

**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 рубежному контролю №2
«Модульное тестирование»

Выполнил:
Студент группы ИУ5-31Б
Куртинец Роман

Проверил:
Гапанюк Ю. Е.

2025 г.

Листинг программы

main.py

```
1 # main.py
2 from typing import List, Any, Callable, Dict
3 from dataclasses import dataclass
4 from collections import defaultdict
5
6 @dataclass
7 class HardDrive:
8     hdd_id: int
9     model: str
10    capacity: int
11    computer_id: int
12
13 @dataclass
14 class Computer:
15     comp_id: int
16     name: str
17
18 @dataclass
19 class HardDriveComputer:
20     hard_drive_id: int
21     computer_id: int
22
23 def get_computers() -> List[Computer]:
24     """ Функция для получения данных компьютеров. """
25     return [
26         Computer(1, "Компьютер Dell XPS"),
27         Computer(2, "Компьютер HP Pavilion"),
28         Computer(3, "Сервер Lenovo"),
29         Computer(4, "Компьютер Apple MacBook")
30     ]
31
32 def get_hard_drives() -> List[HardDrive]:
33     """ Функция для получения данных жёстких дисков. """
34     return [
35         HardDrive(1, "Seagate 1TB", 1000, 1),
36         HardDrive(2, "WD 500GB", 500, 1),
37         HardDrive(3, "Samsung 2TB", 2000, 2),
38         HardDrive(4, "Crucial 1TB SSD", 1000, 3),
39         HardDrive(5, "Toshiba 750GB", 750, 2)
40     ]
```

```
41
42 def get_hard_drive_computers() -> List[HardDriveComputer]:
43     """ Функция для получения связей многие ко многим. """
44     return [
45         HardDriveComputer(1, 1),
46         HardDriveComputer(1, 4),
47         HardDriveComputer(2, 1),
48         HardDriveComputer(3, 2),
49         HardDriveComputer(4, 3),
50         HardDriveComputer(5, 2),
51         HardDriveComputer(5, 4)
52     ]
53
54 def print_data(data: List[Any], headers: List[str], title: str, column_width: int = 30) -> None:
55     """
56     Функция для вывода данных в виде таблицы.
57     Принимает данные, заголовки столбцов и заголовок таблицы.
58     """
59     total_length = len(headers) * column_width
60     columns = len(headers)
61
62     print(f"title:={total_length}")
63     print((":{<{column_width}}" * columns).format(*headers, column_width=column_width))
64     print()
65     print("\n".join(
66         (":{<{column_width}}" * columns).format(*i, column_width=column_width) for i in data
67     ))
68     print()
69
70 def first_query(computers: List[Computer], hard_drives: List[HardDrive]) -> List[Any]:
71     """ Реализация первого запроса: Связанные объекты (один-ко-многим). """
72     computer_dict = {c.comp_id: c for c in computers}
73
74     joined = [(computer_dict[hd.computer_id].name, hd.model)
75               for hd in hard_drives
76               if hd.computer_id in computer_dict]
77
78     return sorted(joined, key=lambda x: x[0])
79
80 def second_query(computers: List[Computer], hard_drives: List[HardDrive]) -> List[Any]:
```

```
81     """ Реализация второго запроса: Суммарная ёмкость (один-ко-многим). """
82     computer_dict = {c.comp_id: c for c in computers}
83
84     capacity_sum = defaultdict(int)
85     for hd in hard_drives:
86         capacity_sum[hd.computer_id] += hd.capacity
87
88     comp_capacities = [(computer_dict[cid].name, total)
89                         for cid, total in capacity_sum.items()
90                         if cid in computer_dict]
91
92     return sorted(comp_capacities, key=lambda x: x[1], reverse=True)
93
94     def third_query(computers: List[Computer], hard_drives: List[HardDrive],
95                      relations: List[HardDriveComputer], condition: Callable) -> List[Any]:
96         """ Реализация третьего запроса: Фильтрация по условию (многие-ко-многим). """
97         filtered_computers = [c for c in computers
98                               if condition(c.name.lower())]
99
100        hard_drive_dict = {hd.hdd_id: hd for hd in hard_drives}
101
102        comp_hds = defaultdict(list)
103        for link in relations:
104            hd = hard_drive_dict.get(link.hard_drive_id)
105            if hd:
106                comp_hds[link.computer_id].append(hd)
107
108        result = []
109        for comp in filtered_computers:
110            for hd in comp_hds.get(comp.comp_id, []):
