Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования »

Рубежный контроль №2
Вариант Е10

Выполнил:

студент группы
РТ5-31Б

Ермаков И.А.

Москва, 2024 г

Текст программы

Рефакторинг программы:

```
from operator import itemgetter

class Computer:
    """Компьютер"""
    def __init__(self, computer_id, owner_name,
processing_power, browser_id):
        self.computer_id = computer_id
        self.owner_name = owner_name
        self.processing_power = processing_power
        self.browser_id = browser_id

class Browser:
    """Браузер"""
    def __init__(self, browser_id, name):
        self.browser_id = browser_id
        self.name = name

class ComputerBrowser:
    """Компьютеры и браузеры для связи многие-ко-многим"""
    def __init__(self, computer_id, browser_id):
        self.computer_id = computer_id
        self.browser_id = browser_id

# Тестовые данные
computers = [
    Computer(1, 'Андреев', 3.2, 1),
    Computer(2, 'Петров', 4.1, 2),
    Computer(3, 'Антонов', 2.9, 1),
    Computer(4, 'Иванов', 4.0, 3),
    Computer(5, 'Смирнов', 3.0, 1),
]

browsers = [
    Browser(1, 'Интернет-браузер Chrome'),
    Browser(2, 'Браузер безопасности Firefox'),
    Browser(3, 'Мобильный браузер Safari'),
]

computer_browsers = [
    ComputerBrowser(1, 1),
    ComputerBrowser(2, 2),
    ComputerBrowser(3, 1),
    ComputerBrowser(4, 3),
    ComputerBrowser(5, 1),
```

```
]

# Функция для подготовки данных один-ко-многим
def one_to_many(computers, browsers):
    return [
        (c.owner_name, c.processing_power, b.name)
        for b in browsers
        for c in computers
        if c.browser_id == b.browser_id
    ]

# Функция для подготовки данных многие-ко-многим
def many_to_many(computers, browsers, computer_browsers):
    temp = [
        (b.name, cb.browser_id, cb.computer_id)
        for b in browsers
        for cb in computer_browsers
        if b.browser_id == cb.browser_id
    ]
    return [
        (c.owner_name, c.processing_power, browser_name)
        for browser_name, _, computer_id in temp
        for c in computers if c.computer_id == computer_id
    ]

# Задание A1
def get_browsers_with_computers(one_to_many_result):
    result = {}
    for record in one_to_many_result:
        owner_name, _, browser_name = record
        if "браузер" in browser_name.lower():
            if browser_name not in result:
                result[browser_name] = []
            result[browser_name].append(owner_name)
    return result

# Задание A2
def get_browsers_with_avg_processing(one_to_many_result):
    avg_result = []
    for browser_name in set([record[2] for record in
one_to_many_result]):
        power_list = [processing for _, processing, b_name in
one_to_many_result if b_name == browser_name]
        if power_list:
            avg_power = round(sum(power_list) / len(power_list),
2)
            avg_result.append((browser_name, avg_power))
    return sorted(avg_result, key=itemgetter(1))

# Задание A3
```

```
def
get_computers_with_browsers_starting_with_A(many_to_many_result)
:
    result = {}
    for record in many_to_many_result:
        owner_name, _, browser_name = record
        if owner_name.startswith("A"):
            if owner_name not in result:
                result[owner_name] = []
            result[owner_name].append(browser_name)
    return result

if __name__ == "__main__":
    # Соединение данных один-ко-многим
    one_to_many_result = one_to_many(computers, browsers)

    # Задание A1
    print("Задание A1:")
    browsers_with_computers =
get_browsers_with_computers(one_to_many_result)
    for browser_name, owners in browsers_with_computers.items():
        print(f"{browser_name}:")
        for owner in owners:
            print(f"  - {owner}")

    # Задание A2: Средняя производительность по браузерам
    print("\nЗадание A2:")
    browsers_with_avg_processing =
get_browsers_with_avg_processing(one_to_many_result)
    for browser_name, avg_power in browsers_with_avg_processing:
        print(f"{browser_name}: {avg_power}")

    # Соединение данных многие-ко-многим
    many_to_many_result = many_to_many(computers, browsers,
computer_browsers)

    # Задание A3
    print("\nЗадание A3:")
    computers_with_browsers_A =
get_computers_with_browsers_starting_with_A(many_to_many_result)
    for owner_name, browser_list in
computers_with_browsers_A.items():
        print(f"{owner_name}:")
        for browser_name in browser_list:
            print(f"  - {browser_name}")
```

Модульные тесты с применением TDD - фреймворка:

```
import unittest
from refact_main import Computer, Browser, ComputerBrowser,
one_to_many, many_to_many, get_browsers_with_computers,
get_browsers_with_avg_processing,
get_computers_with_browsers_starting_with_A

class TestComputerBrowserFunctions(unittest.TestCase):
    def setUp(self):
        # Инициализация тестовых данных
        self.computers = [
            Computer(1, 'Андреев', 3.2, 1),
            Computer(2, 'Петров', 4.1, 2),
            Computer(3, 'Антонов', 2.9, 1),
            Computer(4, 'Иванов', 4.0, 3),
            Computer(5, 'Смирнов', 3.0, 1),
        ]

        self.browsers = [
            Browser(1, 'Интернет-браузер Chrome'),
            Browser(2, 'Браузер безопасности Firefox'),
            Browser(3, 'Мобильный браузер Safari'),
        ]

        self.computer_browsers = [
            ComputerBrowser(1, 1),
            ComputerBrowser(2, 2),
            ComputerBrowser(3, 1),
            ComputerBrowser(4, 3),
            ComputerBrowser(5, 1),
        ]

    def test_get_browsers_with_computers(self):
        one_to_many_result = one_to_many(self.computers,
self.browsers)
        expected = {
            'Интернет-браузер Chrome': ['Андреев', 'Антонов',
'Смирнов'],
            'Браузер безопасности Firefox': ['Петров'],
            'Мобильный браузер Safari': ['Иванов']
        }

self.assertEqual(get_browsers_with_computers(one_to_many_result)
, expected)

    def test_get_browsers_with_avg_processing(self):
```

```
        one_to_many_result = one_to_many(self.computers,
self.browsers)
        expected = [
            ('Интернет-браузер Chrome', 3.03),
            ('Мобильный браузер Safari', 4.0),
            ('Браузер безопасности Firefox', 4.1)
        ]

self.assertEqual(get_browsers_with_avg_processing(one_to_many_re
sult), expected)

    def test_get_computers_with_browsers_starting_with_A(self):
        many_to_many_result = many_to_many(self.computers,
self.browsers, self.computer_browsers)
        expected = {
            'Андреев': ['Интернет-браузер Chrome'],
            'Антонов': ['Интернет-браузер Chrome']
        }

self.assertEqual(get_computers_with_browsers_starting_with_A(man
y_to_many_result), expected)

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

Результат тестов:

```
Ran 3 tests in 0.000s

OK
```