', 81, 4),

Student(5, 'Сыса', 17, 5),

Student(6, 'Слепченкова', 43, 11),

Student(7, 'Кузьмин', 31, 22),

Student(8, 'Соколов', 29, 33),

Student(9, 'Заточен', 48, 44),

Student(10, 'Проценко', 33, 55),

Student(11, 'Калинников', 21, 1),

Student(12, 'Милевич', 21, 2),

Student(13, 'Слоква', 65, 3),

Student(14, 'Барабанщиков', 15, 4),

Student(15, 'Акулова', 83, 5),

Student(16, 'Ашуров', 12, 11),

Student(17, 'Бекетов', 14, 22),

Student(18, 'Поляков', 11, 33),

Student(19, 'Нигматуллин', 10, 44),

Student(20, 'Собакевич', 10, 55),

Student(21, 'Ахтамбаев', 32, 3),

Student(22, 'Цуприков', 15, 33),

]

students\_groups = [

StudentGroup(1,1),

StudentGroup(1,11),

StudentGroup(2,2),

StudentGroup(2,12),

StudentGroup(3,3),

StudentGroup(3,13),

StudentGroup(4,4),

StudentGroup(4,13),

StudentGroup(5,5),

StudentGroup(5,15),

StudentGroup(11,6),

StudentGroup(11,16),

StudentGroup(22,7),

StudentGroup(22,17),

StudentGroup(33,8),

StudentGroup(33,18),

StudentGroup(44,9),

StudentGroup(44,19),

StudentGroup(55,10),

StudentGroup(55,20),

StudentGroup(3,21),

StudentGroup(33,22),

]

def main():

"""Основная функция"""

# Соединение данных один-ко-многим

one\_to\_many = [(s.surname, s.dolgi, g.name)

for g in groups

for s in students

if s.group\_id==g.id]

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [(g.name, sg.group\_id, sg.student\_id)

for g in groups

for sg in students\_groups

if g.id==sg.group\_id]

many\_to\_many = [(s.surname, s.dolgi, group\_name)

for group\_name, group\_id, student\_id in many\_to\_many\_temp

for s in students if s.id==student\_id]

print('Задание А1')

res\_11 = sorted(one\_to\_many, key=itemgetter(2))

print(res\_11)

print('\nЗадание А2')

res\_12\_unsorted = []

# Перебираем все группы

for g in groups:

# Список студентов группы

g\_students = list(filter(lambda i: i[2]==g.name, one\_to\_many))

# Если группа не пустая

if len(g\_students) > 0:

# Долги студентов группы

g\_dolgi = [dolgi for \_,dolgi,\_ in g\_students]

# Суммарное количество долгов студентов группы

g\_dolgi\_sum = sum(g\_dolgi)

res\_12\_unsorted.append((g.name, g\_dolgi\_sum))

# Сортировка по суммарной зарплате

res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)

print(res\_12)

print('\nЗадание А3')

res\_13 = {}

# Перебираем все группы

for g in groups:

if 'Б' in g.name:

# Список студентов группы

g\_students = list(filter(lambda i: i[2]==g.name, many\_to\_many))

# Только ФИО сотрудников

g\_students\_surnames = [x for x,\_,\_ in g\_students]

# Добавляем результат в словарь

# ключ - отдел, значение - список фамилий

res\_13[g.name] = g\_students\_surnames

print(res\_13)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Результат выполнения программы**

