Отчет

По рубежному контролю – 1

Дисциплина «Парадигмы и конструкции языков программирования»

Студент: Артёмова Дарья

Группа: ИБМ3 – 34Б

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

За счет рефакторинга основного кода из РК-1, он разделился на 4 части: 3 части основной преобразованный код, 4-ая часть проведение тестирование кода через модуль unittest.

1. *models.py*

Файл с определением классов “Driver”, “CarPark”, “DriverCar”.

```
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
```

2. *services.py*

Файл содержит функции для обработки данных, такие как соединение один-ко-многим и многие-ко-многим, а также подсчет зарплат.

```
from operator import itemgetter
from models import Driver, CarPark, DriverCar

def connect_one_to_many(drivers, car_parks):
    """Соединение данных один-ко-многим"""
    return [(d.name, d.salary, c.location)
            for c in car_parks
            for d in drivers
            if d.car_id == c.id]

def connect_many_to_many(drivers, car_parks, driver_car_relations):
    """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]

    return [(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]

def calculate_salaries(car_parks, one_to_many):
    """Подсчет суммарных зарплат водителей по автопаркам"""
    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))

    return sorted(res_12_unsorted, key=itemgetter(1), reverse=True)

def get_driver_names_by_park(car_parks, many_to_many):
    """Получение имен водителей по автопаркам"""
    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
    return res_13
```

3. *main.py*

Файл содержит основную функцию, которая собирает данные и выводит результаты.

```
from models import Driver, CarPark, DriverCar
from services import connect_one_to_many, connect_many_to_many, calculate_salaries,
get_driver_names_by_park

# 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),
]

def main():
    """Основная функция"""
    one_to_many = connect_one_to_many(drivers, car_parks)
    many_to_many = connect_many_to_many(drivers, car_parks, driver_car_relations)
```

```
    print('Задание А1')
    res_11 = sorted(one_to_many, key=itemgetter(2))
    print(res_11)

    print('\nЗадание А2')
    res_12 = calculate_salaries(car_parks, one_to_many)
    print(res_12)

    print('\nЗадание А3')
    res_13 = get_driver_names_by_park(car_parks, many_to_many)
    print(res_13)

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

4. *tests.py*

Файл, содержащий модульные тесты для проверки функциональности

```
import unittest
from models import Driver, CarPark, DriverCar
from services import connect_one_to_many, connect_many_to_many, calculate_salaries,
get_driver_names_by_park

class TestDriverCarFunctions(unittest.TestCase):

    def setUp(self):
        self.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),
        ]
        self.car_parks = [
            CarPark(1, 'Центральный автопарк'),
            CarPark(2, 'Западный автопарк'),
            CarPark(3, 'Восточный автопарк'),
        ]
        self.driver_car_relations = [
            DriverCar(1, 1),
            DriverCar(2, 2),
            DriverCar(3, 3),
            DriverCar(3, 4),
            DriverCar(3, 5),
        ]

    def test_connect_one_to_many(self):
        result = connect_one_to_many(self.drivers, self.car_parks)
        expected = [
            ('Иванов', 50000, 'Центральный автопарк'),
            ('Петров', 60000, 'Западный автопарк'),
            ('Сидоров', 55000, 'Восточный автопарк'),
            ('Козлов', 58000, 'Восточный автопарк'),
            ('Смирнов', 52000, 'Восточный автопарк'),
        ]
        self.assertEqual(sorted(result), sorted(expected))

    def test_calculate_salaries(self):
        one_to_many = connect_one_to_many(self.drivers, self.car_parks)
        result = calculate_salaries(self.car_parks, one_to_many)
        expected = [
            ('Восточный автопарк', 165000),
            ('Западный автопарк', 60000),
            ('Центральный автопарк', 50000),
        ]
        self.assertEqual(result, expected)
```

```
    def test_get_driver_names_by_park(self):
        one_to_many = connect_one_to_many(self.drivers, self.car_parks)
        many_to_many = connect_many_to_many(self.drivers, self.car_parks,
self.driver_car_relations)
        result = get_driver_names_by_park(self.car_parks, many_to_many)
        expected = {
            'Центральный автопарк': ['Иванов'],
            'Западный автопарк': ['Петров'],
            'Восточный автопарк': ['Сидоров', 'Козлов', 'Смирнов'],
        }
        self.assertEqual(result, expected)

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

Вывод тестов:

```
Testing started at 12:34 ...
Launching pytest with arguments C:\Users\26276\PycharmProjects\pythonProject1\tests.py --no-

============================= test session starts =============================
collecting ... collected 3 items

tests.py::TestDriverCarFunctions::test_calculate_salaries PASSED         [ 33%]
tests.py::TestDriverCarFunctions::test_connect_one_to_many PASSED        [ 66%]
tests.py::TestDriverCarFunctions::test_get_driver_names_by_park PASSED   [100%]

============================== 3 passed in 0.04s ==============================

Process finished with exit code 0
```

```
(venv) PS C:\Users\26276\PycharmProjects\pythonProject1> python -m unittest tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```