Отчет

Парадигмы и конструкции языков программирования

Рубежный контроль №2

Вариант 12

Студент: Крючков Д.И.

Группа: ИБМ3-34Б

Преподаватель: Гапанюк Ю.Е.

Текст задания:

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).

Задача №1:

```
def get_data():
    # Список средств разработки
    tools = [
        {"id": 1, "title": "Visual Studio Code"},
        {"id": 2, "title": "PyCharm"},
        {"id": 3, "title": "IntelliJ IDEA"},
        {"id": 4, "title": "Xcode"},
    ]

    # Список языков программирования
    languages = [
        {"id": 1, "name": "Python",      "vacancies": 950,
"primary_tool_id": 2},
        {"id": 2, "name": "Java",        "vacancies": 1100,
"primary_tool_id": 3},
        {"id": 3, "name": "C#",          "vacancies": 700,
"primary_tool_id": 1},
        {"id": 4, "name": "Swift",       "vacancies": 400,
"primary_tool_id": 4},
        {"id": 5, "name": "JavaScript",  "vacancies": 1300,
"primary_tool_id": 1},
    ]

    # Связь многие-ко-многим: язык ↔ средство разработки
    lang_tools = [
        # Python
        {"lang_id": 1, "tool_id": 1},
        {"lang_id": 1, "tool_id": 2},
        # Java
        {"lang_id": 2, "tool_id": 1},
        {"lang_id": 2, "tool_id": 3},
        # C#
        {"lang_id": 3, "tool_id": 1},
        # Swift
        {"lang_id": 4, "tool_id": 4},
        # JavaScript
        {"lang_id": 5, "tool_id": 1},
        {"lang_id": 5, "tool_id": 3},
    ]
```

```python
    return tools, languages, lang_tools


# ====== ЗАПРОС A1 (1 -> M) ======

def task_a1(tools, languages):
    """
    A1: 1 -> M
    Для каждого средства разработки вернуть список языков,
    где это средство является ОСНОВНЫМ.
    Результат: список кортежей (название_средства, [список_языков])
    """
    result = []

    # сортируем инструменты по названию
    tools_sorted = sorted(tools, key=lambda t: t["title"].lower())

    for tool in tools_sorted:
        langs_for_tool = []

        # ищем языки, у которых primary_tool_id = id этого инструмента
        for lang in languages:
            if lang["primary_tool_id"] == tool["id"]:
                langs_for_tool.append(lang["name"])

        # сортируем языки по алфавиту
        langs_for_tool.sort()

        # в результат кладём кортеж (строка, список)
        result.append((tool["title"], langs_for_tool))

    return result


# ====== ЗАПРОС A2 (M <-> M, сумма вакансий) ======

def task_a2(tools, languages, lang_tools):
    """
    A2: M <-> M
    Для каждого средства разработки посчитать СУММУ вакансий
    по всем связанным с ним языкам (через lang_tools).
    Результат: список (название_средства, сумма_вакансий), отсортирован по
убыванию.
    """
    result = []

    for tool in tools:
        # Сначала найдём id языков, связанных с этим инструментом
        lang_ids = []
        for lt in lang_tools:
            if lt["tool_id"] == tool["id"]:
                lang_ids.append(lt["lang_id"])

        # Теперь считаем сумму вакансий по этим языкам
        total = 0
        for lang in languages:
            if lang["id"] in lang_ids:
                total += lang["vacancies"]

        result.append((tool["title"], total))

    # сортируем по сумме вакансий по убыванию
    result.sort(key=lambda pair: pair[1], reverse=True)
    return result
```

```python
# ====== ЗАПРОС А3 (М <-> М + фильтр по слову в названии) ======

def task_a3(tools, languages, lang_tools, keyword="code"):
    """
    А3: М <-> М + фильтр по названию средства разработки.
    Находим средства, в названии которых есть keyword (по умолчанию "code"),
    и возвращаем словарь: {название_средства: [список_языков]}.
    """
    keyword = keyword.lower()
    result = {}

    for tool in tools:
        # проверяем, есть ли keyword в названии
        if keyword in tool["title"].lower():
            # собираем id языков для этого инструмента
            lang_ids = []
            for lt in lang_tools:
                if lt["tool_id"] == tool["id"]:
                    lang_ids.append(lt["lang_id"])

            # собираем имена языков
            langs_for_tool = []
            for lang in languages:
                if lang["id"] in lang_ids:
                    langs_for_tool.append(lang["name"])

            # убираем дубли и сортируем
            langs_for_tool = sorted(set(langs_for_tool))

            result[tool["title"]] = langs_for_tool

    return result


# ====== ПРОСТО ВЫВОД ДЛЯ ПРОВЕРКИ (НЕ ДЛЯ ТЕСТОВ) ======

def main():
    tools, languages, lang_tools = get_data()

    print("A1:")
    for tool_title, langs in task_a1(tools, languages):
        print(tool_title, "->", ", ".join(langs) if langs else "-")

    print("\nA2:")
    for tool_title, total in task_a2(tools, languages, lang_tools):
        print(tool_title, "->", total)

    print("\nA3:")
    result = task_a3(tools, languages, lang_tools)
    for tool_title, langs in result.items():
        print(tool_title, "->", ", ".join(langs) if langs else "-")


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

```
(.venv) daniilkryuchkov@MacBook-Air-Daniil RK2 % python refRK1.py
A1:
IntelliJ IDEA -> Java
PyCharm -> Python
Visual Studio Code -> C#, JavaScript
Xcode -> Swift

A2:
Visual Studio Code -> 4050
IntelliJ IDEA -> 2400
PyCharm -> 950
Xcode -> 400

A3:
Visual Studio Code -> C#, Java, JavaScript, Python
Xcode -> Swift
```

Задача №2:

```python
import unittest

from refRK1 import get_data, task_a1, task_a2, task_a3


class TestVariant12A(unittest.TestCase):
    def setUp(self):
        self.tools, self.languages, self.lang_tools = get_data()

    def test_task_a1(self):
        # Проверяем связь 1 -> М и порядок
        result = task_a1(self.tools, self.languages)

        expected = [
            ("IntelliJ IDEA", ["Java"]),
            ("PyCharm", ["Python"]),
            ("Visual Studio Code", ["C#", "JavaScript"]),
            ("Xcode", ["Swift"]),
        ]

        self.assertEqual(result, expected)

    def test_task_a2(self):
        # Проверяем суммы вакансий по М <-> М и сортировку
        result = task_a2(self.tools, self.languages, self.lang_tools)

        expected = [
            ("Visual Studio Code", 4050),  # Python + Java + C# + JavaScript
            ("IntelliJ IDEA", 2400),       # Java + JavaScript
            ("PyCharm", 950),              # Python
            ("Xcode", 400),                # Swift
        ]

        self.assertEqual(result, expected)

    def test_task_a3(self):
        # Проверяем выборку по слову "code" в названии
        result = task_a3(self.tools, self.languages, self.lang_tools,
keyword="code")
```

```
        expected = {
            "Visual Studio Code": ["C#", "Java", "JavaScript", "Python"],
            "Xcode": ["Swift"],
        }

        self.assertEqual(result, expected)


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

```
(.venv) daniilkryuchkov@MacBook-Air-Daniil RK2 % python -m unittest tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s
```