Вариант 8
Гайдуков Артём
ИБМ3-34Б

Рубежный контроль 2

Проводим рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

```
class HardDisk:
    def __init__(self, id, name, capacity, computer_id=None):
        self.id = id
        self.name = name
        self.capacity = capacity
        self.computer_id = computer_id

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

class ComputerDiskUsage:
    def __init__(self, disk_id, computer_id, usage):
        self.disk_id = disk_id
        self.computer_id = computer_id
        self.usage = usage

def get_computers_with_disks(computers, hard_disks):
    """
    Возвращает словарь, где ключ - имя компьютера, значение - список подключенных
к нему жестких дисков.
    """
    computers_with_disks = {}
    for comp in computers:
        disks = [disk for disk in hard_disks if disk.computer_id == comp.id]
        computers_with_disks[comp.name] = disks
    return computers_with_disks

def get_computers_usage(computers, computer_disk_usages):
    """
    Возвращает словарь, где ключ - имя компьютера, значение - суммарное
использование дисков.
    """
    return {
        comp.name: sum(usage.usage for usage in computer_disk_usages if
usage.computer_id == comp.id)
        for comp in computers
    }
```

```python
def get_computers_with_named_disks(computers, hard_disks, name_filter):
    """
    Возвращает словарь, где ключ - имя компьютера, значение - список дисков,
содержащих в названии name_filter.
    """
    return {
        comp.name: [
            disk for disk in hard_disks if disk.computer_id == comp.id and
name_filter in disk.name
        ]
        for comp in computers
    }
# Данные
computers = [
    Computer(1, "Computer_A"),
    Computer(2, "Computer_B"),
    Computer(3, "Computer_C")
]

hard_disks = [
    HardDisk(1, "Disk_1TB", 1000, 1),
    HardDisk(2, "Disk_500GB", 500, 1),
    HardDisk(3, "Disk_2TB", 2000, 2),
    HardDisk(4, "Disk_1TB_2", 1000, 3)
]
computer_disk_usages = [
    ComputerDiskUsage(1, 1, 300),
    ComputerDiskUsage(2, 1, 200),
    ComputerDiskUsage(3, 2, 1500),
    ComputerDiskUsage(4, 3, 750)
]
# Пример использования функций
if __name__ == "__main__":
    # список компьютеров и подключенных жестких дисков
    computers_with_disks = get_computers_with_disks(computers, hard_disks)
    print("список компьютеров и подключенных жестких дисков:")
    for comp_name, disks in sorted(computers_with_disks.items()):
        print(f"Компьютер: {comp_name}")
        for disk in disks:
            print(f"  Жесткий диск: {disk.name}, Емкость: {disk.capacity}GB")

    # список компьютеров с суммарным объемом использованного места
    computers_usage = get_computers_usage(computers, computer_disk_usages)
    sorted_computers_usage = sorted(computers_usage.items(), key=lambda x: x[1],
reverse=True)
    print("\nсписок компьютеров с суммарным объемом использованного места на
жестких дисках:")
    for comp_name, total_usage in sorted_computers_usage:
        print(f"Компьютер: {comp_name}, Общий использованный объем:
{total_usage}GB")
```

```
    # список компьютеров с жесткими дисками, содержащими "Disk"
    computers_with_named_disks = get_computers_with_named_disks(computers,
hard_disks, "Disk")
    print("\nсписок всех компьютеров с жесткими дисками, содержащими 'Disk' в
названии:")
    for comp_name, disks in computers_with_named_disks.items():
        if disks:
            print(f"Компьютер: {comp_name}")
            for disk in disks:
                print(f"  Жесткий диск: {disk.name}, Емкость: {disk.capacity}GB")
```

Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

```
import unittest
from Control_Task_2 import Computer, HardDisk, ComputerDiskUsage,
get_computers_with_disks, get_computers_usage, get_computers_with_named_disks



class TestDiskManagement(unittest.TestCase):
    def setUp(self):
        self.computers = [
            Computer(1, "Computer_A"),
            Computer(2, "Computer_B"),
            Computer(3, "Computer_C")
        ]
        self.hard_disks = [
            HardDisk(1, "Disk_1TB", 1000, 1),
            HardDisk(2, "Disk_500GB", 500, 1),
            HardDisk(3, "Disk_2TB", 2000, 2),
            HardDisk(4, "Disk_1TB_2", 1000, 3)
        ]
        self.computer_disk_usages = [
            ComputerDiskUsage(1, 1, 300),
            ComputerDiskUsage(2, 1, 200),
            ComputerDiskUsage(3, 2, 1500),
            ComputerDiskUsage(4, 3, 750)
        ]

    def test_get_computers_with_disks(self):
        result = get_computers_with_disks(self.computers, self.hard_disks)
        self.assertIn("Computer_A", result)
        self.assertEqual(len(result["Computer_A"]), 2)
        self.assertEqual(result["Computer_A"][0].name, "Disk_1TB")
```

```
    def test_get_computers_usage(self):
        result = get_computers_usage(self.computers, self.computer_disk_usages)
        self.assertEqual(result["Computer_A"], 500)
        self.assertEqual(result["Computer_B"], 1500)
        self.assertEqual(result["Computer_C"], 750)

    def test_get_computers_with_named_disks(self):
        result = get_computers_with_named_disks(self.computers, self.hard_disks,
"Disk")
        self.assertIn("Computer_A", result)
        self.assertEqual(len(result["Computer_A"]), 2)
        self.assertEqual(result["Computer_B"][0].name, "Disk_2TB")

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

Вывод