Московский государственный технический университет им. Н.Э. Баумана

Факультет “Радиотехнический”
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2

Вариант предметной области №8: Жесткий диск, Компьютер
Вариант запросов: Е

Выполнил:
студент группы РТ5-31Б:
Зеленева И.Е.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2025 г.

Описание задания

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).

Текст программы

data_classes.py

```
class HardDisk:
    def __init__(self, id, name, capacity, price, computer_id):
        self.id = id
        self.name = name
        self.capacity = capacity
        self.price = price
        self.computer_id = computer_id

    def __repr__(self):
        return f"HardDisk(id={self.id}, name='{self.name}')"


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

    def __repr__(self):
        return f"Computer(id={self.id}, name='{self.name}')"


class HardDiskComputer:
    """Для реализации связи многие-ко-многим"""
    def __init__(self, computer_id, hard_disk_id):
        self.computer_id = computer_id
        self.hard_disk_id = hard_disk_id

    def __repr__(self):
        return f"HardDiskComputer(computer_id={self.computer_id},
hard_disk_id={self.hard_disk_id})"
```

data_service.py

```
from data_classes import Computer, HardDisk, HardDiskComputer
```

```
def create_test_data():
    computers = [
        Computer(1, "Домашний компьютер"),
        Computer(2, "Рабочий"),
        Computer(3, "Игровой компьютер"),
        Computer(4, "Офисный"),
        Computer(5, "Серверный компьютер"),
        Computer(6, "Сломанный")
    ]

    hard_disks = [
        HardDisk(1, "Барракуда 1TB", 1000, 45.90, 1),
        HardDisk(2, "Восток 500GB", 500, 25.50, 2),
        HardDisk(3, "Север 2TB", 2000, 85.00, 3),
        HardDisk(4, "WD Blue 1TB", 1000, 42.80, 3),
        HardDisk(5, "Samsung 870 2TB", 2000, 92.75, 3),
        HardDisk(6, "Toshiba 500GB", 500, 28.30, 4),
        HardDisk(7, "Hitachi 1TB", 1000, 47.50, 5),
        HardDisk(8, "Kingston 2TB", 2000, 88.00, 6)
    ]

    hard_disks_computers = [
        HardDiskComputer(1, 1),
        HardDiskComputer(2, 2),
        HardDiskComputer(3, 3),
        HardDiskComputer(3, 4),
        HardDiskComputer(3, 5),
        HardDiskComputer(4, 6),
        HardDiskComputer(5, 7),
        HardDiskComputer(6, 3),
        HardDiskComputer(6, 4),
        HardDiskComputer(6, 5)
    ]

    return computers, hard_disks, hard_disks_computers


def create_many_to_many_relation(computers, hard_disks, hard_disks_computers):

    many_to_many_temp = [
        (c.name, hdc.computer_id, hdc.hard_disk_id)
```

```
        for c in computers
        for hdc in hard_disks_computers
        if c.id == hdc.computer_id
    ]

    many_to_many = [
        (hd.name, hd.capacity, hd.price, comp_name)
        for comp_name, comp_id, hd_id in many_to_many_temp
        for hd in hard_disks if hd.id == hd_id
    ]

    return many_to_many
```

query_service.py

```
from data_classes import Computer, HardDisk, HardDiskComputer


def create_test_data():
    """Создание тестовых данных"""
    computers = [
        Computer(1, "Домашний компьютер"),
        Computer(2, "Рабочий"),
        Computer(3, "Игровой компьютер"),
        Computer(4, "Офисный"),
        Computer(5, "Серверный компьютер"),
        Computer(6, "Сломанный")
    ]

    hard_disks = [
        HardDisk(1, "Барракуда 1TB", 1000, 45.90, 1),
        HardDisk(2, "Восток 500GB", 500, 25.50, 2),
        HardDisk(3, "Север 2TB", 2000, 85.00, 3),
        HardDisk(4, "WD Blue 1TB", 1000, 42.80, 3),
        HardDisk(5, "Samsung 870 2TB", 2000, 92.75, 3),
        HardDisk(6, "Toshiba 500GB", 500, 28.30, 4),
        HardDisk(7, "Hitachi 1TB", 1000, 47.50, 5),
        HardDisk(8, "Kingston 2TB", 2000, 88.00, 6)
    ]

    hard_disks_computers = [
        HardDiskComputer(1, 1),
        HardDiskComputer(2, 2),
```

```
        HardDiskComputer(3, 3),
        HardDiskComputer(3, 4),
        HardDiskComputer(3, 5),
        HardDiskComputer(4, 6),
        HardDiskComputer(5, 7),
        HardDiskComputer(6, 3),
        HardDiskComputer(6, 4),
        HardDiskComputer(6, 5)
    ]

    return computers, hard_disks, hard_disks_computers


def create_many_to_many_relation(computers, hard_disks, hard_disks_computers):
    many_to_many_temp = [
        (c.name, hdc.computer_id, hdc.hard_disk_id)
        for c in computers
        for hdc in hard_disks_computers
        if c.id == hdc.computer_id
    ]

    many_to_many = [
        (hd.name, hd.capacity, hd.price, comp_name)
        for comp_name, comp_id, hd_id in many_to_many_temp
        for hd in hard_disks if hd.id == hd_id
    ]

    return many_to_many
```

run_tests.py

```
import unittest
import sys

if __name__ == '__main__':
    test_suite = unittest.defaultTestLoader.discover('.', pattern='test_*.py')
    test_runner = unittest.TextTestRunner(verbosity=2)

    result = test_runner.run(test_suite)
    sys.exit(0 if result.wasSuccessful() else 1)
```

