

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 ИУ5-36Б

Левочкин В.В.

**Преподаватель:**

Нардид. А.Н.

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 так чтобы он был пригоден для модульного тестирования.

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

### Refactored\_code.py

```
class Computer:
    """Класс Компьютер"""
    def __init__(self, id, model):
        self.id=id
        self.model=model

    def __repr__(self):
        return f"Computer(id={self.id}, model='{self.model}')"

class Browser:
    """Класс Браузер"""
    def __init__(self, id, name, memory_usage, computer_id):
        self.id=id
        self.name=name
        self.memory_usage=memory_usage # количественный признак - использование
        # памяти в МБ
        self.computer_id=computer_id

    def __repr__(self):
        return f"Browser(id={self.id}, name='{self.name}',
        memory_usage={self.memory_usage}, computer_id={self.computer_id})"

class BrowserComputer:
    """Класс для связи многие-ко-многим между Браузером и Компьютером"""
    def __init__(self, browser_id, computer_id):
        self.browser_id=browser_id
        self.computer_id=computer_id

    def __repr__(self):
        return f"BrowserComputer(browser_id={self.browser_id},
        computer_id={self.computer_id})"
```

```

def has_russian_ov_ending(name):
    """Проверяет, заканчивается ли имя на 'ов' (русский суффикс)"""
    words = name.split()
    for word in words:
        if word.lower().endswith('ов'):
            return True
    return False

def query1_browsers_with_ov_ending(browsers, computers):
    """
    Запрос 1: Список всех браузеров, у которых название заканчивается на «ов»,
    и названия компьютеров, на которых они установлены
    """
    result = []

    browsers_with_ov = [browser for browser in browsers if has_russian_ov_ending(browser.name)]

    for browser in browsers_with_ov:
        computer = next((comp for comp in computers if comp.id == browser.computer_id),
            None)
        if computer:
            result.append((browser, computer))

    return result

def query2_computers_avg_memory(browsers, computers):
    """
    Запрос 2: Список компьютеров со средней нагрузкой памяти браузеров
    на каждом компьютере, отсортированный по средней нагрузке
    """
    from collections import defaultdict

    # Группируем браузеры по компьютерам
    browsers_by_computer = defaultdict(list)
    for browser in browsers:
        browsers_by_computer[browser.computer_id].append(browser)

    # Вычисляем среднюю нагрузку памяти для каждого компьютера
    computer_avg_memory = []
    for comp_id, comp_browsers in browsers_by_computer.items():
        if comp_browsers:
            total_memory = sum(browser.memory_usage for browser in comp_browsers)
            avg_memory = total_memory / len(comp_browsers)
            comp_model = next(comp.model for comp in computers if comp.id == comp_id)
            computer_avg_memory.append((comp_model, avg_memory))

    # Сортируем по средней нагрузке памяти (по возрастанию)
    computer_avg_memory.sort(key=lambda x: x[1])

    return computer_avg_memory

def query3_computers_starting_with_a(browsers, computers, browser_computers):

```

```

"""
Запрос 3: Список всех компьютеров, у которых модель начинается с буквы «А»,
и список браузеров, установленных на них (многие-ко-многим)
"""

from collections import defaultdict

# Создаем словари для быстрого доступа
browser_dict = {browser.id: browser for browser in browsers}
computer_dict = {comp.id: comp for comp in computers}

# Группируем связи по компьютерам
connections_by_computer = defaultdict(list)
for bc in browser_computers:
    browser = browser_dict.get(bc.browser_id)
    computer = computer_dict.get(bc.computer_id)
    if browser and computer:
        connections_by_computer[computer].append(browser)

# Фильтруем компьютеры, начинающиеся на "А"
computers_with_a = [comp for comp in connections_by_computer.keys()
                    if comp.model.startswith('A')]

# Сортируем компьютеры по названию модели
sorted_computers = sorted(computers_with_a, key=lambda x: x.model)

# Формируем результат
result = []
for computer in sorted_computers:
    computer_browsers = connections_by_computer[computer]
    result.append((computer, computer_browsers))

return result

def get_test_data():
    """Возвращает тестовые данные для программы"""
    # Создаем компьютеры
    computers = [
        Computer(1, "Dell XPS 15"),
        Computer(2, "HP Pavilion"),
        Computer(3, "Lenovo ThinkPad"),
        Computer(4, "Asus ROG"),
        Computer(5, "Acer Aspire")
    ]

    # Создаем браузеры (связь один-ко-многим)
    browsers = [
        Browser(1, "Chrome", 450, 1),
        Browser(2, "Firefox", 320, 1),
        Browser(3, "Edge", 280, 2),
        Browser(4, "Safari", 350, 3),
        Browser(5, "Opera", 180, 3),
        Browser(6, "Chrome", 420, 4),
        Browser(7, "Firefox", 300, 5),
        Browser(8, "Иванов Браузер", 250, 2),
        Browser(9, "Петров Навигатор", 190, 4),
        Browser(10, "Сидоров Обзоратель", 220, 1),
        Browser(11, "Google Chrome", 400, 2)
    ]

