ОТЧЕТ

По РК-2

Дисциплина «Парадигмы и конструкции языков программирования»

Студент: Коваленко Е. Ю.

Группа: ИБМ3-34Б

Вариант 10: класс 1 – Браузер; класс 2 – Компьютер

Задание:

- Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования
- Для текста программы рубежного контроля №1 создать модульные тесты с применением TDD-фреймворка

Текст получившейся программы:

```
from dataclasses import dataclass

from typing import List, Dict, Optional

import unittest




@dataclass

class Computer:

    id: int

    name: str


@dataclass

class Browser:

    id: int

    name: str

    memory_usage: int

    computer_id: int


@dataclass

class ComputerBrowser:

    computer_id: int

    browser_id: int
```

```
class DataRepository:


    def __init__(self, computers=None, browsers=None, computer_browsers=None):
        self.computers = computers or []
        self.browsers = browsers or []
        self.computer_browsers = computer_browsers or []


    def get_default_data(self):
        self.computers = [
            Computer(1, "Офисный компьютер"),
            Computer(2, "Игровой компьютер"),
            Computer(3, "Серверный отдел"),
            Computer(4, "Главный компьютер")
        ]


        self.browsers = [
            Browser(1, "Chrome", 512, 1),
            Browser(2, "Firefox", 256, 1),
            Browser(3, "Edge", 384, 2),
            Browser(4, "Opera", 128, 2),
            Browser(5, "Safari", 192, 3),
            Browser(6, "Chrome", 512, 4)
        ]


        self.computer_browsers = [
            ComputerBrowser(1, 1),
```

```
            ComputerBrowser(1, 2),
            ComputerBrowser(2, 3),
            ComputerBrowser(2, 4),
            ComputerBrowser(3, 5),
            ComputerBrowser(4, 6),
            ComputerBrowser(1, 3),
            ComputerBrowser(2, 1)
        ]

        return self


class BrowserManager:
    def __init__(self, data_repository: Optional[DataRepository] = None):
        if data_repository:
            self.data = data_repository
        else:
            self.data = DataRepository().get_default_data()

    def _get_computers_dict(self) -> Dict[int, Computer]:
        return {comp.id: comp for comp in self.data.computers}

    def _get_browsers_dict(self) -> Dict[int, Browser]:
        return {browser.id: browser for browser in self.data.browsers}

    def task_1(self) -> Dict[int, List[Browser]]:
        browsers_by_computer = {}
```

```
    for browser in self.data.browsers:
        if browser.computer_id not in browsers_by_computer:
            browsers_by_computer[browser.computer_id] = []
        browsers_by_computer[browser.computer_id].append(browser)

    return browsers_by_computer

def print_task_1(self):
    print(" ЗАПРОС 1")
    browsers_by_computer = self.task_1()
    computer_dict = self._get_computers_dict()

    for computer_id in sorted(browsers_by_computer.keys()):
        computer = computer_dict[computer_id]
        print(f"\nКомпьютер: {computer.name} (ID: {computer.id})")
        for browser in browsers_by_computer[computer_id]:
            print(f"  - {browser.name} (использует памяти: {browser.memory_usage} МБ)")

def task_2(self) -> List[tuple]:
    memory_by_computer = {}

    for browser in self.data.browsers:
        if browser.computer_id not in memory_by_computer:
            memory_by_computer[browser.computer_id] = 0
        memory_by_computer[browser.computer_id] += browser.memory_usage
```

```
        return sorted(memory_by_computer.items(), key=lambda x: x[1])


    def print_task_2(self):
        print("\n ЗАПРОС 2")
        sorted_memory = self.task_2()
        computer_dict = self._get_computers_dict()

        for computer_id, total_memory in sorted_memory:
            computer = computer_dict[computer_id]
            print(f"Компьютер: {computer.name} - Суммарное использование памяти: {total_memory}
МБ")


    def task_3(self) -> Dict[int, List[Browser]]:
        filtered_computers = [comp for comp in self.data.computers
                              if "компьютер" in comp.name.lower()]

        computer_browser_set = {(cb.computer_id, cb.browser_id)
                                for cb in self.data.computer_browsers}

        browser_dict = self._get_browsers_dict()

        result = {}
        for computer in filtered_computers:
            related_browsers = []
            for computer_id, browser_id in computer_browser_set:
                if computer_id == computer.id:
                    browser = browser_dict.get(browser_id)
```

```
                if browser:
                    related_browsers.append(browser)
