**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по рубежному контролю №1

Вариант Г9

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-54Б |  | преподаватель каф. ИУ5 |
| Киреев А.А. |  | Ю.Е. Гапанюк. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Условия рубежного контроля №1 по курсу РИП**

Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

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

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
   * ID записи о сотруднике;
   * Фамилия сотрудника;
   * Зарплата (количественный признак);
   * ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс «Отдел», содержащий поля:
   * ID записи об отделе;
   * Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
   * ID записи о сотруднике;
   * ID записи об отделе.

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 Г.**

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с максимальной зарплатой сотрудников в каждом отделе, отсортированный по максимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.

**Задание по варианту.**

|  |  |  |
| --- | --- | --- |
| 9 | Операционная система | Компьютер |

**Текст программы.**

# используется для сортировки  
from operator import itemgetter  
  
  
class Os:  
 *"""Операционная система"""* def \_\_init\_\_(self, id, name, start, comp\_id):  
 self.id = id  
 self.name = name  
 self.start = start  
 self.comp\_id = comp\_id  
  
class Computer:  
 *"""Компьютер"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class OsComp:  
 *"""  
 'Операционные системы в компьютере' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, os\_id, comp\_id):  
 self.os\_id = os\_id  
 self.comp\_id = comp\_id  
  
  
# Компьютеры  
Comps = [  
 Computer(1, 'Настольный ПК'),  
 Computer(2, 'Ноутбук'),  
 Computer(3, 'Игровой ПК'),  
]  
  
# Операционные системы  
OSs = [  
 Os(1, 'Windows', 1239, 3),  
 Os(2, 'Mint', 267, 1),  
 Os(3, 'MacOS', 873, 2),  
 Os(4, 'Ubuntu', 365, 3),  
 Os(5, 'Debian', 89, 1),  
 Os(6, 'Chrome OS', 99, 3),  
 Os(7, 'Fedora', 17, 1),  
]  
  
oss\_comps = [  
 OsComp(1, 3),  
 OsComp(2, 1),  
 OsComp(3, 2),  
 OsComp(4, 3),  
 OsComp(5, 1),  
 OsComp(6, 3),  
 OsComp(7, 1),  
]  
  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(os\_temp.name, os\_temp.start, comp\_temp.name)  
 for comp\_temp in Comps  
 for os\_temp in OSs  
 if os\_temp.comp\_id == comp\_temp.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(comp\_temp.name, os\_temp.comp\_id, os\_temp.os\_id)  
 for comp\_temp in Comps  
 for os\_temp in oss\_comps  
 if comp\_temp.id == os\_temp.comp\_id]  
  
 many\_to\_many = [(os\_temp.name, os\_temp.start, comp\_name)  
 for comp\_name, comp\_id, os\_id in many\_to\_many\_temp  
 for os\_temp in OSs if os\_temp.id == os\_id]  
  
 print('Задание Г1')  
 res\_11 = [(os\_temp.name, comp\_temp.name)  
 for comp\_temp in Comps  
 for os\_temp in OSs  
 if (os\_temp.comp\_id == comp\_temp.id)&(comp\_temp.name[:13] == "Настольный ПК")]  
 print(res\_11)  
  
  
  
 print('\nЗадание Г2')  
 res\_12\_unsorted = []  
 # Перебираем все компьютеры  
 for comp\_temp in Comps:  
 # Список ОС в компьютерах  
 c\_oses = list(filter(lambda i: i[2] == comp\_temp.name, one\_to\_many))  
 # Если компьютер не пуст  
 if len(c\_oses) > 0:  
 # Сколько раз запускали ОС  
 c\_starts = [start for \_, start, \_ in c\_oses]  
 # Максимальное число запусков  
 c\_starts\_max = max(c\_starts)  
 res\_12\_unsorted.append((comp\_temp.name, c\_starts\_max))  
  
 # Сортировка по максимальному числу запусков  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 print(res\_12)  
  
 print('\nЗадание Г3')  
 res\_13 = sorted(many\_to\_many, key=itemgetter(2))  
 print(res\_13)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результаты работы программы.**