```

```

]

# Создаем связи многие-ко-многим
browser_computers=[
    BrowserComputer(1, 1), # Chrome - Dell XPS 15
    BrowserComputer(2, 1), # Firefox - Dell XPS 15
    BrowserComputer(3, 2), # Edge - HP Pavilion
    BrowserComputer(4, 3), # Safari - Lenovo ThinkPad
    BrowserComputer(5, 3), # Opera - Lenovo ThinkPad
    BrowserComputer(6, 4), # Chrome - Asus ROG
    BrowserComputer(7, 5), # Firefox - Acer Aspire
    BrowserComputer(8, 2), # Иванов Браузер - HP Pavilion
    BrowserComputer(9, 4), # Петров Навигатор - Asus ROG
    BrowserComputer(10, 1), # Сидоров Обозреватель - Dell XPS 15
    BrowserComputer(11, 2), # Google Chrome - HP Pavilion
    # Дополнительные связи для демонстрации многие-ко-многим
    BrowserComputer(1, 2), # Chrome также на HP Pavilion
    BrowserComputer(3, 1), # Edge также на Dell XPS 15
    BrowserComputer(5, 4), # Opera также на Asus ROG
    BrowserComputer(7, 3), # Firefox также на Lenovo ThinkPad
]

return computers, browsers, browser_computers

def main():
    """Основная функция программы"""
    computers, browsers, browser_computers = get_test_data()

    print("="*80)
    print("ТЕСТОВЫЕ ДАННЫЕ")
    print("="*80)

    print("\nКомпьютеры:")
    for comp in computers:
        print(f" {comp}")

    print("\nБраузеры:")
    for browser in browsers:
        print(f" {browser}")

    print("\nСвязи браузеры-компьютеры:")
    for bc in browser_computers:
        print(f" {bc}")

    print("\n"+"="*80)
    print("ЗАПРОС №1")
    print("="*80)
    result1 = query1_browsers_with_ov_ending(browsers, computers)
    for browser, computer in result1:
        print(f"Браузер: '{browser.name}', Компьютер: '{computer.model}'")

    print("\n"+"="*80)
    print("ЗАПРОС №2")
    print("="*80)

```

```

        result2=query2_computers_avg_memory(browsers, computers)
        for comp_model, avg_memory in result2:
            print(f"Компьютер '{comp_model}': средняя нагрузка памяти =
{avg_memory:.2f}МБ")

        print("\n"+"="*80)
        print("ЗАПРОС №3")
        print("="*80)
        result3=query3_computers_starting_with_a(browsers, computers,
browser_computers)
        for computer, computer_browsers in result3:
            print(f"\nКомпьютер: {computer.model}")
            for browser in computer_browsers:
                print(f"    - Браузер '{browser.name}' (память:
{browser.memory_usage}МБ)")

if __name__=="__main__":
    main()

