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»

РК №1

Вариант №1 (Г)

Выполнил:
Студент группы ИУ5-34Б
Беккиев Рашид
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2023 г.

Постановка задачи:

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).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результаты ее выполнения.

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.

1	Студент	Группа

Код программы:

```
class Group:
    def __init__(self, group_id, name):
        self.group_id = group_id
        self.name = name
        self.students = []

    def add_student(self, student):
        self.students.append(student)

class Student:
    def __init__(self, student_id, name, birth_year, group):
        self.student_id = student_id
        self.name = name
        self.birth_year = birth_year
        self.group = group

    def __str__(self):
        return f"{self.name} ({self.birth_year})"

class many_to_many:
    def __init__(self, student_id, group_id):
        self.student_id = student_id
        self.group_id = group_id

# Создаем тестовые данные
group1 = Group(1, "А1")
group2 = Group(2, "Б1")

student1 = Student(1, "Иванов", 1998, group1)
student2 = Student(2, "Петров", 1997, group1)
student3 = Student(3, "Сидоров", 1999, group1)
student4 = Student(4, "Иваненко", 1996, group2)
student5 = Student(5, "Артемов", 1999, group2)

group1.add_student(student1)
group1.add_student(student2)
group1.add_student(student3)
group2.add_student(student4)
group2.add_student(student5)

groups = [group1, group2]
students = [student1, student2, student3, student4, student5]

many_to_many = [
    many_to_many(1, 1),
```

```
    many_to_many(2, 1),
    many_to_many(3, 1),
    many_to_many(4, 2),
    many_to_many(5, 2),
]

students_dict = {student.student_id: student for student in students}
groups_dict = {group.group_id: group for group in groups}

# Запрос 1
print("Задание 1")
for group in groups:
    if group.name.startswith('A'):
        print(group.name)
        for student in group.students:
            print(f"\t{student.name} {student.birth_year}")

# Запрос 2
print("\nЗадание 2")
max_birth_years = {}
for group in groups:
    max_year = max(student.birth_year for student in group.students)
    max_birth_years[group.name] = max_year

for group, year in max_birth_years.items():
    print((group, year))

# Запрос 3
print("\nЗадание 3")
for group in groups:
    print(group.name)
    for student in group.students:
        print(f"\t{student.name}")
```

Результат выполнения программы:

```
/Users/razzle/PycharmProjects/PK1/venv/bin/python /Users/razzle/PycharmProjects/PK1/main.py
Задание 1
А1
    Иванов 1998
    Петров 1997
    Сидоров 1999

Задание 2
('А1', 1999)
('Б1', 1999)

Задание 3
А1
    Иванов
    Петров
    Сидоров
Б1
    Иваненко
    Артемов

Process finished with exit code 0
```