rk1_refactored.py

```python
#РК 2 Королев А.С. ИУ5Ц-51Б (А/28)

# Классы:
class Student:
    def __init__(self, student_id, name, scholarship, department_id):
        self.student_id = student_id
        self.name = name
        self.scholarship = scholarship
        self.department_id = department_id

    def __repr__(self):
        return f"Student(id={self.student_id}, name={self.name}, scholarship={self.scholarship},
department_id={self.department_id})"

class Department:
    def __init__(self, department_id, name):
        self.department_id = department_id
        self.name = name
        self.students = []

    def __repr__(self):
        return f"Department(id={self.department_id}, name={self.name})"

class StudentDepartment:
    def __init__(self, student_id, department_id):
        self.student_id = student_id
        self.department_id = department_id

    def __repr__(self):
        return f"StudentDepartment(student_id={self.student_id}, department_id={self.department_id})"

# Функции для обработки данных
def link_students_to_departments(students, departments):
    for student in students:
        for department in departments:
            if student.department_id == department.department_id:
                department.students.append(student)

def query_1(departments):
    result = []
    for department in departments:
        result.append((department, department.students))
    return result

def query_2(departments):
    result = []
    for department in departments:
        total_scholarship = sum(student.scholarship for student in department.students)
        result.append((department, total_scholarship))
    return sorted(result, key=lambda x: x[1], reverse=True)

def query_3(departments):
    result = []
    for department in departments:
        if "кафедра" in department.name.lower():
            result.append((department, department.students))
    return result

# Пример использования
if __name__ == "__main__":
    # Создание списков объектов классов с тестовыми данными
    students = [
        Student(1, "Королев", 10000, 1),
        Student(2, "Петров", 12000, 1),
```

```
    Student(3, "Пронин", 11000, 2),
    Student(4, "Иванов", 13000, 2),
    Student(5, "Смирнов", 14000, 3)
]

departments = [
    Department(1, "Кафедра математики"),
    Department(2, "Кафедра физики"),
    Department(3, "Кафедра информатики")
]

student_departments = [
    StudentDepartment(1, 1),
    StudentDepartment(2, 1),
    StudentDepartment(3, 2),
    StudentDepartment(4, 2),
    StudentDepartment(5, 3)
]

# Связываем студентов с кафедрами
link_students_to_departments(students, departments)

# Запрос 1: Список всех связанных студентов и кафедр
print("Запрос 1:")
for department, students in query_1(departments):
    print(f"Кафедра: {department.name}")
    for student in students:
        print(f"  Студент: {student.name}")

# Запрос 2: Список кафедр с суммарной стипендией студентов на каждой кафедре, отсортированный по суммарной стипендии
print("\nЗапрос 2:")
for department, total_scholarship in query_2(departments):
    print(f"Кафедра: {department.name}, Суммарная стипендия: {total_scholarship}")

# Запрос 3: Список всех кафедр, у которых в названии присутствует слово "кафедра", и список студентов, обучающихся на них
print("\nЗапрос 3:")
for department, students in query_3(departments):
    print(f"Кафедра: {department.name}")
    for student in students:
        print(f"  Студент: {student.name}")
```

test_rk1.py

```
import unittest
from rk1_refactored import Student, Department, link_students_to_departments, query_1, query_2, query_3

class TestRK1(unittest.TestCase):

    def setUp(self):
        self.students = [
            Student(1, "Королев", 10000, 1),
            Student(2, "Петров", 12000, 1),
            Student(3, "Пронин", 11000, 2),
            Student(4, "Иванов", 13000, 2),
            Student(5, "Смирнов", 14000, 3)
        ]

        self.departments = [
            Department(1, "Кафедра математики"),
            Department(2, "Кафедра физики"),
            Department(3, "Кафедра информатики")
        ]
```

```
        link_students_to_departments(self.students, self.departments)
    #Проверяет, что функция query_1 возвращает правильное количество кафедр и студентов.
    def test_query_1(self):
        result = query_1(self.departments)
        self.assertEqual(len(result), 3)
        self.assertEqual(len(result[0][1]), 2)
        self.assertEqual(len(result[1][1]), 2)
        self.assertEqual(len(result[2][1]), 1)
    #Проверяет, что функция query_2 возвращает правильные суммы стипендий для каждой кафедры.
    def test_query_2(self):
        result = query_2(self.departments)
        self.assertEqual(len(result), 3)
        self.assertEqual(result[0][1], 24000)
        self.assertEqual(result[1][1], 22000)
        self.assertEqual(result[2][1], 14000)
    #Проверяет, что функция query_3 возвращает только те кафедры, в названии которых есть слово "кафедра".
    def test_query_3(self):
        result = query_3(self.departments)
        self.assertEqual(len(result), 3)
        self.assertEqual(result[0][0].name, "Кафедра математики")
        self.assertEqual(result[1][0].name, "Кафедра физики")
        self.assertEqual(result[2][0].name, "Кафедра информатики")

if __name__ == "__main__":
    unittest.main()
```

Результат:

...

--

Ran 3 tests in 0.001s

OK