**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»

Отчет по РК №1

Вариант запросов: В

Вариант предметной области: 2

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Абрамов А. В. |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 г.

**Вариант запросов B.**

# «Школьник» и «Класс» связаны соотношением один-ко-многим. Выведите список всех школьников, фамилия которых начинается на букву «А».

# «Школьник» и «Класс» связаны соотношением один-ко-многим. Выведите список классов с минимальным средним баллом в каждом классе, отсортированный по минимальному среднему баллу.

# «Школьник» и «Класс» связаны соотношением многие-ко-многим. Выведите список всех связанных школьников и классов, отсортированный по школьникам, сортировка классов произвольная.

# **Код программы**

class Student:  
 def \_\_init\_\_(self, student\_id, name, class\_id, avg\_rating):  
 self.student\_id = student\_id  
 self.name = name  
 self.class\_id = class\_id  
 self.avg\_rating = avg\_rating  
  
  
class Class:  
 def \_\_init\_\_(self, class\_id, name):  
 self.class\_id = class\_id  
 self.name = name  
  
  
# Создаем объекты класса Class  
second\_B = Class(1, "2Б")  
tenth\_A = Class(2, "10А")  
fifth\_V = Class(3, "5В")  
  
# Создаем объекты класса Student  
student1 = Student(1, "Лупарев", 1, 4.9)  
student2 = Student(2, "Гукасян", 1, 3.8)  
student3 = Student(3, "Абрамов", 2, 4.5)  
student4 = Student(4, "Иноземцев", 2, 2.7)  
student5 = Student(5, "Барсукова", 3, 5.0)  
  
# Создаем список "Школьникик и классы" для связи один-ко-многим  
student\_class = [  
 (student1, second\_B),  
 (student2, second\_B),  
 (student3, tenth\_A),  
 (student4, tenth\_A),  
 (student5, fifth\_V)  
]  
  
  
def query1():  
 # Задание В1  
 print("Задание В1")  
 for student, cls in student\_class:  
 if student.name.startswith('А'):  
 print(f"{student.name} - {cls.name}")  
  
  
def query2():  
 # Задание В2  
 print("\nЗадание В2")  
 class\_min\_ratings = {}  
 for student, cls in student\_class:  
 if cls.name in class\_min\_ratings:  
 if student.avg\_rating < class\_min\_ratings[cls.name]:  
 class\_min\_ratings[cls.name] = student.avg\_rating  
 else:  
 class\_min\_ratings[cls.name] = student.avg\_rating  
  
 sorted\_classes = sorted(class\_min\_ratings.items(), key=lambda x: x[1])  
 for group, min\_rating in sorted\_classes:  
 print(f"{group} - Минимальный рейтинг: {min\_rating}")  
  
  
def query3():  
 # Задание В3  
 print("\nЗадание В3")  
 student\_class.sort(key=lambda x: x[0].name)  
 for student, cls in student\_class:  
 print(f"{student.name} - {cls.name}")  
  
  
def main():  
 query1()  
 query2()  
 query3()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# **Результат выполнения программы**

# 