Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Рубежный контроль №2
Вариант Е8

Выполнил:

студент группы
РТ5-31Б

Данилова А.С.

Москва, 2024 г

Текст программы

Рефакторинг программы:

```
#Рефакторинг программы

# Определение классов данных
class HardDrive:
    def __init__(self, id, model, capacity):
        self.id = id  # ID жесткого диска
        self.model = model  # Модель жесткого диска
        self.capacity = capacity  # Вместимость жесткого диска в ГБ


class Computer:
    def __init__(self, id, name):
        self.id = id  # ID компьютера
        self.name = name  # Название компьютера
        self.hard_drives = []  # Список жестких дисков, связанных с
компьютером

    def add_hard_drive(self, hard_drive):
        self.hard_drives.append(hard_drive)  # Добавление жесткого диска к
компьютеру


# Функции для запросов
def list_computers_with_hard_drives(computers):
    result = []
    for pc in computers:
        pc_info = {"name": pc.name, "hard_drives": []}
        for hd in pc.hard_drives:
            pc_info["hard_drives"].append({"model": hd.model, "capacity":
hd.capacity})
        result.append(pc_info)
    return result


def average_capacity_per_computer(computers):
    avg_capacities = {}
    for pc in computers:
        if pc.hard_drives:
            total_capacity = sum(hd.capacity for hd in pc.hard_drives)
            avg_capacity = round(total_capacity / len(pc.hard_drives), 2)
            avg_capacities[pc.name] = avg_capacity
    return sorted(avg_capacities.items(), key=lambda x: x[1])


def list_large_hard_drives(computers, threshold=1000):
    return [hd for pc in computers for hd in pc.hard_drives if hd.capacity >
threshold]


# Создание тестовых данных
def create_test_data():
    # Создание жестких дисков
    hd1 = HardDrive(1, "Seagate Barracuda", 1000)
    hd2 = HardDrive(2, "Western Digital Blue", 2000)
    hd3 = HardDrive(3, "Samsung 970 EVO", 500)

    # Создание компьютеров
    pc1 = Computer(1, "Gaming PC")
    pc2 = Computer(2, "Office PC")
```

```
    # Связывание жестких дисков с компьютерами
    pc1.add_hard_drive(hd1)
    pc1.add_hard_drive(hd2)
    pc2.add_hard_drive(hd3)

    return [pc1, pc2]


if __name__ == "__main__":
    computers = create_test_data()
    print("Запрос 1:", list_computers_with_hard_drives(computers))
    print("Запрос 2:", average_capacity_per_computer(computers))
    print("Запрос 3:", list_large_hard_drives(computers))
```

Модульные тесты с применением TDD - фреймворка:

```
#Модульные тесты с использованием TDD

import unittest

from ref import HardDrive, Computer, list_computers_with_hard_drives,
average_capacity_per_computer, list_large_hard_drives, create_test_data

class TestComputerFunctions(unittest.TestCase):

    def setUp(self):
        self.computers = create_test_data()

    def test_list_computers_with_hard_drives(self):
        result = list_computers_with_hard_drives(self.computers)
        self.assertEqual(len(result), 2)
        self.assertEqual(result[0]['name'], "Gaming PC")
        self.assertEqual(len(result[0]['hard_drives']), 2)

    def test_average_capacity_per_computer(self):
        result = average_capacity_per_computer(self.computers)
        self.assertEqual(len(result), 2)
        self.assertEqual(result[0][0], "Gaming PC")
        self.assertAlmostEqual(result[0][1], 1500.0)  # Средняя вместимость
для Gaming PC

    def test_list_large_hard_drives(self):
        result = list_large_hard_drives(self.computers)
        self.assertEqual(len(result), 1)
        self.assertEqual(result[0].model, "Western Digital Blue")

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