Баршин Сергей ИУ5-31б Вариант 3

In [2]:
```python
import unittest


# Водитель
class Driver:
    def __init__(self, driver_id, name, experience, fleet_id):
        self.driver_id = driver_id
        self.name = name
        self.experience = experience
        self.fleet_id = fleet_id


# Автопарк
class Fleet:
    def __init__(self, fleet_id, name):
        self.fleet_id = fleet_id
        self.name = name  # Название автопарка


# многие-ко-многим
class DriverFleet:
    def __init__(self, driver_id, fleet_id):
        self.driver_id = driver_id
        self.fleet_id = fleet_id


# Данные
drivers = [
    Driver(1, "Иванов", 5, 1),
    Driver(2, "Петров", 10, 1),
    Driver(3, "Сидоров", 3, 2),
    Driver(4, "Кузнецов", 8, 3),
    Driver(5, "Алексеев", 6, 2),
]

fleets = [
    Fleet(1, "Центральный автопарк"),
    Fleet(2, "Южный автопарк"),
    Fleet(3, "Северный автопарк"),
]

driver_fleets = [
    DriverFleet(1, 1),
    DriverFleet(2, 1),
    DriverFleet(3, 2),
    DriverFleet(4, 3),
    DriverFleet(5, 2),
]

# Функции


def list_drivers_by_fleet():
    sorted_fleets = sorted(fleets, key=lambda f: f.name)
    result = {}
    for fleet in sorted_fleets:
        fleet_drivers = [d.name for d in drivers if d.fleet_id == fleet.fleet_id]
        result[fleet.name] = fleet_drivers
    return result


def list_fleets_by_experience():
    fleet_experience = {fleet.fleet_id: 0 for fleet in fleets}
    for driver in drivers:
        fleet_experience[driver.fleet_id] += driver.experience
    sorted_fleet_experience = sorted(
        fleet_experience.items(), key=lambda item: item[1], reverse=True
    )
    result = {}
    for fleet_id, experience in sorted_fleet_experience:
        fleet_name = next(f.name for f in fleets if f.fleet_id == fleet_id)
        result[fleet_name] = experience
    return result


def list_fleets_with_drivers_containing_word(word="автопарк"):
    fleets_with_word = [fleet for fleet in fleets if word in fleet.name.lower()]
    result = {}
    for fleet in fleets_with_word:
        associated_drivers = [
            d.name
            for df in driver_fleets
            if df.fleet_id == fleet.fleet_id
            for d in drivers
            if d.driver_id == df.driver_id
        ]
        result[fleet.name] = associated_drivers
    return result


# Тесты
class TestFleetFunctions(unittest.TestCase):
    def test_list_drivers_by_fleet(self):
        result = list_drivers_by_fleet()
        expected = {
            "Центральный автопарк": ["Иванов", "Петров"],
            "Южный автопарк": ["Сидоров", "Алексеев"],
            "Северный автопарк": ["Кузнецов"],
        }
        self.assertEqual(result, expected)

    def test_list_fleets_by_experience(self):
        result = list_fleets_by_experience()
        expected = {
            "Центральный автопарк": 15,
            "Южный автопарк": 9,
            "Северный автопарк": 8,
        }
        self.assertEqual(result, expected)

    def test_list_fleets_with_drivers_containing_word(self):
        result = list_fleets_with_drivers_containing_word("автопарк")
        expected = {
            "Центральный автопарк": ["Иванов", "Петров"],
            "Южный автопарк": ["Сидоров", "Алексеев"],
            "Северный автопарк": ["Кузнецов"],
        }
        self.assertEqual(result, expected)


if __name__ == "__main__":
    unittest.main(argv=[''], exit=False)
```

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
```