111                result.append((comp.name, hd.model, hd.capacity))
112
113        return result
114
115    def main() -> None:
116        computers = get_computers()
117        hard_drives = get_hard_drives()
118        relations = get_hard_drive_computers()
119
120        # Первый запрос
121        print_data(first_query(computers, hard_drives),
122                   ["Компьютер", "Жёсткий диск"], "Запрос 1")
123
124        # Второй запрос
125        print_data(second_query(computers, hard_drives),
126                   ["Компьютер", "Суммарная ёмкость (ГБ)"], "Запрос 2")
127
128        # Третий запрос
129        print_data(third_query(computers, hard_drives, relations,
130                               lambda name: "компьютер" in name),
131                               ["Компьютер", "Жёсткий диск", "Ёмкость (ГБ)"], "Запрос 3")
132
133    if __name__ == "__main__":
134        main()
135
```

```
test_main.py
1 # test_main.py
2 import pytest
3 from pytest_unordered import unordered
4
5 from main import (
6     HardDrive, Computer, HardDriveComputer,
7     first_query, second_query, third_query
8 )
9
10 @pytest.fixture
11 def test_one_to_many_data():
12     computers = [
13         Computer(1, "Компьютер А"),
14         Computer(2, "Компьютер В"),
15         Computer(3, "Сервер С"),
16     ]
17     hard_drives = [
18         HardDrive(1, "HDD1", 1000, 1),
19         HardDrive(2, "HDD2", 500, 1),
20         HardDrive(3, "HDD3", 2000, 2),
21         HardDrive(4, "HDD4", 750, 3),
22         HardDrive(5, "HDD5", 1000, 2),
23     ]
24     return computers, hard_drives
25
26 @pytest.fixture
27 def test_many_to_many_data():
28     computers = [
29         Computer(1, "Компьютер А"),
30         Computer(2, "Компьютер В"),
31         Computer(3, "Сервер С"),
32         Computer(4, "Компьютер D"),
33     ]
34     hard_drives = [
35         HardDrive(1, "HDD1", 1000, -1),
36         HardDrive(2, "HDD2", 500, -1),
37         HardDrive(3, "HDD3", 2000, -1),
38         HardDrive(4, "HDD4", 750, -1),
39         HardDrive(5, "HDD5", 1000, -1),
40     ]
```

```
41     relations = [
42         HardDriveComputer(1, 1),
43         HardDriveComputer(1, 2),
44         HardDriveComputer(2, 1),
45         HardDriveComputer(3, 2),
46         HardDriveComputer(4, 3),
47         HardDriveComputer(5, 2),
48         HardDriveComputer(5, 4),
49         HardDriveComputer(3, 4),
50         HardDriveComputer(2, 3),
51     ]
52     return computers, hard_drives, relations
53
54 def test_first_query(test_one_to_many_data):
55     computers, hard_drives = test_one_to_many_data
56     expected = [
57         ("Компьютер А", "HDD1"),
58         ("Компьютер А", "HDD2"),
59         ("Компьютер В", "HDD3"),
60         ("Компьютер В", "HDD5"),
61         ("Сервер С", "HDD4"),
62     ]
63     result = first_query(computers, hard_drives)
64     assert result == expected
65
66 def test_second_query(test_one_to_many_data):
67     computers, hard_drives = test_one_to_many_data
68     expected = [
69         ("Компьютер В", 3000),
70         ("Компьютер А", 1500),
71         ("Сервер С", 750),
72     ]
73     result = second_query(computers, hard_drives)
74     assert result == expected
75
76 def test_third_query(test_many_to_many_data):
77     computers, hard_drives, relations = test_many_to_many_data
```

```
78    expected = unordered([
79        ("Компьютер А", "HDD1", 1000),
80        ("Компьютер А", "HDD2", 500),
81        ("Компьютер В", "HDD1", 1000),
82        ("Компьютер В", "HDD3", 2000),
83        ("Компьютер В", "HDD5", 1000),
84        ("Компьютер Д", "HDD3", 2000),
85        ("Компьютер Д", "HDD5", 1000),
86    ])
87    result = third_query(computers, hard_drives, relations,
88                          lambda name: "компьютер" in name)
89    assert result == expected
90
```

Результат выполнения

```
(venv) at0r@Roma-M14:~/Kurtinets_Labs_3sem$ pytest -v
=====
platform linux -- Python 3.12.3, pytest-9.0.2, pluggy-1.6.0 -- /home/at0r/Kurtinets_Labs_3sem/venv/bin/python3
cachedir: .pytest_cache
rootdir: /home/at0r/Kurtinets_Labs_3sem
plugins: unordered-0.7.0
collected 3 items

test_main.py::test_first_query PASSED
test_main.py::test_second_query PASSED
test_main.py::test_third_query PASSED

===== 3 passed in 0.03s =====
```