Текст Программы

entities.py:

```python
class Driver:
   def __init__(self, id_: int, name: str, age: int, carpark_id: int):
       self.id = id_
       self.name = name
       self.age = age
       self.carpark_id = carpark_id


class CarPark:
   def __init__(self, id_: int, name: str):
       self.id = id_
       self.name = name


class DriverCarPark:
   """
   класс для реализации связи многие-ко-многим
   """

   def __init__(self, driver_id: int, carpark_id: int):
       self.driver_id = driver_id
       self.carpark_id = carpark_id
```

data.py

```python
from entities import CarPark, Driver, DriverCarPark

carpark_data = [
   CarPark(1, "АвтоСтоп"),
   CarPark(2, "Перевозки24"),
   CarPark(3, "Абобавоз"),
   CarPark(4, "ГаражСити"),
   CarPark(5, "ЗдесьМашины"),
   CarPark(6, "Cars&Wheels")
]

driver_data = [
   Driver(1, "Олег Шариков", 22, 6),
   Driver(2, "Фёдор Петухов", 55, 3),
   Driver(3, "Марфа Карбанчикова", 34, 2),
   Driver(4, "Степан Черепахин", 36, 1),
   Driver(5, "Агафья Мышкина", 33, 5),
   Driver(6, "Прокофий Ёжов", 62, 3),
   Driver(7, "Галатея Холодильникова", 21, 4),
   Driver(8, "Пафнутий Тыковский", 28, 5)
]

driver_carpark_data = [
   DriverCarPark(1, 2),
   DriverCarPark(1, 4),
   DriverCarPark(1, 6),
   DriverCarPark(2, 5),
   DriverCarPark(2, 3),
   DriverCarPark(3, 1),
   DriverCarPark(3, 2),
   DriverCarPark(4, 1),
   DriverCarPark(4, 5),
   DriverCarPark(5, 1),
   DriverCarPark(5, 2),
   DriverCarPark(5, 5),
   DriverCarPark(6, 3),
   DriverCarPark(7, 4),
   DriverCarPark(7, 6),
   DriverCarPark(8, 5)
]
```

main.py

```
from statistics import mean
import data


def query1(drivers: list, carparks: list) -> list:
    '''возвращает имена всех водителей с фамилией на "ов" и их автопарк'''
    return [(driver.name, carpark.name)
            for driver in drivers
            for carpark in carparks
            if driver.carpark_id == carpark.id
            and driver.name[-2:] == 'ов']
```

```
def carpark_ages(carpark_id: int, drivers: list) -> list:
   return [driver.age
           for driver in drivers
           if driver.carpark_id == carpark_id]


def query2(drivers: list, carparks: list) -> list:
   '''возвращает средний возраст по каждому отделу'''
   return [(carpark.name, mean(carpark_ages(carpark.id, drivers)))
           for carpark in carparks]


def carpark_drivers(carpark_id: int, drivers: list, driver_carpark_data: list) -> list:
   return [driver.name
           for driver in drivers
           for driver_carpark in driver_carpark_data
           if driver_carpark.driver_id == driver.id
          and driver_carpark.carpark_id == carpark_id]


def query3(drivers: list, carparks: list, driver_carpark_data: list) -> list:
   '''возвращает список водителей каждого автопарка, название которого начинается на "А"'''
   return [(carpark.name, carpark_drivers(carpark.id, drivers, driver_carpark_data))
          for carpark in carparks
          if carpark.name[0] == 'А']


def main():
   print('Запрос Д1')
   print(query1(data.driver_data, data.carpark_data))

   print('Запрос Д2')
   print(query2(data.driver_data, data.carpark_data))

   print('Запрос Д3')
   print(query3(data.driver_data, data.carpark_data, data.driver_carpark_data))


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

tests.py

```
import unittest
from collections import Counter
```

```
import data
from main import query1, query2, query3, carpark_ages, carpark_drivers


class TestFunctions(unittest.TestCase):
    def setUp(self):
        self.data = data

    def test_query1(self):
        want = [('Олег Шариков', 'Cars&Wheels'), ('Фёдор Петухов', 'Абобавоз'), ('Прокофий Ёжов',
'Абобавоз')]
        actual = query1(data.driver_data, data.carpark_data)
        self.assertCountEqual(want, actual)

    def test_carpark_ages(self):
        want = [[36], [34], [55, 62], [21], [33, 28], [22]]
        actual = [carpark_ages(i, data.driver_data)
                  for i in Counter([carpark.id for carpark in data.carpark_data]).keys()]
        self.assertCountEqual(want, actual)

    def test_query2(self):
        want = [('АвтоСтоп', 36), ('Перевозки24', 34), ('Абобавоз', 58.5), ('ГаражСити', 21),
('ЗдесьМашины', 30.5),
                ('Cars&Wheels', 22)]
        actual = query2(data.driver_data, data.carpark_data)

        self.assertCountEqual(want, actual)

    def test_catpark_drivers(self):
        want = [['Марфа Карбанчикова', 'Степан Черепахин', 'Агафья Мышкина'],
                ['Олег Шариков', 'Марфа Карбанчикова', 'Агафья Мышкина'],
                ['Фёдор Петухов', 'Прокофий Ёжов'], ['Олег Шариков', 'Галатея Холодильникова'],
                ['Фёдор Петухов', 'Степан Черепахин', 'Агафья Мышкина', 'Пафнутий Тыковский'],
                ['Олег Шариков', 'Галатея Холодильникова']]
        actual = [carpark_drivers(i, data.driver_data, data.driver_carpark_data)
                  for i in Counter([carpark.id for carpark in data.carpark_data]).keys()]
        self.assertCountEqual(want, actual)

    def test_query3(self):
        want = [('АвтоСтоп', ['Марфа Карбанчикова', 'Степан Черепахин', 'Агафья Мышкина']),
                ('Абобавоз', ['Фёдор Петухов', 'Прокофий Ёжов'])]
        actual = query3(data.driver_data, data.carpark_data, data.driver_carpark_data)

        self.assertCountEqual(want, actual)
```

Результаты выполнения

```
Запрос Д1
[('Олег Шариков', 'Cars&Wheels'), ('Фёдор Петухов', 'Абобавоз'), ('Прокофий Ёжов', 'Абобавоз')]
Запрос Д2
[('АвтоСтоп', 36), ('Перевозки24', 34), ('Абобавоз', 58.5), ('ГаражСити', 21), ('ЗдесьМашины', 30.5),
('Cars&Wheels', 22)]
Запрос Д3
[('АвтоСтоп', ['Марфа Карбанчикова', 'Агафья Мышкина', 'Прокофий Ёжов']), ('Абобавоз',
['Галатея Холодильникова'])]
```

```
 python -m unittest tests.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.000s

OK
```