```

### Test\_browser\_system.py

```

import unittest
from refactored_code import(
    Computer, Browser, BrowserComputer,
    has_russian_ov_ending,
    query1_browsers_with_ov_ending,
    query2_computers_avg_memory,
    query3_computers_starting_with_a
)

class TestBrowserSystem(unittest.TestCase):

    def setUp(self):
        self.computers=[
            Computer(1, "Dell XPS 15"),
            Computer(2, "HP Pavilion"),
            Computer(3, "Lenovo ThinkPad"),
            Computer(4, "Asus ROG"),
            Computer(5, "Acer Aspire")
        ]

        self.browsers=[
            Browser(1, "Chrome", 450, 1),
            Browser(2, "Firefox", 320, 1),
            Browser(3, "Edge", 280, 2),
            Browser(4, "Safari", 350, 3),
            Browser(5, "Opera", 180, 3),
            Browser(6, "Chrome", 420, 4),
            Browser(7, "Firefox", 300, 5),
            Browser(8, "Иванов Браузер", 250, 2),
            Browser(9, "Петров Навигатор", 190, 4),
            Browser(10, "Сидоров Обозреватель", 220, 1),
            Browser(11, "Google Chrome", 400, 2)
        ]

        self.browser_computers=[

```

```

        BrowserComputer(1, 1),
        BrowserComputer(2, 1),
        BrowserComputer(3, 2),
        BrowserComputer(4, 3),
        BrowserComputer(5, 3),
        BrowserComputer(6, 4),
        BrowserComputer(7, 5),
        BrowserComputer(8, 2),
        BrowserComputer(9, 4),
        BrowserComputer(10, 1),
        BrowserComputer(11, 2),
        BrowserComputer(1, 2),
        BrowserComputer(3, 1),
        BrowserComputer(5, 4),
        BrowserComputer(7, 3),
    ]

    def test_has_russian_ov_ending(self):
        self.assertTrue(has_russian_ov_ending("Иванов Браузер"))
        self.assertTrue(has_russian_ov_ending("Петров Навигатор"))
        self.assertTrue(has_russian_ov_ending("Сидоров Обозреватель"))
        self.assertTrue(has_russian_ov_ending("Сергеев Эксплорер"))

        self.assertFalse(has_russian_ov_ending("Chrome"))
        self.assertFalse(has_russian_ov_ending("Firefox"))
        self.assertFalse(has_russian_ov_ending("Microsoft Edge"))
        self.assertFalse(has_russian_ov_ending("Google Chrome"))

        self.assertTrue(has_russian_ov_ending("ИВАНОВ БРАУЗЕР"))
        self.assertTrue(has_russian_ov_ending("петров навигатор"))

    def test_query1_browsers_with_ov_ending(self):
        result=query1_browsers_with_ov_ending(self.browsers, self.computers)

        self.assertEqual(len(result), 3)

        browser_names=[browser.name for browser, _ in result]
        expected_names=["Иванов Браузер", "Петров Навигатор", "Сидоров
Обозреватель"]

        for name in browser_names:
            self.assertTrue(has_russian_ov_ending(name))

        computer_models=[computer.model for _, computer in result]
        self.assertIn("HP Pavilion", computer_models)
        self.assertIn("Asus ROG", computer_models)
        self.assertIn("Dell XPS 15", computer_models)

    def test_query2_computers_avg_memory(self):
        result=query2_computers_avg_memory(self.browsers, self.computers)

        self.assertEqual(len(result), 5)

        avg_memories=[avg_memory for _, avg_memory in result]
        self.assertEqual(avg_memories, sorted(avg_memories))

        dell_result=next((item for item in result if item[0] == "Dell XPS 15"), None)
        self.assertIsNotNone(dell_result)
        self.assertAlmostEqual(dell_result[1], 330.0, places=2)

```

```

acer_result=next((itemforiteminresultifitem[0] == "Acer Aspire"), None)
self.assertIsNotNone(acer_result)
self.assertAlmostEqual(acer_result[1], 300.0, places=2)

def test_query3_computers_starting_with_a(self):
    result=query3_computers_starting_with_a(
        self.browsers, self.computers, self.browser_computers
    )

    self.assertEqual(len(result), 2)

    computer_models=[computer.model for computer, _in result]
    self.assertEqual(computer_models, sorted(computer_models))

    expected_models=["Acer Aspire", "Asus ROG"]
    actual_models=[model for model in computer_models]
    self.assertEqual(actual_models, expected_models)

    asus_result=next((itemforiteminresultifitem[0].model == "Asus ROG"), None)
    self.assertIsNotNone(asus_result)
    asus_computer, asus_browsers=asus_result

    browser_names=[browser.name for browser in asus_browsers]
    self.assertIn("Chrome", browser_names)
    self.assertIn("Петров Навигатор", browser_names)
    self.assertIn("Opera", browser_names)

if __name__ == "__main__":
    unittest.main(verbosity=2)

