Отчет

По рубежному контролю – 1

Дисциплина «Парадигмы и конструкции языков программирования»

Студент: Артёмова Дарья

Группа: ИБМЗ – 34Б

Вариант предметной области - 3
Класс 1: “Водитель”
Класс 2: “Автопарк”

Текст программы:

```python
from operator import itemgetter

class Driver:
 """Водитель"""
 def __init__(self, id, name, salary, car_id):
   self.id = id
   self.name = name
   self.salary = salary
   self.car_id = car_id

class CarPark:
 """Автопарк"""
 def __init__(self, id, location):
   self.id = id
   self.location = location

class DriverCar:
 """'Водители автопарка' для реализации связи многие-ко-многим"""
 def __init__(self, car_id, driver_id):
   self.car_id = car_id
   self.driver_id = driver_id

# Автопарки
car_parks = [
 CarPark(1, 'Центральный автопарк'),
 CarPark(2, 'Западный автопарк'),
 CarPark(3, 'Восточный автопарк'),
]

# Водители
drivers = [
 Driver(1, 'Иванов', 50000, 1),
 Driver(2, 'Петров', 60000, 2),
 Driver(3, 'Сидоров', 55000, 3),
 Driver(4, 'Козлов', 58000, 3),
 Driver(5, 'Смирнов', 52000, 3),
]

# Связи многие-ко-многим
driver_car_relations = [
 DriverCar(1, 1),
 DriverCar(2, 2),
 DriverCar(3, 3),
 DriverCar(3, 4),
 DriverCar(3, 5),
]
```

```python
def main():
 """Основная функция"""

 # Соединение данных один-ко-многим
 one_to_many = [(d.name, d.salary, c.location)
         for c in car_parks
         for d in drivers
         if d.car_id == c.id]

 # Соединение данных многие-ко-многим
 many_to_many_temp = [(c.location, dc.car_id, dc.driver_id)
            for c in car_parks
            for dc in driver_car_relations
            if c.id == dc.car_id]

 many_to_many = [(d.name, d.salary, park_location)
         for park_location, car_id, driver_id in many_to_many_temp
         for d in drivers if d.id == driver_id]

 print('Задание A1')
 res_11 = sorted(one_to_many, key=itemgetter(2))
 print(res_11)

 print('\nЗадание A2')
 res_12_unsorted = []
 for c in car_parks:
   c_drivers = list(filter(lambda i: i[2] == c.location, one_to_many))
   if len(c_drivers) > 0:
     c_salaries = [sal for _, sal, _ in c_drivers]
     c_salaries_sum = sum(c_salaries)
     res_12_unsorted.append((c.location, c_salaries_sum))

 res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
 print(res_12)

 print('\nЗадание A3')
 res_13 = {}
 for c in car_parks:
   if 'автопарк' in c.location.lower():
     c_drivers = list(filter(lambda i: i[2] == c.location, many_to_many))
     c_driver_names = [x for x, _, _ in c_drivers]
     res_13[c.location] = c_driver_names

 print(res_13)

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

Результат программы:

Задание A1

[('Сидоров', 55000, 'Восточный автопарк'), ('Козлов', 58000, 'Восточный автопарк'), ('Смирнов', 52000, 'Восточный автопарк'), ('Петров', 60000, 'Западный автопарк'), ('Иванов', 50000, 'Центральный автопарк')]

Задание A2

[('Восточный автопарк', 165000), ('Западный автопарк', 60000), ('Центральный автопарк', 50000)]

Задание A3

{'Центральный автопарк': ['Иванов'], 'Западный автопарк': ['Петров'], 'Восточный автопарк': ['Сидоров', 'Козлов', 'Смирнов']}