Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю № 2
Вариант Г-10

Выполнил:
студент группы ИУ5-34Б
Нечаева Мария

Проверил:
преподаватель каф. ИУ5
Нардид А. Н.

Москва, 2024 г.

1) Был проведен рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования:

```
class Browser:
    """Браузер"""
    def __init__(self, id, name, comp_id):
        self.id = id
        self.name = name
        self.comp_id = comp_id
class Computer:
    """Компьютер"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
class BrowserComputer:
    """ для реализации связи многие-ко-многим"""
    def __init__(self, computer_id, browser_id, users_count):
        self.computer_id = computer_id
        self.browser_id = browser_id
        self.users_count = users_count

def create_one_to_many(Computers,Browsers):
     return [(brows.name, comp.id, comp.name)
                        for comp in Computers
                        for brows in Browsers
                        if brows.comp_id == comp.id]

def create_many_to_many(Browsers_Computers, Browsers, Computers):
    return [(brows.name, comp.name, cob.users_count)
                    for comp in Computers
                    for cob in Browsers_Computers
                    for brows in Browsers if brows.id == cob.browser_id and
comp.id == cob.computer_id]

def task_1(one_to_many,string):
    return list(filter(lambda x: x[2].startswith(string), one_to_many))
def task_2(many_to_many):
    max_users={}
    for brows_name, comp_name, users_count in many_to_many:
        if comp_name not in max_users or max_users[comp_name]<users_count:
            max_users[comp_name]=users_count
    return sorted(max_users.items(),key=lambda x: x[1])
def task_3(many_to_many):
    return sorted(many_to_many, key=lambda x: (x[1],x[0]))

def main():
    # Список компьютеров
    Computers = [
        Computer(1, "AComputer"),
        Computer(2, "AComputer1"),
        Computer(3, "BComputer"),
    ]

    # Список браузеров
    Browsers = [
        Browser(1, "Chrome", 1),
        Browser(2, "Firefox", 3),
        Browser(3, "Safari", 2),
    ]

    # Связь браузеров и компьютеров
```

```
    Browsers_Computers = [
        BrowserComputer(1, 1, 1000),
        BrowserComputer(2, 4, 2000),
        BrowserComputer(3, 2, 1500),
    ]
    one_to_many=create_one_to_many(Computers, Browsers)
    many_to_many=create_many_to_many(Browsers_Computers, Browsers, Computers)

    # Г1: Выводим список всех компьютеров, которые начинаются с буквы «А»
    print('Задание Г1')
    result = task_1(one_to_many, 'A')
    for i in result:
        print(f"Компьютер: {i[2]}, Браузер: {i[0]}")

    # Г2: Список компьютеров с максимальным количеством пользователей для
каждого браузера
    print('\nЗадание Г2')
    sorted_comp=task_2(many_to_many)
    for comp_name, users_count in sorted_comp:
        print(f"Компьютер: {comp_name}, Максимальное количество
пользователей: {users_count}")

    # Г3: Список всех связанных браузеров и компьютеров
    print('\nЗадание Г3')
    sorted_browsers = task_3(many_to_many)
    for brows in sorted_browsers:
        print(f"Средство разработки: {brows[1]}, Язык программирования:
{brows[0]}")

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

2) Для текста программы рубежного контроля №1 были созданы модульные тесты с применением TDD - фреймворка (3 теста):

```
import main
import unittest

class TestMethods(unittest.TestCase):

    def setUp(self):
        self.Computer = [
            main.Computer(1, "AComputer"),
            main.Computer(2, "AComputer1"),
            main.Computer(3, "BComputer"),
        ]
        self.Browser = [
            main.Browser(1, "Chrome", 1),
            main.Browser(2, "Firefox", 3),
            main.Browser(3, "Safari", 2),
        ]
        self.BrowserComputer = [
            main.BrowserComputer(1, 1, 1000),
            main.BrowserComputer(2, 4, 2000),
            main.BrowserComputer(3, 2, 1500),
        ]

        self.one_to_many = main.create_one_to_many(self.Computer,
self.Browser)
        self.many_to_many = main.create_many_to_many(self.BrowserComputer,
self.Browser, self.Computer)

    def test_first_task_method(self):
```

```
        result = [(i[2],i[0]) for i in main.task_1(self.one_to_many, 'A')]
        true_result = [('AComputer', 'Chrome'),
                       ('AComputer1', 'Safari')]
        self.assertEqual(result, true_result)

    def test_second_task_method(self):
        result = main.task_2(self.many_to_many)
        true_result = [('AComputer', 1000),
                       ('BComputer', 1500)]
        self.assertEqual(result, true_result)

    def test_third_task_method(self):
        result = [(i[1], i[0]) for i in main.task_3(self.many_to_many)]
        true_result = [('AComputer','Chrome'),
                       ('BComputer','Firefox')]
        self.assertEqual(result, true_result)


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

Результат выполнения программы:

```
Testing started at 0:22 ...
Launching unittests with arguments python -m unittest C:\INSTITUTE\2 YEAR\PROGRAMMING\RK2\RK2\unit_test.py in



Ran 3 tests in 0.003s

OK

Process finished with exit code 0
```