tests_queries.py

```
import unittest
from data_service import create_test_data, create_many_to_many_relation
from query_service import task_e1, task_e2, task_e3


class TestQueries(unittest.TestCase):

    def setUp(self):
        self.computers, self.hard_disks, self.hard_disks_computers = create_test_data()
        self.many_to_many = create_many_to_many_relation(
            self.computers, self.hard_disks, self.hard_disks_computers
        )

    def test_task_e1(self):
        result = task_e1(self.computers, self.many_to_many)

        self.assertIsInstance(result, dict)

        expected_computers = ['Домашний компьютер', 'Игровой компьютер', 'Серверный
компьютер']
        self.assertListEqual(sorted(list(result.keys())), sorted(expected_computers))

        self.assertIn('Барракуда 1ТВ', result['Домашний компьютер'])

        self.assertNotIn('Рабочий', result)
        self.assertNotIn('Офисный', result)

    def test_task_e2(self):
        result = task_e2(self.computers, self.many_to_many)

        self.assertIsInstance(result, list)

        prices = [item[1] for item in result]
        self.assertEqual(prices, sorted(prices))

        for item in result:
            self.assertIsInstance(item, tuple)
            self.assertEqual(len(item), 3)  # (имя, средняя цена, количество)
            self.assertIsInstance(item[0], str)  # имя компьютера
            self.assertIsInstance(item[1], float)  # средняя цена
            self.assertIsInstance(item[2], int)  # количество дисков
```

```
    def test_task_e3(self):
        result = task_e3(self.hard_disks, self.many_to_many)

        self.assertIsInstance(result, list)

        for hd_name, hd_price, comp_name in result:
            self.assertTrue(hd_name.startswith('Б'))

        disk_names = [item[0] for item in result]
        self.assertEqual(disk_names, sorted(disk_names))

        expected_result = ('Барракуда 1ТВ', 45.9, 'Домашний компьютер')
        self.assertIn(expected_result, result)


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

main.py

```
from data_service import create_test_data, create_many_to_many_relation
from query_service import task_e1, task_e2, task_e3


def print_results(res1, res2, res3):
    print('Задание Е1')
    print('Список всех компьютеров, у которых в названии присутствует слово
"компьютер", и список их жестких дисков:')

    for computer_name, hard_disks_list in res1.items():
        print(f'\nКомпьютер: {computer_name}')
        for hd_name in hard_disks_list:
            print(f'  Жесткий диск: {hd_name}')

    print('\nЗадание Е2')
    print('Список компьютеров со средней ценой жестких дисков в каждом,
отсортированный по средней цене:')

    for computer_name, avg_price, count in res2:
        print(f'Компьютер: {computer_name}, Средняя цена: {avg_price} руб. (дисков:
{count})')

    print('\nЗадание Е3')
```

```
    print('Список всех жестких дисков, у которых название начинается с буквы "Б", и названия их компьютеров:')

    for hd_name, hd_price, comp_name in res3:
        print(f'Жесткий диск: {hd_name}, Цена: {hd_price} руб. -> Компьютер: {comp_name}')


def main():
    computers, hard_disks, hard_disks_computers = create_test_data()
    many_to_many = create_many_to_many_relation(computers, hard_disks, hard_disks_computers)

    res1 = task_e1(computers, many_to_many)
    res2 = task_e2(computers, many_to_many)
    res3 = task_e3(hard_disks, many_to_many)

    print_results(res1, res2, res3)


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

Примеры выполнения программы

```
C:\Users\User\Репозиторий\rk2>python main.py
Задание E1
Список всех компьютеров, у которых в названии присутствует слово "компьютер", и список их жестких дисков:

Компьютер: Домашний компьютер
  Жесткий диск: Барракуда 1TB

Компьютер: Игровой компьютер
  Жесткий диск: Север 2TB
  Жесткий диск: WD Blue 1TB
  Жесткий диск: Samsung 870 2TB

Компьютер: Серверный компьютер
  Жесткий диск: Hitachi 1TB

Задание E2
Список компьютеров со средней ценой жестких дисков в каждом, отсортированный по средней цене:
Компьютер: Рабочий, Средняя цена: 25.5 руб. (дисков: 1)
Компьютер: Офисный, Средняя цена: 28.3 руб. (дисков: 1)
Компьютер: Домашний компьютер, Средняя цена: 45.9 руб. (дисков: 1)
Компьютер: Серверный компьютер, Средняя цена: 47.5 руб. (дисков: 1)
Компьютер: Игровой компьютер, Средняя цена: 73.52 руб. (дисков: 3)
Компьютер: Сломанный, Средняя цена: 73.52 руб. (дисков: 3)

Задание E3
Список всех жестких дисков, у которых название начинается с буквы "Б", и названия их компьютеров:
Жесткий диск: Барракуда 1TB, Цена: 45.9 руб. -> Компьютер: Домашний компьютер

C:\Users\User\Репозиторий\rk2>python test_queries.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK

C:\Users\User\Репозиторий\rk2>python run_tests.py
test_task_e1 (test_queries.TestQueries.test_task_e1)
Тест задания E1: поиск компьютеров с 'компьютер' в названии ... ok
test_task_e2 (test_queries.TestQueries.test_task_e2)
Тест задания E2: средняя цена дисков по компьютерам ... ok
test_task_e3 (test_queries.TestQueries.test_task_e3)
Тест задания E3: диски, начинающиеся на 'Б' ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
```