Вариант В3

Текст основной программы (driver_fleet_management.py):

FleetDrivers (bottom)

```python
class Driver:
    def __init__(self, driver_id, surname, salary, fleet_id):
        self.driver_id = driver_id
        self.surname = surname
        self.salary = salary
        self.fleet_id = fleet_id

class Fleet:
    def __init__(self, fleet_id, name):
        self.fleet_id = fleet_id
        self.name = name

class FleetDrivers:
    def __init__(self, driver_id, fleet_id):
        self.driver_id = driver_id
        self.fleet_id = fleet_id


drivers = [
    Driver(1, "Ivanov", 50000, 1),
    Driver(2, "Alexeev", 60000, 1),
    Driver(3, "Alekseeva", 55000, 2),
    Driver(4, "Abramov", 52000, 2),
    Driver(5, "Petrov", 58000, 2)
]

fleets = [
    Fleet(1, "Fleet 1"),
    Fleet(2, "Fleet 2")
]

fleet_drivers = [
    FleetDrivers(1, 1),
    FleetDrivers(2, 1),
    FleetDrivers(3, 2),
    FleetDrivers(4, 2),
    FleetDrivers(5, 2)
]


def A1(drivers,fleets):
    drivers_with_a = [(driver.surname, fleet.name) for driver in drivers for fleet in fleets if driver.surname.startswith("A")]
    return (drivers_with_a)

def A2(drivers,fleets):
    min_salary_per_fleet = {fleet.name: min(driver.salary for driver in drivers if driver.fleet_id == fleet.fleet_id) for fleet in fleets}
    sorted_fleets_by_min_salary = [(fleet, min_salary) for fleet, min_salary in sorted(min_salary_per_fleet.items(), key=lambda x: x[1])]
    return(sorted_fleets_by_min_salary)


def A3(drivers,fleets,fleet_drivers):
    fleet_drivers_sorted_by_driver = sorted(fleet_drivers, key=lambda x: drivers[x.driver_id-1].surname)
    drivers_fleets = [(drivers[fd.driver_id-1].surname, fleets[fd.fleet_id-1].name) for fd in fleet_drivers_sorted_by_driver]
    return(drivers_fleets)

#A1(drivers,fleets)
#A2(drivers,fleets)
#A3(drivers,fleets,fleet_drivers)
```

Текст программы с кодом тестов (test_driver_fleet_management.py):

(top)

```python

import unittest
from driver_fleet_management import *

class TestDriverFleetManagement(unittest.TestCase):
    def setUp(self):
        self.drivers = [
            Driver(1, "Ivanov", 50000, 1),
            Driver(2, "Alexeev", 60000, 1),
            Driver(3, "Alekseeva", 55000, 2),
            Driver(4, "Abramov", 52000, 2),
            Driver(5, "Petrov", 58000, 2)
        ]
        self.fleets = [
            Fleet(1, "Fleet 1"),
            Fleet(2, "Fleet 2")
        ]
        self.fleet_drivers = [
            FleetDrivers(1, 1),
            FleetDrivers(2, 1),
            FleetDrivers(3, 2),
            FleetDrivers(4, 2),
            FleetDrivers(5, 2)
        ]

    def test_get_drivers_with_surname_starting_with(self):
        result = A1(self.drivers, self.fleets)
        expected_result = [('Alexeev', 'Fleet 1'), ('Alexeev', 'Fleet 2'), ('Alekseeva', 'Fleet 1'), ('Alekseeva', 'Fleet 2'), ('Abramov', 'Fleet 1'), ('Abramov', 'Fleet 2')]
        self.assertEqual(result, expected_result)

    def test_get_fleets_sorted_by_min_salary(self):
        result = A2(self.drivers, self.fleets)
        expected_result = [('Fleet 1', 50000), ('Fleet 2', 52000)]
        self.assertEqual(result, expected_result)

    def test_get_drivers_and_fleets_sorted_by_driver(self):
        result = A3(self.drivers, self.fleets,self.fleet_drivers)
        expected_result = [('Abramov', 'Fleet 2'), ('Alekseeva', 'Fleet 2'), ('Alexeev', 'Fleet 1'), ('Ivanov', 'Fleet 1'), ('Petrov', 'Fleet 2')]
        self.assertEqual(result, expected_result)

if __name__ == '__main__':
    unittest.main()

```

Результат работы тестового файла (без ошибок):

```
Debug I/O    Python Shell

Commands execute without debug.  Use arrow keys for history.                    Options

   Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)]
   Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate test_driver_fleet_management.py]
   ...
   ----------------------------------------------------------------------
   Ran 3 tests in 0.000s

   OK
>>> |
```

Результат работы с заведомо ошибочным ответом:

Было:

```
def test_get_drivers_with_surname_starting_with(self):
    result = A1(self.drivers, self.fleets)
    expected_result = [('Alexeev', 'Fleet 1'), ('Alexeev', 'Fleet 2'), ('Alekseeva', 'Fleet 1'), ('Alekseeva', 'Fleet 2'), ('Abramov', 'Fleet 1'), ('Abramov', 'Fleet 2')]
    self.assertEqual(result, expected_result)
```

Стало:

```
def test_get_drivers_with_surname_starting_with(self):
    result = A1(self.drivers, self.fleets)
    expected_result = [('Alexeev', 'Fleet 1'), ('Alexeev', 'Fleet11 2'), ('Alekseeva', 'Fleet 1'), ('Alekseeva', 'Fleet 2'), ('Abramov', 'Fleet 1'), ('Abramov', 'Fleet 2')]
    self.assertEqual(result, expected_result)
```

Debug I/O Python Shell

Commands execute without debug. Use arrow keys for history. Options

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate test_driver_fleet_management.py]
.F.
======================================================================
FAIL: test_get_drivers_with_surname_starting_with (__main__.TestDriverFleetManagement)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "x-wingide-python-shell://132700032/2", line 29, in test_get_drivers_with_surname_starting_with
AssertionError: Lists differ: [('Al[33 chars]Fleet 2'), ('Alekseeva', 'Fleet 1'), ('Aleksee[59 chars] 2')] != [('Al[33 chars]Fleet11 2'), ('Alekseeva', 'Fleet 1'), ('Aleks[61 chars] 2')]

First differing element 1:
('Alexeev', 'Fleet 2')
('Alexeev', 'Fleet11 2')

  [('Alexeev', 'Fleet 1'),
-  ('Alexeev', 'Fleet 2'),
+  ('Alexeev', 'Fleet11 2'),
?                    ++

   ('Alekseeva', 'Fleet 1'),
   ('Alekseeva', 'Fleet 2'),
   ('Abramov', 'Fleet 1'),
   ('Abramov', 'Fleet 2')]

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (failures=1)
>>>
```