            result[computer.id] = related_browsers

        return result

    def print_task_3(self):
        print("\n ЗАПРОС 3 ")
        result = self.task_3()
        computer_dict = self._get_computers_dict()

        for computer_id, browsers in result.items():
            computer = computer_dict[computer_id]
            print(f"\nКомпьютер: {computer.name}")

            if browsers:
                for browser in browsers:
                    print(f"  - {browser.name} (использует памяти: {browser.memory_usage} МБ)")
            else:
                print("  - Нет связанных браузеров")


def main():
    manager = BrowserManager()
    manager.print_task_1()
    manager.print_task_2()
    manager.print_task_3()
```

```
class TestBrowserManager(unittest.TestCase):

    def setUp(self):
        self.test_computers = [
            Computer(1, "Офисный компьютер"),
            Computer(2, "Игровой компьютер"),
            Computer(3, "Серверный отдел")
        ]

        self.test_browsers = [
            Browser(1, "Chrome", 512, 1),
            Browser(2, "Firefox", 256, 1),
            Browser(3, "Edge", 384, 2),
            Browser(4, "Safari", 192, 3)
        ]

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

        self.data_repository = DataRepository(
            computers=self.test_computers,
```

```
            browsers=self.test_browsers,
            computer_browsers=self.test_computer_browsers
        )

        self.manager = BrowserManager(self.data_repository)

    def test_task_1_group_browsers_by_computer(self):
        result = self.manager.task_1()

        self.assertIsInstance(result, dict)
        self.assertIn(1, result)
        self.assertEqual(len(result[1]), 2)

        browser_names = [browser.name for browser in result[1]]
        self.assertIn("Chrome", browser_names)
        self.assertIn("Firefox", browser_names)

    def test_task_2_total_memory_by_computer(self):
        result = self.manager.task_2()

        self.assertIsInstance(result, list)
        memory_values = [memory for _, memory in result]
        self.assertEqual(memory_values, sorted(memory_values))

        expected_data = {1: 768, 2: 384, 3: 192}
        for computer_id, total_memory in result:
            self.assertEqual(total_memory, expected_data[computer_id])
```

```
    def test_task_3_filter_computers_with_keyword(self):
        result = self.manager.task_3()

        self.assertIsInstance(result, dict)
        expected_computer_ids = {1, 2}
        self.assertEqual(set(result.keys()), expected_computer_ids)
        self.assertNotIn(3, result)


if __name__ == "__main__":
    print("=== Запуск основной программы ===")
    main()

    print("\n\n=== Запуск тестов ===")
    suite = unittest.TestLoader().loadTestsFromTestCase(TestBrowserManager)
    runner = unittest.TextTestRunner(verbosity=2)
    runner.run(suite)
```

```
=== Запуск основной программы ===
 ЗАПРОС 1

Компьютер: Офисный компьютер (ID: 1)
  - Chrome (использует памяти: 512 МБ)
  - Firefox (использует памяти: 256 МБ)

Компьютер: Игровой компьютер (ID: 2)
  - Edge (использует памяти: 384 МБ)
  - Opera (использует памяти: 128 МБ)

Компьютер: Серверный отдел (ID: 3)
  - Safari (использует памяти: 192 МБ)

Компьютер: Главный компьютер (ID: 4)
  - Chrome (использует памяти: 512 МБ)

 ЗАПРОС 2
Компьютер: Серверный отдел - Суммарное использование памяти: 192 МБ
Компьютер: Игровой компьютер - Суммарное использование памяти: 512 МБ
Компьютер: Главный компьютер - Суммарное использование памяти: 512 МБ
Компьютер: Офисный компьютер - Суммарное использование памяти: 768 МБ

 ЗАПРОС 3

Компьютер: Офисный компьютер
  - Firefox (использует памяти: 256 МБ)
  - Chrome (использует памяти: 512 МБ)
  - Edge (использует памяти: 384 МБ)

Компьютер: Игровой компьютер
  - Opera (использует памяти: 128 МБ)
  - Chrome (использует памяти: 512 МБ)
  - Edge (использует памяти: 384 МБ)

Компьютер: Главный компьютер
  - Chrome (использует памяти: 512 МБ)
```

```
=== Запуск тестов ===
test_task_1_group_browsers_by_computer (__main__.TestBrowserManager.test_task_1_
group_browsers_by_computer)
Тест 1: Проверка группировки браузеров по компьютерам ... ok
test_task_2_total_memory_by_computer (__main__.TestBrowserManager.test_task_2_to
tal_memory_by_computer)
Тест 2: Проверка суммарного использования памяти по компьютерам ... ok
test_task_3_filter_computers_with_keyword (__main__.TestBrowserManager.test_task
_3_filter_computers_with_keyword)
Тест 3: Проверка фильтрации компьютеров по ключевому слову ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.083s

OK
```