Московский государственный технический университет им. Н.Э. Баумана

Факультет РТ
Кафедра РТ5

Курс «Парадигмы и конструкции языков программирования»

Отчет по Рубежному контролю №2

Выполнил:

студент группы РТ5-21Б:

Хашиева Раяна Харомовна

Проверил:

преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Москва, 2025 г.

Постановка задачи

Рубежный контроль представляет собой разработку тестов на языке Python.
1) Проведите рефакторинг токста программы рубежного контроля №1 таким образом,
чтобы он оыл пригоден для модульного тестирования
2) Для текста программы рубежного контроля №l создайте модульные тесты с применением TDD - фреймворка (3 теста).

Текст программы

models.py:

```
class Operator:
    def __init__(self, id_operator, name, salary, id_language=None):
        self.id_operator = id_operator
        self.name = name
        self.salary = salary
        self.id_language = id_language

    def __repr__(self):
        return f"Operator({self.name}, зарплата: {self.salary})"


class ProgrammingLanguage:
    def __init__(self, id_language, name):
        self.id_language = id_language
        self.name = name

    def __repr__(self):
        return f"Language({self.name})"


class OperatorLanguage:
    def __init__(self, id_operator, id_language):
        self.id_operator = id_operator
        self.id_language = id_language
```

data.py:

```
from models import Operator, ProgrammingLanguage, OperatorLanguage

languages = [
    ProgrammingLanguage(1, "Python"),
    ProgrammingLanguage(2, "C++"),
    ProgrammingLanguage(3, "JavaScript"),
]

operators = [
    Operator(1, "Алиса", 85000, 1),
    Operator(2, "Борис", 90000, 2),
    Operator(3, "Андрей", 75000, 1),
    Operator(4, "Виктор", 95000, 3),
    Operator(5, "Алина", 88000, 2),
]

operator_languages = [
    OperatorLanguage(1, 1),
    OperatorLanguage(1, 2),
    OperatorLanguage(2, 2),
    OperatorLanguage(3, 1),
    OperatorLanguage(3, 3),
    OperatorLanguage(4, 3),
    OperatorLanguage(5, 1),
    OperatorLanguage(5, 2),
]
```

logic.py:

```
def get_languages_with_c(languages, operators):
    result = {}
    for lang in languages:
        if "C" in lang.name:
            ops = [op.name for op in operators if op.id_language == lang.id_language]
            result[lang.name] = ops
    return result


def get_average_salary_by_language(languages, operators):
    result = {}
    for lang in languages:
        lang_ops = [op.salary for op in operators if op.id_language == lang.id_language]
        if lang_ops:
```

```
            result[lang.name] = round(sum(lang_ops) / len(lang_ops), 2)
        else:
            result[lang.name] = None
    return result


def get_operators_starting_with_a(operators, languages, relations):
    result = {}
    for op in operators:
        if op.name.startswith("A"):
            langs = [
                lang.name
                for rel in relations
                for lang in languages
                if rel.id_operator == op.id_operator
                and rel.id_language == lang.id_language
            ]
            result[op.name] = langs
    return result
```

main.py:

```
from data import languages, operators, operator_languages
from logic import *

if __name__ == "__main__":
    print("Запрос 1:")
    print(get_languages_with_c(languages, operators))

    print("\nЗапрос 2:")
    print(get_average_salary_by_language(languages, operators))

    print("\nЗапрос 3:")
    print(get_operators_starting_with_a(operators, languages, operator_languages))
```

test_logic.py:

```
import unittest
from data import languages, operators, operator_languages
from logic import *

class TestOperatorLogic(unittest.TestCase):

    def test_languages_with_c(self):
        result = get_languages_with_c(languages, operators)
        self.assertIn("C++", result)
        self.assertEqual(result["C++"], ["Борис", "Алина"])

    def test_average_salary(self):
        result = get_average_salary_by_language(languages, operators)
        self.assertEqual(result["Python"], 80000.0)

    def test_operators_starting_with_a(self):
        result = get_operators_starting_with_a(
            operators, languages, operator_languages
        )
        self.assertIn("Алиса", result)
        self.assertIn("Python", result["Алиса"])


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

Анализ результатов

```
/Users/raana/Desktop/RK2/main.py
Запрос 1:
{'C++': ['Борис', 'Алина']}

Запрос 2:
{'Python': 80000.0, 'C++': 89000.0, 'JavaScript': 95000.
0}

Запрос 3:
{'Алиса': ['Python', 'C++'], 'Андрей': ['Python', 'JavaS
cript'], 'Алина': ['Python', 'C++']}
```