

Отчёт

По рк-1

**Дисциплина «Парадигмы и конструкции языков
программирования»**

Студент: Якубович Анна

Группа: ИБМ3-23Б

Преподаватель: Гапанюк Ю.Е.

Задание № 1:

Задание:

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 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.

Вариант №26:

Предметная область: Студенческая группа и учебный курс

Код программы:

```
class Student:
    """Класс «Студент», содержащий поля:
    • ID записи о студенте;
    • Фамилия студента;
    • Средний балл (количественный признак);
    • ID записи о группе. (для реализации связи один-ко-многим)"""

    def __init__(self, student_id, last_name, average_grade, group_id):
        self.student_id = student_id
        self.last_name = last_name
        self.average_grade = average_grade
        self.group_id = group_id

    def __repr__(self):
        return f"Student(ID={self.student_id}, Фамилия='{self.last_name}', Средний балл={self.average_grade}, GroupID={self.group_id})"

class StudyGroup:
    def __init__(self, group_id, group_name):
        self.group_id = group_id
        self.group_name = group_name

    def __repr__(self):
        return f"StudyGroup(ID={self.group_id}, Наименование='{self.group_name}')"

class StudentCourse:
    """Класс «Студенты курса», содержащий поля:
    • ID записи о студенте;
    • ID записи о группе.
    (Для реализации связи многие-ко-многим)"""

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

    def __repr__(self):
        return f"StudentCourse(StudentID={self.student_id}, GroupID={self.group_id})"

def main():
    print("=" * 80)
    print("РУБЕЖНЫЙ КОНТРОЛЬ №1")
    print("Предметная область: Студенческая группа и Учебный курс")
    print("=" * 80)
    groups = [
        StudyGroup(1, "ИТ-101"),
        StudyGroup(2, "ПИ-202"),
        StudyGroup(3, "Отдел математики"),
        StudyGroup(4, "ФИ-404"),
        StudyGroup(5, "Отдел программирования")
    ]
```

```

]
students = [
    Student(1, "Иванов", 4.5, 1),
    Student(2, "Петров", 3.8, 1),
    Student(3, "Сидоров", 4.2, 2),
    Student(4, "Кузнецов", 3.9, 2),
    Student(5, "Смирнов", 4.8, 3),
    Student(6, "Васильев", 4.1, 4),
    Student(7, "Попов", 3.7, 5)
]
student_courses = [
    StudentCourse(1, 1),
    StudentCourse(2, 1),
    StudentCourse(3, 2),
    StudentCourse(4, 2),
    StudentCourse(5, 3),
    StudentCourse(6, 4),
    StudentCourse(7, 5),
    StudentCourse(1, 3),
    StudentCourse(3, 5)
]

print("\nТЕСТОВЫЕ ДАННЫЕ")
print("-" * 40)

print("\nУчебные группы:")
for group in groups:
    print(f" {group}")

print("\nСтуденты:")
for student in students:
    print(f" {student}")

print("\nСвязи студентов и групп (многие-ко-многим):")
for sc in student_courses:
    print(f" {sc}")
print("\n" + "=" * 80)
print("РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЗАПРОСОВ")
print("=" * 80)
print("\nЗАПРОС 1: Список всех связанных студентов и групп (отсортированный по группам)")
print("-" * 60)
groups_dict = {group.group_id: group for group in groups}
students_by_group = {}
for student in students:
    if student.group_id not in students_by_group:
        students_by_group[student.group_id] = []
    students_by_group[student.group_id].append(student)
sorted_groups = sorted(groups, key=lambda g: g.group_name)

for group in sorted_groups:
    group_students = students_by_group.get(group.group_id, [])
    print(f"\nГруппа: {group.group_name}")

