МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”
Отчет по рубежному контролю №2

Выполнил:
Студент группы ИУ5-35Б
Евдокимов М. С.
Проверил:
Гапанюк Ю. Е.
подпись, дата
_____/_____

Москва 2025

Задание

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).

Листинг программы

Файл refactored_browsers.py

```
from operator import itemgetter


class Browser:
    def __init__(self, id, name, memory_usage, computer_id):
        self.id = id
        self.name = name
        self.memory_usage = memory_usage
        self.computer_id = computer_id


class Computer:
    def __init__(self, id, model):
        self.id = id
        self.model = model


class BrowserComputer:
    def __init__(self, computer_id, browser_id):
        self.computer_id = computer_id
        self.browser_id = browser_id


computers = [
    Computer(1, 'Asus ROG'),
    Computer(2, 'Apple MacBook Pro'),
    Computer(3, 'Dell XPS'),
    Computer(11, 'Acer Predator'),
    Computer(22, 'Apple iMac'),
    Computer(33, 'Asus ZenBook'),
]

browsers = [
    Browser(1, 'Chrome', 512, 1),
    Browser(2, 'Firefox', 256, 2),
    Browser(3, 'Safari', 384, 3),
    Browser(4, 'Edge', 320, 3),
    Browser(5, 'Opera', 192, 3),
]

browsers_computers = [
    BrowserComputer(1, 1),
    BrowserComputer(2, 2),
    BrowserComputer(3, 3),
    BrowserComputer(3, 4),
    BrowserComputer(3, 5),
    BrowserComputer(11, 1),
    BrowserComputer(22, 2),
    BrowserComputer(33, 3),
    BrowserComputer(33, 4),
    BrowserComputer(33, 5),
]
```

```
def create_one_to_many(computers, browsers):
    return [(b.name, b.memory_usage, c.model)
            for c in computers
            for b in browsers
            if b.computer_id == c.id]


def create_many_to_many(computers, browsers, browsers_computers):
    many_to_many_temp = [(c.model, bc.computer_id, bc.browser_id)
                         for c in computers
                         for bc in browsers_computers
                         if c.id == bc.computer_id]

    return [(b.name, b.memory_usage, comp_model)
            for comp_model, comp_id, browser_id in many_to_many_temp
            for b in browsers if b.id == browser_id]


def task1_filter_computers_starting_with_a(many_to_many):
    comps_with_a = list(filter(lambda i: i[2].startswith('A'), many_to_many))

    result = {}
    for browser_name, memory, comp_model in comps_with_a:
        if comp_model not in result:
            result[comp_model] = []
        if browser_name not in result[comp_model]:
            result[comp_model].append(browser_name)

    return result


def task2_max_memory_per_computer(computers, many_to_many):
    result_unsorted = []

    for c in computers:
        c_browsers = list(filter(lambda i: i[2] == c.model, many_to_many))
        if len(c_browsers) > 0:
            c_memory = [memory for _, memory, _ in c_browsers]
            c_memory_max = max(c_memory)
            result_unsorted.append((c.model, c_memory_max))
        else:
            result_unsorted.append((c.model, 0))

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


def task3_all_connections_sorted(many_to_many):
    sorted_connections = sorted(many_to_many, key=itemgetter(2))

    grouped_result = {}
    for browser_name, memory, comp_model in sorted_connections:
        if comp_model not in grouped_result:
            grouped_result[comp_model] = []
        grouped_result[comp_model].append(f"{browser_name} ({memory} МБ)")
```

```
    return grouped_result


def print_task1(result):
    print('Задание Г1')
    print('Список всех компьютеров, у которых модель начинается с буквы "А", и список браузеров на них:')
    for comp, browsers_list in sorted(result.items()):
        print(f'{comp}: {browsers_list}')


def print_task2(result):
    print('\nЗадание Г2')
    print('Список компьютеров с максимальным использованием памяти браузеров на каждом компьютере,
    отсортированный по максимальному использованию памяти:')
    for comp, max_memory in result:
        print(f'{comp}: {max_memory} МБ')


def print_task3(result):
    print('\nЗадание Г3')
    print('Список всех связанных браузеров и компьютеров, отсортированный по компьютерам:')
    for comp_model, browsers_list in sorted(result.items()):
        print(f'{comp_model}: {browsers_list}')


def main():
    one_to_many = create_one_to_many(computers, browsers)
    many_to_many = create_many_to_many(computers, browsers, browsers_computers)

    task1_result = task1_filter_computers_starting_with_a(many_to_many)
    print_task1(task1_result)

    task2_result = task2_max_memory_per_computer(computers, many_to_many)
    print_task2(task2_result)

    task3_result = task3_all_connections_sorted(many_to_many)
    print_task3(task3_result)


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

Файл test_browsers.py

```
import unittest
from refactored_browsers import (
    task1_filter_computers_starting_with_a,
    task2_max_memory_per_computer,
    task3_all_connections_sorted
)


class TestBrowserSystem(unittest.TestCase):

    def test_task1_filter_computers_starting_with_a(self):
        """Тест для задания 1: фильтрация компьютеров на букву 'А'"""
```

