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 РК №2
Вариант предметной области: 8

Выполнил:
студент группы ИУ5-33Б
Горенков А.А.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

Условия рубежного контроля №2 по курсу ПиК ЯП

Рубежный контроль представляет собой разработку тестов на языке Python.

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

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

Код программы

```
class HardDrive:
    def __init__(self, hd_id, capacity_gb, computer_id):
        self.hd_id = hd_id
        self.capacity_gb = capacity_gb
        self.computer_id = computer_id


class Computer:
    def __init__(self, computer_id, brand, model):
        self.computer_id = computer_id
        self.brand = brand
        self.model = model
        self.hard_drives = []

    def add_hard_drive(self, hard_drive):
        self.hard_drives.append(hard_drive)

    def get_brand_starting_with(self, letter):
        return self.brand.startswith(letter)


def filter_computers_by_brand_starting_with(computers_and_hard_drives,
letter):
    filtered_computers = []
    for computer, hard_drive in computers_and_hard_drives:
        if computer.get_brand_starting_with(letter):
            filtered_computers.append((computer, hard_drive))
    return filtered_computers


def get_computers_with_minimum_capacity(computers_and_hard_drives):
    min_capacity_by_computer = {}
    for computer, hard_drive in computers_and_hard_drives:
        if computer.computer_id in min_capacity_by_computer:
            if hard_drive.capacity_gb <
min_capacity_by_computer[computer.computer_id]:
                min_capacity_by_computer[computer.computer_id] =
hard_drive.capacity_gb
        else:
            min_capacity_by_computer[computer.computer_id] =
hard_drive.capacity_gb

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

```python
def sort_computers_and_hard_drives(computers_and_hard_drives):
    return sorted(computers_and_hard_drives, key=lambda x: (x[0].brand,
x[0].model))


# Модульные тесты
import unittest

class TestComputerMethods(unittest.TestCase):
    def setUp(self):
        self.computer1 = Computer(1, "Dell", "XPS 13")
        self.computer2 = Computer(2, "HP", "Pavilion")
        self.hard_drive1 = HardDrive(1, 512, 1)
        self.hard_drive2 = HardDrive(2, 256, 1)
        self.hard_drive3 = HardDrive(3, 1000, 2)
        self.computers_and_hard_drives = [
            (self.computer1, self.hard_drive1),
            (self.computer1, self.hard_drive2),
            (self.computer2, self.hard_drive3)
        ]

    def test_get_brand_starting_with(self):
        self.assertTrue(self.computer1.get_brand_starting_with("D"))
        self.assertFalse(self.computer2.get_brand_starting_with("D"))

    def test_filter_computers_by_brand_starting_with(self):
        filtered =
filter_computers_by_brand_starting_with(self.computers_and_hard_drives, "D")
        self.assertEqual(len(filtered), 2)
        self.assertEqual(filtered[0][0].brand, "Dell")
        self.assertEqual(filtered[1][0].brand, "Dell")

    def test_get_computers_with_minimum_capacity(self):
        min_capacity_computers =
get_computers_with_minimum_capacity(self.computers_and_hard_drives)
        self.assertEqual(len(min_capacity_computers), 2)
        self.assertEqual(min_capacity_computers[0][1], 256)
        self.assertEqual(min_capacity_computers[1][1], 1000)


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

Результат

```
Launching unittests with arguments python -m unittest rk2.TestComput



Ran 3 tests in 0.001s

OK

Process finished with exit code 0
```