Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Инженерный бизнес и менеджмент

КАФЕДРА Бизнес-информатика

Отчет по рубежному контролю №2

“Парадигмы и конструкции языков программирования”

Вариант №10

Студент **В.Д. Зазовский**

Группа ИБМ3-34Б

Преподаватель **Гапанюк В.Ю.**

Москва, 2024

Код программы (main):

```
from operator import itemgetter

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

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

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

def get_one_to_many(browsers, computers):
    """Возвращает связь один-ко-многим: Браузер -> Компьютер"""
    return [(c.model, b.name) for b in browsers for c in computers if
c.browser_id == b.id]

def get_many_to_many(browsers, computers, browsers_computers):
    """Возвращает связь многие-ко-многим: Браузер -> Компьютер"""
    many_to_many_temp = [(b.name, bc.browser_id, bc.computer_id) for b in
browsers for bc in browsers_computers if b.id == bc.browser_id]
    return [(c.model, b_name) for b_name, browser_id, computer_id in
many_to_many_temp for c in computers if c.id == computer_id]

def task_1(one_to_many):
    """Все браузеры и связанные с ними компьютеры, отсортированные по
названию браузеров"""
    return sorted(one_to_many, key=itemgetter(1))

def task_2(one_to_many, browsers):
    """Количество компьютеров для каждого браузера, отсортированное по
количеству компьютеров"""
```

```
    result = []
    for b in browsers:
        computers_list = list(filter(lambda x: x[1] == b.name,
one_to_many))
        result.append((b.name, len(computers_list)))
    return sorted(result, key=itemgetter(1), reverse=True)

def task_3(many_to_many, browsers):
    """Все браузеры, содержащие слово 'Chrome', и их компьютеры"""
    result = {}
    for b in browsers:
        if "Chrome" in b.name:
            computers_list = list(filter(lambda x: x[1] == b.name,
many_to_many))
            computers_names = [x[0] for x in computers_list]
            result[b.name] = computers_names
    return result
```

Вывод (main):

```
Задание 1:
[('Dell XPS', 'Chrome'), ('HP Spectre', 'Chrome'), ('MacBook Pro', 'Firefox'), ('ThinkPad', 'Firefox'), ('Mac Mini', 'Edge')]

Задание 2:
[('Chrome', 2), ('Firefox', 2), ('Edge', 1)]

Задание 3:
{'Chrome': ['Dell XPS', 'HP Spectre']}
```

Код (Тесты TDD):

```
import unittest

class TestBrowserFunctions(unittest.TestCase):
    def setUp(self):
        self.browsers = [
            Browser(1, "Chrome"),
            Browser(2, "Firefox"),
            Browser(3, "Edge"),
        ]
        self.computers = [
            Computer(1, "Dell XPS", 1),
```

```
            Computer(2, "MacBook Pro", 2),
            Computer(3, "HP Spectre", 1),
            Computer(4, "ThinkPad", 2),
            Computer(5, "Mac Mini", 3),
        ]
        self.browsers_computers = [
            BrowserComputer(1, 1),
            BrowserComputer(2, 2),
            BrowserComputer(1, 3),
            BrowserComputer(2, 4),
            BrowserComputer(3, 5),
        ]

    def test_task_1(self):
        one_to_many = get_one_to_many(self.browsers, self.computers)
        result = task_1(one_to_many)
        expected = [
            ('Dell XPS', 'Chrome'),
            ('HP Spectre', 'Chrome'),
            ('MacBook Pro', 'Firefox'),
            ('ThinkPad', 'Firefox'),
            ('Mac Mini', 'Edge')
        ]
        self.assertEqual(result, expected)

    def test_task_2(self):
        one_to_many = get_one_to_many(self.browsers, self.computers)
        result = task_2(one_to_many, self.browsers)
        expected = [('Chrome', 2), ('Firefox', 2), ('Edge', 1)]
        self.assertEqual(result, expected)

    def test_task_3(self):
        many_to_many = get_many_to_many(self.browsers, self.computers,
self.browsers_computers)
        result = task_3(many_to_many, self.browsers)
        expected = {'Chrome': ['Dell XPS', 'HP Spectre']}
        self.assertEqual(result, expected)

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

Вывод (Тесты TDD):

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```