```

```

        if group_students:
            for student in sorted(group_students, key=lambda s: s.last_name):
                print(f"    └ {student.last_name} (средний балл: {stu-
dent.average_grade}")
            else:
                print("    └ В группе нет студентов")
print("\nЗАПРОС 2: Список групп с суммарным средним баллом студентов")
print("-" * 60)
group_total_grades = [
    (group,
     sum(student.average_grade for student in students if student.group_id ==
group.group_id),
     len([student for student in students if student.group_id == group.group_id]))
    for group in groups
]
group_total_grades.sort(key=lambda x: x[1], reverse=True)

for group, total_grade, student_count in group_total_grades:
    avg_per_student = total_grade / student_count if student_count > 0 else 0
    print(f"\nГруппа: {group.group_name}")
    print(f"    Количество студентов: {student_count}")
    print(f"    Суммарный средний балл: {total_grade:.2f}")
    print(f"    Средний балл на студента: {avg_per_student:.2f}")
print("\nЗАПРОС 3: Список всех групп с 'отдел' в названии и их студенты")
print("-" * 60)

department_groups = [group for group in groups if "отдел" in
group.group_name.lower()]

students_dict = {student.student_id: student for student in students}

for group in department_groups:
    print(f"\nГруппа: {group.group_name}")
    group_student_ids = [sc.student_id for sc in student_courses if sc.group_id ==
group.group_id]
    group_students = [students_dict[student_id] for student_id in group_student_ids]

    if group_students:
        for student in sorted(group_students, key=lambda s: s.last_name):
            print(f"    └ {student.last_name} (средний балл: {stu-
dent.average_grade}")
        else:
            print("    └ В этой группе нет студентов")
print("\n" + "=" * 80)
print("ДОПОЛНИТЕЛЬНЫЙ АНАЛИЗ")
print("=" * 80)
student_group_count = {}
for sc in student_courses:
    student_group_count[sc.student_id] = student_group_count.get(sc.student_id, 0) +
1

multi_group_students = [(student_id, count) for student_id, count in stu-
dent_group_count.items() if count > 1]

```

```

if multi_group_students:
    print("\nСтуденты, обучающиеся в нескольких группах:")
    for student_id, count in multi_group_students:
        student = students_dict[student_id]
        student_groups = [groups_dict[sc.group_id].group_name for sc in student_courses if sc.student_id == student_id]
        print(f" {student.last_name}: {count} групп - {' '.join(student_groups)}")
    else:
        print("\nНет студентов, обучающихся в нескольких группах")

if __name__ == "__main__":
    main()

```

Пример работы программы:

ЗАПРОС 1: Список всех связанных студентов и групп (отсортированный по группам)

Группа: ИТ-101

- └ Иванов (средний балл: 4.5)
- └ Петров (средний балл: 3.8)

Группа: Отдел математики

- └ Смирнов (средний балл: 4.8)

Группа: Отдел программирования

- └ Попов (средний балл: 3.7)

Группа: ПИ-202

- └ Кузнецов (средний балл: 3.9)
- └ Сидоров (средний балл: 4.2)

Группа: ФИ-404

- └ Васильев (средний балл: 4.1)

ЗАПРОС 2: Список групп с суммарным средним баллом студентов

Группа: ИТ-101

Количество студентов: 2
 Суммарный средний балл: 8.30
 Средний балл на студента: 4.15

Группа: ПИ-202

Количество студентов: 2
 Суммарный средний балл: 8.10
 Средний балл на студента: 4.05

Группа: Отдел математики
Количество студентов: 1
Суммарный средний балл: 4.80
Средний балл на студента: 4.80

Группа: ФИ-404
Количество студентов: 1
Суммарный средний балл: 4.10
Средний балл на студента: 4.10

Группа: Отдел программирования
Количество студентов: 1
Суммарный средний балл: 3.70
Средний балл на студента: 3.70

ЗАПРОС 3: Список всех групп с 'отдел' в названии и их студенты

Группа: Отдел математики
└─ Иванов (средний балл: 4.5)
└─ Смирнов (средний балл: 4.8)

Группа: Отдел программирования
└─ Попов (средний балл: 3.7)
└─ Сидоров (средний балл: 4.2)

=====

ДОПОЛНИТЕЛЬНЫЙ АНАЛИЗ

=====

Студенты, обучающиеся в нескольких группах:
Иванов: 2 групп - ИТ-101, Отдел математики
Сидоров: 2 групп - ПИ-202, Отдел программирования