```
        test_data = [
            ('Chrome', 512, 'Asus ROG'),
            ('Firefox', 256, 'Apple MacBook Pro'),
            ('Safari', 384, 'Acer Predator'),
            ('Edge', 320, 'Dell XPS'),
            ('Opera', 192, 'Asus ZenBook'),
        ]

        result = task1_filter_computers_starting_with_a(test_data)

        self.assertIsInstance(result, dict)

        self.assertIn('Asus ROG', result)
        self.assertIn('Apple MacBook Pro', result)
        self.assertIn('Acer Predator', result)
        self.assertIn('Asus ZenBook', result)
        self.assertNotIn('Dell XPS', result)

        self.assertIn('Chrome', result['Asus ROG'])

        test_data_with_duplicates = [
            ('Chrome', 512, 'Asus ROG'),
            ('Chrome', 512, 'Asus ROG'),
            ('Firefox', 256, 'Asus ROG'),
        ]
        result2 = task1_filter_computers_starting_with_a(test_data_with_duplicates)
        self.assertEqual(len(result2['Asus ROG']), 2)

    def test_task2_max_memory_per_computer(self):
        """Тест для задания 2: максимальная память на каждом компьютере"""
        from refactored_browsers import Computer

        test_data = [
            ('Chrome', 512, 'Asus ROG'),
            ('Firefox', 256, 'Asus ROG'),
            ('Safari', 384, 'Apple MacBook Pro'),
            ('Edge', 320, 'Apple MacBook Pro'),
            ('Opera', 192, 'Dell XPS'),
        ]

        computers_list = [
            Computer(1, 'Asus ROG'),
            Computer(2, 'Apple MacBook Pro'),
            Computer(3, 'Dell XPS'),
            Computer(4, 'Empty Computer'),
        ]

        result = task2_max_memory_per_computer(computers_list, test_data)

        self.assertIsInstance(result, list)
        self.assertEqual(len(result), 4)

        self.assertEqual(result[0][0], 'Asus ROG')
        self.assertEqual(result[1][0], 'Apple MacBook Pro')
        self.assertEqual(result[2][0], 'Dell XPS')
        self.assertEqual(result[3][0], 'Empty Computer')
```

```
        result_dict = dict(result)
        self.assertEqual(result_dict['Asus ROG'], 512)
        self.assertEqual(result_dict['Apple MacBook Pro'], 384)
        self.assertEqual(result_dict['Dell XPS'], 192)
        self.assertEqual(result_dict['Empty Computer'], 0)

    def test_task3_all_connections_sorted(self):
        """Тест для задания 3: сортировка всех связей по компьютерам"""
        test_data = [
            ('Firefox', 256, 'Apple MacBook Pro'),
            ('Chrome', 512, 'Asus ROG'),
            ('Safari', 384, 'Apple MacBook Pro'),
            ('Edge', 320, 'Asus ROG'),
            ('Opera', 192, 'Dell XPS'),
        ]

        result = task3_all_connections_sorted(test_data)

        self.assertIsInstance(result, dict)

        keys = list(result.keys())
        self.assertEqual(keys, ['Apple MacBook Pro', 'Asus ROG', 'Dell XPS'])

        self.assertIn('Chrome (512 МБ)', result['Asus ROG'])
        self.assertIn('Edge (320 МБ)', result['Asus ROG'])
        self.assertIn('Firefox (256 МБ)', result['Apple MacBook Pro'])
        self.assertIn('Safari (384 МБ)', result['Apple MacBook Pro'])
        self.assertIn('Opera (192 МБ)', result['Dell XPS'])

        self.assertEqual(len(result['Apple MacBook Pro']), 2)
        self.assertEqual(len(result['Asus ROG']), 2)
        self.assertEqual(len(result['Dell XPS']), 1)


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

Скриншоты работы программы

```
C:\Users\Серьга\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Серьга\PycharmProjects\pythonProject\refactored_browsers.py
Задание Г1
Список всех компьютеров, у которых модель начинается с буквы "A", и список браузеров на них:
Acer Predator: ['Chrome']
Apple MacBook Pro: ['Firefox']
Apple iMac: ['Firefox']
Asus ROG: ['Chrome']
Asus ZenBook: ['Safari', 'Edge', 'Opera']

Задание Г2
Список компьютеров с максимальным использованием памяти браузеров на каждом компьютере, отсортированный по максимальному использованию памяти:
Asus ROG: 512 МБ
Acer Predator: 512 МБ
Dell XPS: 384 МБ
Asus ZenBook: 384 МБ
Apple MacBook Pro: 256 МБ
Apple iMac: 256 МБ

Задание Г3
Список всех связанных браузеров и компьютеров, отсортированный по компьютерам:
Acer Predator: ['Chrome (512 МБ)']
Apple MacBook Pro: ['Firefox (256 МБ)']
Apple iMac: ['Firefox (256 МБ)']
Asus ROG: ['Chrome (512 МБ)']
Asus ZenBook: ['Safari (384 МБ)', 'Edge (320 МБ)', 'Opera (192 МБ)']
Dell XPS: ['Safari (384 МБ)', 'Edge (320 МБ)', 'Opera (192 МБ)']

Process finished with exit code 0
```

```
Testing started at 22:13 ...
Launching unittests with arguments python -m unittest test_browsers.TestBrowserSystem in C:\Users\Серьга\PycharmProjects\pythonProject



Ran 3 tests in 0.002s

OK

Process finished with exit code 0
```