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 Е7

Выполнил:

студент группы
РТ5-31Б

Губанова В.Е.

Москва, 2024 г

Текст программы

```
class Microprocessor:
    def __init__(self, name):
        self.name = name
        self.computers = []

    def add_computer(self, computer):
        self.computers.append(computer)

    def average_power(self):
        if not self.computers:
            return 0
        total_power = sum(computer.power for computer in self.computers)
        return total_power / len(self.computers)

    def __str__(self):
        return self.name


class Computer:
    def __init__(self, name, microprocessor, power):
        self.name = name
        self.microprocessor = microprocessor
        self.power = power
        self.microprocessor.add_computer(self)
        self.owners = []

    def __str__(self):
        return f'{self.name} (Power: {self.power})'

    def add_owner(self, owner):
        if owner not in self.owners:
            self.owners.append(owner)
            owner.add_computer(self)

class Owner:
    def __init__(self, name):
        self.name = name
        self.computers = []

    def add_computer(self, computer):
        if computer not in self.computers:
            self.computers.append(computer)

    def __str__(self):
        return self.name

micro1 = Microprocessor("Intel Core i5")
micro2 = Microprocessor("AMD Ryzen 5")
micro3 = Microprocessor("Intel Core i7")


computer1 = Computer("Gaming PC", micro1, 400)
computer2 = Computer("Office PC", micro2, 200)
computer3 = Computer("Workstation", micro3, 600)
computer4 = Computer("Budget PC", micro1, 300)

owner1 = Owner("Alice")
owner2 = Owner("Bob")
owner3 = Owner("Anna")

computer1.add_owner(owner1)
computer2.add_owner(owner2)
```

```
computer3.add_owner(owner1)
computer4.add_owner(owner3)
print('--------------------------------')
#Список миикропроцессорв с "Intel" в названии
intel_microprocessors = [micro for micro in [micro1, micro2, micro3] if
"Intel" in micro.name]

for micro in intel_microprocessors:
    print(f'Микропроцессор: {micro}')
    print('Компьютеры, которые его используют:')
    for computer in micro.computers:
        print(f' - {computer}')


#Сортировка по мощности
print('--------------------------------')
microprocessors = [micro1, micro2, micro3]
average_powers = [(micro, micro.average_power()) for micro in
microprocessors]
average_powers.sort(key=lambda x: x[1])
print("Микропроцессоры с их средней мощностью")
for micro, avg_power in average_powers:
    print(f'Микропроцессор: {micro}, Средняя мощность: {avg_power:.2f}')
print('--------------------------------')


## Выводкомпьютеров с владельцами на "А" и названиями их микропроцессоров
print("компьютеры с владельцами на 'А'")
for owner in [owner1, owner3]:
    for computer in owner.computers:
        print(f'Owner: {owner}, Computer: {computer}')
```

Модульные тесты

```
import unittest
class Test(unittest.TestCase):

    def setUp(self):
        self.micro1 = Microprocessor("Intel Core i5")
        self.micro2 = Microprocessor("AMD Ryzen 5")
        self.computer1 = Computer("Gaming PC", self.micro1, 400)
        self.computer2 = Computer("Office PC", self.micro2, 200)
        self.owner1 = Owner("Alice")

    #Проверяем,что компьютер был добавлен к микропроцессору
    def test_add_computer_to_microprocessor(self):
        self.assertIn(self.computer1, self.micro1.computers)
        self.assertIn(self.computer2, self.micro2.computers)

    # Проверяем расчет средней мощности
    def test_average_power(self):
        self.micro1.add_computer(self.computer1)
        self.micro2.add_computer(self.computer2)
        self.assertEqual(self.micro1.average_power(), 400)
        self.assertEqual(self.micro2.average_power(), 200)

    #Проверяем, что owner был добавлен к компьютеру
    def test_add_owner_to_computer(self):
        self.computer1.add_owner(self.owner1)
        self.assertIn(self.owner1, self.computer1.owners)
        self.assertIn(self.computer1, self.owner1.computers)
```