Московский государственный технический университет им. Н.Э. Баумана

Факультет Радиотехнический
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по Рубежному контролю №2
Вариант №1

Выполнил:
студент группы РТ5-31Б:
Аверьянов Александр
Дмитриевич
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич
Подпись и дата:

Москва, 2025 г.

Постановка задачи

Условия рубежного контроля №2 по курсу ПиК ЯП

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Текст программы

main.py

```
from models import create_test_data
from queries import запрос_1_отделы_со_словом_отдел,
запрос_2_средняя_зарплата_по_отделам, запрос_3_сотрудники_на_а

def main():
    отделы, сотрудники, _ = create_test_data()

    print("=== Запрос 1 ===")
    for item in запрос_1_отделы_со_словом_отдел(отделы, сотрудники):
        print(f"  {item['отдел']}:")
        for s in item["сотрудники"]:
            print(f"    {s['фамилия']} (ID: {s['id']}, Зарплата: {s['зарплата']})")

    print("\n=== Запрос 2 ===")
    for название, средняя in запрос_2_средняя_зарплата_по_отделам(отделы,
сотрудники):
        print(f"  {название}: {средняя}")

    print("\n=== Запрос 3 ===")
    for s in запрос_3_сотрудники_на_а(отделы, сотрудники):
        print(f"  {s['фамилия']} (Отдел: {s['отдел']})")

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

models.py

```
class Сотрудник:
    def __init__(self, id_сотрудника, фамилия, зарплата, id_отдела):
        self.id_сотрудника = id_сотрудника
        self.фамилия = фамилия
        self.зарплата = зарплата
        self.id_отдела = id_отдела

class Отдел:
    def __init__(self, id_отдела, наименование):
        self.id_отдела = id_отдела
        self.наименование = наименование

class СотрудникиОтдела:
    def __init__(self, id_сотрудника, id_отдела):
        self.id_сотрудника = id_сотрудника
        self.id_отдела = id_отдела

# Функция для инициализации тестовых данных
def create_test_data():
    отделы = [
        Отдел(1, "Отдел разработки"),
        Отдел(2, "Отдел маркетинга"),
        Отдел(3, "Отдел продаж"),
        Отдел(4, "Отдел поддержки"),
        Отдел(5, "Отдел аналитики")
    ]

    сотрудники = [
        Сотрудник(1, "Иванов", 75000.0, 1),
        Сотрудник(2, "Петров", 82000.0, 1),
        Сотрудник(3, "Сидоров", 68000.0, 2),
        Сотрудник(4, "Алексеев", 71000.0, 3),
        Сотрудник(5, "Андреева", 79000.0, 4),
```

```
        Сотрудник(6, "Белов", 65000.0, 5),
        Сотрудник(7, "Афанасьев", 85000.0, 1),
        Сотрудник(8, "Григорьева", 73000.0, 3)
    ]

    сотрудники_отдела = [
        СотрудникиОтдела(1, 1),
        СотрудникиОтдела(1, 3),
        СотрудникиОтдела(2, 1),
        СотрудникиОтдела(3, 2),
        СотрудникиОтдела(4, 3),
        СотрудникиОтдела(5, 4),
        СотрудникиОтдела(6, 5),
        СотрудникиОтдела(7, 1),
        СотрудникиОтдела(8, 3),
        СотрудникиОтдела(8, 5)
    ]

    return отделы, сотрудники, сотрудники_отдела
```

queries.py

```
def запрос_1_отделы_со_словом_отдел(отделы, сотрудники):
    отделы_с_словом = [отд for отд in отделы if "отдел" in
отд.наименование.lower()]
    результат = []
    for отдел in отделы_с_словом:
        сотрудники_в_отделе = [
            сотр for сотр in сотрудники if сотр.id_отдела == отдел.id_отдела
        ]
        результат.append({
            "отдел": отдел.наименование,
            "сотрудники": [
                {"фамилия": с.фамилия, "id": с.id_сотрудника, "зарплата":
с.зарплата}
```