```

**Вывод программы**

```

===== 6 passed in 0.02s =====
@vasyteri →/workspaces/Python/Lab_4 (main) $ /usr/bin/python3 /workspaces/Python/RK2/refactored_code.py
=====
ТЕСТОВЫЕ ДАННЫЕ
=====

Компьютеры:
Computer(id=1, model='Dell XPS 15')
Computer(id=2, model='HP Pavilion')
Computer(id=3, model='Lenovo ThinkPad')
Computer(id=4, model='Asus ROG')
Computer(id=5, model='Acer Aspire')

Браузеры:
Browser(id=1, name='Chrome', memory_usage=450, computer_id=1)
Browser(id=2, name='Firefox', memory_usage=320, computer_id=1)
Browser(id=3, name='Edge', memory_usage=280, computer_id=2)
Browser(id=4, name='Safari', memory_usage=350, computer_id=3)
Browser(id=5, name='Opera', memory_usage=180, computer_id=3)
Browser(id=6, name='Chrome', memory_usage=420, computer_id=4)
Browser(id=7, name='Firefox', memory_usage=300, computer_id=5)
Browser(id=8, name='Иванов Браузер', memory_usage=250, computer_id=2)
Browser(id=9, name='Петров Навигатор', memory_usage=190, computer_id=4)
Browser(id=10, name='Сидоров Обзоратель', memory_usage=220, computer_id=1)
Browser(id=11, name='Google Chrome', memory_usage=400, computer_id=2)

Связи браузеры-компьютеры:
BrowserComputer(browser_id=1, computer_id=1)
BrowserComputer(browser_id=2, computer_id=1)
BrowserComputer(browser_id=3, computer_id=2)
BrowserComputer(browser_id=4, computer_id=3)
BrowserComputer(browser_id=11, computer_id=2)
BrowserComputer(browser_id=1, computer_id=2)
BrowserComputer(browser_id=3, computer_id=1)
BrowserComputer(browser_id=5, computer_id=4)
BrowserComputer(browser_id=7, computer_id=3)

=====
ЗАПРОС №1
=====
Браузер: 'Иванов Браузер', Компьютер: 'HP Pavilion'
Браузер: 'Петров Навигатор', Компьютер: 'Asus ROG'
Браузер: 'Сидоров Обзоратель', Компьютер: 'Dell XPS 15'

=====
ЗАПРОС №2
=====
Компьютер 'Lenovo ThinkPad': средняя нагрузка памяти = 265.00 МБ
Компьютер 'Acer Aspire': средняя нагрузка памяти = 300.00 МБ
Компьютер 'Asus ROG': средняя нагрузка памяти = 305.00 МБ
Компьютер 'HP Pavilion': средняя нагрузка памяти = 310.00 МБ
Компьютер 'Dell XPS 15': средняя нагрузка памяти = 330.00 МБ

=====
ЗАПРОС №3
=====

Компьютер: Acer Aspire
- Браузер 'Firefox' (память: 300 МБ)

Компьютер: Asus ROG
- Браузер 'Chrome' (память: 420 МБ)
- Браузер 'Петров Навигатор' (память: 190 МБ)
- Браузер 'Opera' (память: 180 МБ)
• @vasyteri →/workspaces/Python/Lab_4 (main) $ /usr/bin/python3 /workspaces/Python/RK2/test_browser_system.py
test_has_russian_ov_ending (__main__.TestBrowserSystem.test_has_russian_ov_ending) ... ok
test_query1_browsers_with_ov_ending (__main__.TestBrowserSystem.test_query1_browsers_with_ov_ending) ... ok
test_query2_computers_avg_memory (__main__.TestBrowserSystem.test_query2_computers_avg_memory) ... ok
test_query3_computers_starting_with_a (__main__.TestBrowserSystem.test_query3_computers_starting_with_a) ... ok

-----
Ran 4 tests in 0.001s

OK

```