|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Факультет «ГУИМЦ»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Курс«Парадигмы и конструкции языков программирования»»

ОТЧЕТ

ПО РУБЕЖНОМУ КОНТРОЛЮ №1

Вариант предметной области28

Вариант запросов: B

Студент: Агапова А.Д., группа ИУ5Ц-52Б

Преподаватель: Гапанюк Ю. Е.

2024г.

**Вариант запросов B. Предметная область 28**

1 «Кафедра» и «Студенческая группа» связаны соотношением один-ко-многим.

Выведите список всех групп, относящихся к факультету ИУ (название

группы начинается с «ИУ»), и названия их кафедр.

2 «Кафедра» и «Студенческая группа» связаны соотношением один-ко-многим.

Выведите список кафедр с максимальным количеством

людей в группе на каждой кафедре, отсортированный по

минимальному количеству людей.

3 «Кафедра» и «Студенческая группа» связаны соотношением многие-ко-

многим. Выведите список всех связанных групп и кафедр,

отсортированный по группам, сортировка по кафедрам

произвольная.

**Рубежный контроль №2:**

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

**Листинг программы**

**main.py**

class Group:  
 def \_\_init\_\_(self, id: int, department: str, sem: int, index: int, faculty\_id: int, students: int):  
 self.\_id = id  
 self.\_name = f"{department}-{sem}{index}"  
 self.\_faculty\_id = faculty\_id  
 self.\_students = students  
  
 @property  
 def id(self) -> int:  
 return self.\_id  
  
 @property  
 def faculty\_id(self) -> int:  
 return self.\_faculty\_id  
  
 @property  
 def name(self) -> str:  
 return self.\_name  
  
 @property  
 def students(self) -> int:  
 return self.\_students  
  
  
class Faculty:  
 def \_\_init\_\_(self, id: int, title: str):  
 self.\_id = id  
 self.\_title = title  
  
 @property  
 def id(self) -> int:  
 return self.\_id  
  
 @property  
 def title(self) -> str:  
 return self.\_title  
  
  
class GroupFaculty:  
 def \_\_init\_\_(self, group\_id: int, faculty\_id: int):  
 self.\_group\_id = group\_id  
 self.\_faculty\_id = faculty\_id  
  
 @property  
 def faculty\_id(self) -> int:  
 return self.\_faculty\_id  
  
 @property  
 def group\_id(self) -> int:  
 return self.\_group\_id  
  
  
def task1(faculties: list[Faculty], groups: list[Group]) -> list[tuple[Group, Faculty]]:  
 return [(g, c)  
 for g in groups  
 for c in faculties  
 if (g.faculty\_id == c.id and g.name.startswith("ИУ"))]  
  
  
def task2(faculties: list[Faculty], groups: list[Group]) -> dict[str, int]:  
 data = {}  
 for faculty in faculties:  
 faculty\_students = [g.students  
 for g in groups  
 if (g.faculty\_id == faculty.id)]  
 if faculty\_students:  
 data[faculty.title] = max(faculty\_students)  
 return data  
  
  
def task3(faculties: list[Faculty], groups: list[Group], groups\_faculties: list[GroupFaculty]) -> list[tuple[Group, Faculty]]:  
 data = [(g, f)  
 for gf in groups\_faculties  
 for g in groups  
 for f in faculties  
 if gf.group\_id == g.id and gf.faculty\_id == f.id]  
  
 data.sort(key=lambda x: x[0].name)  
 return data  
  
  
def main():  
 faculties = [  
 Faculty(1, "Информатика и системы управления"),  
 Faculty(2, "Инженерный бизнес и менеджмент"),  
 Faculty(3, "Робототехника и комплексная автоматизация"),  
 Faculty(4, "Фундаментальные науки"),  
 Faculty(5, "Энергомашиностроение")  
 ]  
  