```
                for c in сотрудники_в_отделе
            ]
        })
    return результат

def запрос_2_средняя_зарплата_по_отделам(отделы, сотрудники):
    результат = []
    for отдел in отделы:
        зарплаты = [c.зарплата for c in сотрудники if c.id_отдела ==
отдел.id_отдела]
        if зарплаты:
            средняя = round(sum(зарплаты) / len(зарплаты), 2)
            результат.append((отдел.наименование, средняя))
    результат.sort(key=lambda x: x[1], reverse=True)
    return результат

def запрос_3_сотрудники_на_а(отделы, сотрудники):
    сотрудники_а = [c for c in сотрудники if c.фамилия.startswith("А")]
    отделы_dict = {отд.id_отдела: отд.наименование for отд in отделы}
    результат = []
    for c in сотрудники_а:
        название_отдела = отделы_dict.get(c.id_отдела, "Неизвестный отдел")
        результат.append({"фамилия": c.фамилия, "отдел": название_отдела})
    return результат
```

test_queries.py

```
import unittest
from models import create_test_data
from queries import запрос_1_отделы_со_словом_отдел,
запрос_2_средняя_зарплата_по_отделам, запрос_3_сотрудники_на_а

class TestQueries(unittest.TestCase):

    def setUp(self):
```

```
        self.отделы, self.сотрудники, _ = create_test_data()

    def test_запрос_1_отделы_со_словом_отдел(self):
        результат = запрос_1_отделы_со_словом_отдел(self.отделы,
self.сотрудники)
        # Проверяем количество отделов
        self.assertEqual(len(результат), 5, "Должно быть 5 отделов")
        # Проверяем, что в каждом отделе есть сотрудники
        for item in результат:
            self.assertGreater(len(item["сотрудники"]), 0, f"В отделе {item['отдел']}
нет сотрудников")

    def test_запрос_2_средняя_зарплата_по_отделам(self):
        результат = запрос_2_средняя_зарплата_по_отделам(self.отделы,
self.сотрудники)
        # Проверяем, что список отсортирован по убыванию
        зарплаты = [средняя for _, средняя in результат]
        self.assertEqual(зарплаты, sorted(зарплаты, reverse=True))

        # Проверяем конкретное значение для Отдела разработки
        разработка = next((средняя for наим, средняя in результат if "разработки"
in наим), None)
        self.assertAlmostEqual(разработка, 80666.67, places=2, msg="Средняя
зарплата в Отделе разработки должна быть ~80666.67")

    def test_запрос_3_сотрудники_на_а(self):
        результат = запрос_3_сотрудники_на_а(self.отделы, self.сотрудники)
        фамилии = [s["фамилия"] for s in результат]
        # Должны быть Алексеев, Андреева, Афанасьев
        ожидаемые = {"Алексеев", "Андреева", "Афанасьев"}
        self.assertEqual(set(фамилии), ожидаемые, "Не все сотрудники на 'А'
найдены")
        # Проверим, что у Алексеева указан правильный отдел
        алексеев = next(s for s in результат if s["фамилия"] == "Алексеев")
        self.assertEqual(алексеев["отдел"], "Отдел продаж", "Отдел Алексеева
должен быть 'Отдел продаж'")
```

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

Анализ результатов

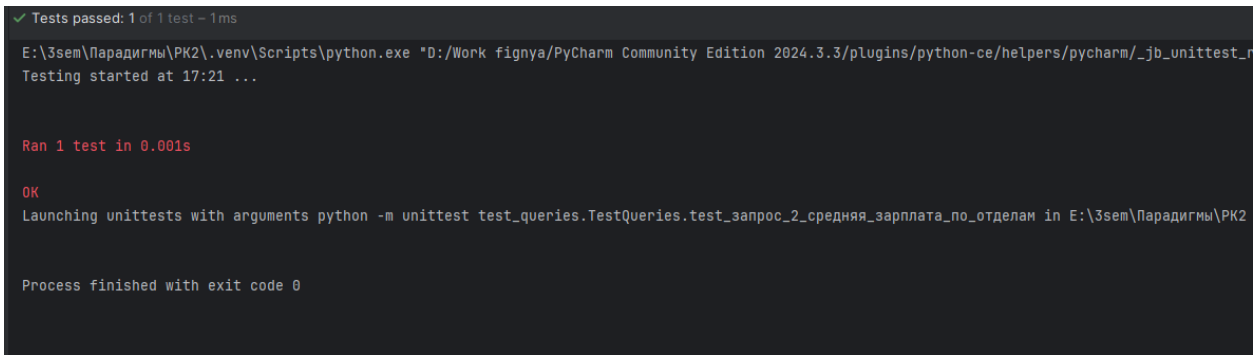
Входные данные:

- Отображены в терминале

Выходные данные:

```
E:\3sem\Парадигмы\РК2\.venv\Scripts\python.exe E:\3sem\Парадигмы\РК2\main.py
=== Запрос 1 ===
  Отдел разработки:
    Иванов (ID: 1, Зарплата: 75000.0)
    Петров (ID: 2, Зарплата: 82000.0)
    Афанасьев (ID: 7, Зарплата: 85000.0)
  Отдел маркетинга:
    Сидоров (ID: 3, Зарплата: 68000.0)
  Отдел продаж:
    Алексеев (ID: 4, Зарплата: 71000.0)
    Григорьева (ID: 8, Зарплата: 73000.0)
  Отдел поддержки:
    Андреева (ID: 5, Зарплата: 79000.0)
  Отдел аналитики:
    Белов (ID: 6, Зарплата: 65000.0)

=== Запрос 2 ===
  Отдел разработки: 80666.67
  Отдел поддержки: 79000.0
  Отдел продаж: 72000.0
  Отдел маркетинга: 68000.0
  Отдел аналитики: 65000.0

=== Запрос 3 ===
  Алексеев (Отдел: Отдел продаж)
  Андреева (Отдел: Отдел поддержки)
  Афанасьев (Отдел: Отдел разработки)
```

```
✓ Tests passed: 1 of 1 test – 0 ms
E:\3sem\Парадигмы\РК2\.venv\Scripts\python.exe "D:/Work fignya/PyCharm Community Edition 2024.3.3/plugins/python-ce/helpers/pycharm/_jb_unittes
Testing started at 17:21 ...

Launching unittests with arguments python -m unittest test_queries.TestQueries.test_запрос_1_отделы_со_словом_отдел in E:\3sem\Парадигмы\РК2
Ran 1 test in 0.001s

OK

Process finished with exit code 0
```


Tests passed: 1 of 1 test – 1 ms
E:\3sem\Парадигмы\РК2\.venv\Scripts\python.exe "D:/Work fignya/PyCharm Community Edition 2024.3.3/plugins/python-ce/helpers/pycharm/_jb_unittest_r
Testing started at 17:21 ...
Ran 1 test in 0.001s
OK
Launching unittests with arguments python -m unittest test_queries.TestQueries.test_запрос_2_средняя_зарплата_по_отделам in E:\3sem\Парадигмы\РК2
Process finished with exit code 0


Tests passed: 1 of 1 test – 0 ms
E:\3sem\Парадигмы\РК2\.venv\Scripts\python.exe "D:/Work fignya/PyCharm Community Edition 2024.3.3/plugins/python-ce/helpers/pycharm/_jb
Testing started at 17:22 ...
Launching unittests with arguments python -m unittest test_queries.TestQueries.test_запрос_3_сотрудники_на_а in E:\3sem\Парадигмы\РК2
Ran 1 test in 0.001s
OK
Process finished with exit code 0