**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: B

Вариант предметной области: 19

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Нагапетян Валерий |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 г.

**Вариант запросов B.**

# «Деталь» и «Производитель» связаны соотношением один-ко-многим. Выведите список всех деталей, которые начинаются с «Тормоз» и названия их производителей.

# «Деталь» и «Производитель» связаны соотношением один-ко-многим. Выведите список производителей с минимальной стоимостью деталей, отсортированный их по минимальной стоимости.

# «Деталь» и «Производитель» связаны соотношением многие-ко-многим. Выведите список всех связанных деталей и производителей, отсортированный по производителям, сортировка деталей по цене.

# **Код программы**

from typing import Union, NoReturn  
  
  
class Component:  
 *"""Класс Деталь"""* def \_\_init\_\_(self, id: int, name: str, price: Union[int, float], fabric\_id: int):  
 self.\_\_id = id  
 self.\_\_name = name  
 self.\_\_price = price  
 self.\_\_fabric\_id = fabric\_id  
  
 @property  
 def id(self) -> int:  
 return self.\_\_id  
  
 @property  
 def name(self) -> str:  
 return self.\_\_name  
  
 @property  
 def price(self) -> Union[int, float]:  
 return self.\_\_price  
  
 @property  
 def fabric\_id(self) -> int:  
 return self.\_\_fabric\_id  
  
  
class Fabric:  
 *"""Класс Производитель"""* def \_\_init\_\_(self, id: int, name: str):  
 self.\_\_id = id  
 self.\_\_name = name  
  
 @property  
 def id(self) -> int:  
 return self.\_\_id  
  
 @property  
 def name(self) -> str:  
 return self.\_\_name  
  
  
class FabricComponent:  
 *"""Класс детали производителя"""* def \_\_init\_\_(self, fabric\_id: int, component\_id: int):  
 self.\_\_fabric\_id = fabric\_id  
 self.\_\_component\_id = component\_id  
  
 @property  
 def fabric\_id(self) -> int:  
 return self.\_\_fabric\_id  
  
 @property  
 def component\_id(self) -> int:  
 return self.\_\_component\_id  
  
  
def request1(components: list[Component], fabrics: list[Fabric]) -> NoReturn:  
 print(  
 'Запрос №1. Список деталей, которые начинаются с "Тормоз", и их производителей.'  
 )  
 response = [  
 (c, f)  
 for c in components  
 for f in fabrics  
 if c.fabric\_id == f.id and c.name.startswith("Тормоз")  
 ]  
 for c, f in response:  
 print(f" Деталь: {c.name} | Производитель: {f.name}")  
 print()  
  
  
def request2(components: list[Component], fabrics: list[Fabric]) -> NoReturn:  
 print(  
 "Запрос №2. Список производителей с минимальной стоимостью деталей."  
 )  
 response = {}  
 for fabric in fabrics:  
 minimal\_price = min([c.price for c in components if c.fabric\_id == fabric.id])  
 response[fabric.name] = minimal\_price  
  
 response\_sorted = dict(sorted(response.items(), key=lambda item: item[1]))  
 for fabric\_name, minimal\_price in response\_sorted.items():  
 print(f" Производитель: {fabric\_name} | Минимальная стоимость детали: {minimal\_price}")  
 print()  
  
  
def request3(components: list[Component], fabrics: list[Fabric], fabric\_components: list[FabricComponent]) -> NoReturn:  
 print(  
 "Запрос №3. Список всех связанных деталей и производителей, отсортированный по производителям. Сортировка деталей по цене."  
 )  
 response = [  
 (f, c)  
 for fc in fabric\_components  
 for f in fabrics  
 for c in components  
 if fc.fabric\_id == f.id and fc.component\_id == c.id  
 ]  
 response.sort(key=lambda item: (item[0].name, item[1].price))  
  
 for fabric, component in response:  
 print(f" Производитель: {fabric.name} | Деталь: {component.name}")  
 print()  
  
  
def main() -> NoReturn:  
 components = [  
 Component(1, "Тормозные колодки", 12000, 1),  
 Component(2, "Фары", 5000, 1),  
 Component(3, "Заднее крыло", 10000, 2),  
 Component(4, "Генератор", 15000, 3),  
 Component(5, "Аккумулятор", 8000, 3),  
 Component(6, "Тормозные диски", 9000, 4),  
 Component(7, "Дворники", 3000, 4)  
 ]  
  
 fabrics = [  
 Fabric(1, "АВТОВАЗ"),  
 Fabric(2, "УАЗ"),  
 Fabric(3, "КАМАЗ"),  
 Fabric(4, "УВЗ"),  
 ]  
  
 fabric\_component = [  
 FabricComponent(1, 1),  
 FabricComponent(1, 2),  
 FabricComponent(2, 3),  
 FabricComponent(3, 4),  
 FabricComponent(3, 5),  
 FabricComponent(4, 6),  
 FabricComponent(4, 7)  
 ]  
  
 request1(components, fabrics)  
 request2(components, fabrics)  
 request3(components, fabrics, fabric\_component)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# **Результат**