 groups = [  
 Group(1, "ИУ5ц", 5, 1, 1, 28),  
 Group(2, "ИУ6ц", 5, 2, 1, 24),  
 Group(3, "ИУ7", 3, 3, 1, 29),  
 Group(4, "ИБМ7", 3, 4, 2, 27),  
 Group(5, "РК6", 3, 5, 3, 28),  
 Group(6, "РК9", 3, 1, 3, 27),  
 Group(7, "ФН12", 3, 2, 4, 21),  
 Group(8, "ФН12", 3, 3, 4, 26),  
 Group(9, "Э9ц", 5, 4, 5, 24)  
 ]  
  
 groups\_faculties = [  
 GroupFaculty(1, 1),  
 GroupFaculty(2, 1),  
 GroupFaculty(3, 1),  
 GroupFaculty(4, 2),  
 GroupFaculty(5, 3),  
 GroupFaculty(6, 3),  
 GroupFaculty(7, 4),  
 GroupFaculty(8, 4),  
 GroupFaculty(9, 5)  
 ]  
  
 # Вывод результатов  
 print("Запрос № 1")  
 for (g, c) in task1(faculties, groups):  
 print(g.name, c.title)  
 print()  
  
 print("Запрос № 2")  
 data\_items = list(task2(faculties, groups).items())  
 data\_items.sort(key=lambda x: x[1])  
 for (faculty, max\_students) in data\_items:  
 print(faculty, max\_students)  
 print()  
  
 print("Запрос № 3")  
 for (group, course) in task3(faculties, groups, groups\_faculties):  
 print(group.name, course.title)  
 print()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**tests.py**

import unittest  
from main import Group, Faculty, GroupFaculty, task1, task2, task3  
  
  
class TestTasks(unittest.TestCase):  
  
 def setUp(self):  
 self.faculties = [  
 Faculty(1, "Информатика и системы управления"),  
 Faculty(2, "Инженерный бизнес и менеджмент"),  
 Faculty(3, "Робототехника и комплексная автоматизация"),  
 Faculty(4, "Фундаментальные науки"),  
 Faculty(5, "Энергомашиностроение")  
 ]  
  
 self.groups = [  
 Group(1, "ИУ5ц", 5, 1, 1, 28),  
 Group(2, "ИУ6ц", 5, 2, 1, 24),  
 Group(3, "ИУ7", 3, 3, 1, 29),  
 Group(4, "ИБМ7", 3, 4, 2, 27),  
 Group(5, "РК6", 3, 5, 3, 28),  
 Group(6, "РК9", 3, 1, 3, 27),  
 Group(7, "ФН12", 3, 2, 4, 21),  
 Group(8, "ФН12", 3, 3, 4, 26),  
 Group(9, "Э9ц", 5, 4, 5, 24)  
 ]  
  
 self.groups\_faculties = [  
 GroupFaculty(1, 1),  
 GroupFaculty(2, 1),  
 GroupFaculty(3, 1),  
 GroupFaculty(4, 2),  
 GroupFaculty(5, 3),  
 GroupFaculty(6, 3),  
 GroupFaculty(7, 4),  
 GroupFaculty(8, 4),  
 GroupFaculty(9, 5)  
 ]  
  
 def test\_task1(self):  
 result = task1(self.faculties, self.groups)  
 self.assertEqual(len(result), 3)  
 self.assertEqual(result[0][0].name, "ИУ5ц-51")  
 self.assertEqual(result[0][1].title, "Информатика и системы управления")  
  
 def test\_task2(self):  
 result = task2(self.faculties, self.groups)  
 self.assertEqual(result["Информатика и системы управления"], 29)  
 self.assertEqual(result["Инженерный бизнес и менеджмент"], 27)  
 self.assertEqual(result["Робототехника и комплексная автоматизация"], 28)  
 self.assertEqual(result["Фундаментальные науки"], 26)  
 self.assertEqual(result["Энергомашиностроение"], 24)  
  
 def test\_task3(self):  
 result = task3(self.faculties, self.groups, self.groups\_faculties)  
 self.assertEqual(len(result), 9)  
 self.assertEqual(result[0][0].name, "ИБМ7-34")  
 self.assertEqual(result[0][1].title, "Инженерный бизнес и менеджмент")  
 self.assertEqual(result[8][0].name, "Э9ц-54")  
 self.assertEqual(result[8][1].title, "Энергомашиностроение")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

**Результат работы программы**

