МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “ Парадигмы и конструкции языков программирования”

Отчет по рубежному контролю №2

Выполнил:
Студент группы ИУ5-36Б
Бусаров К.Д.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

main.py

RK2 > main.py > main

```python
from operator import itemgetter
from dataclasses import dataclass
from typing import List, Dict, Tuple, Optional

@dataclass
class Computer:
    """Компьютер"""
    id: int
    model: str
    price: int
    display_class_id: int

@dataclass
class DisplayClass:
    """Дисплейный класс"""
    id: int
    name: str

@dataclass
class ComputerDisplayClass:
    """Компьютеры в дисплейных классах для связи многие-ко-многим"""
    display_class_id: int
    computer_id: int


class ComputerService:
    """Сервис для работы с компьютерами и дисплейными классами"""

    def __init__(self,
                 computers: List[Computer],
                 display_classes: List[DisplayClass],
                 computer_display_classes: Optional[List[ComputerDisplayClass]] = None):
```

```python
    def __init__(self,
                 computers: List[Computer],
                 display_classes: List[DisplayClass],
                 computer_display_classes: Optional[List[ComputerDisplayClass]] = None):
        self.computers = computers
        self.display_classes = display_classes
        self.computer_display_classes = computer_display_classes or []

    def create_one_to_many_relation(self) -> List[Tuple[Computer, DisplayClass]]:
        """Создает связь один-ко-многим между компьютерами и дисплейными классами"""
        return [
            (computer, display_class)
            for display_class in self.display_classes
            for computer in self.computers
            if computer.display_class_id == display_class.id
        ]

    def get_computers_in_class_a(self) -> Dict[str, List[Computer]]:
        """
        Задание Г1: Возвращает словарь с дисплейными классами на 'A'
        и списками компьютеров в них
        """
        one_to_many = self.create_one_to_many_relation()
        result = {}

        for computer, display_class in one_to_many:
            if display_class.name.startswith('A'):
                if display_class.name not in result:
                    result[display_class.name] = []
                result[display_class.name].append(computer)
class ComputerService:
    def get_computers_in_class_a(self) -> Dict[str, List[Computer]]:
        Задание Г1: Возвращает словарь с дисплейными классами на 'A'
        и списками компьютеров в них
        """
        one_to_many = self.create_one_to_many_relation()
        result = {}

        for computer, display_class in one_to_many:
            if display_class.name.startswith('A'):
                if display_class.name not in result:
                    result[display_class.name] = []
                result[display_class.name].append(computer)

        return dict(sorted(result.items()))

    def get_max_price_per_class(self) -> List[Tuple[str, int]]:
        """
        Задание Г2: Возвращает отсортированный список кортежей
        (название класса, максимальная цена)
        """
        one_to_many = self.create_one_to_many_relation()
        class_computers = {}

        for computer, display_class in one_to_many:
            if display_class.name not in class_computers:
                class_computers[display_class.name] = []
            class_computers[display_class.name].append(computer.price)

        max_prices = {
            class_name: max(prices)
```

```python
    def get_max_price_per_class(self) -> List[Tuple[str, int]]:
            class_name: max(prices)
            for class_name, prices in class_computers.items()
        }

        return sorted(max_prices.items(), key=itemgetter(1), reverse=True)

    def get_all_computer_display_connections(self) -> Dict[str, List[Computer]]:
        """
        Задание Г3: Возвращает все связи компьютеров с дисплейными классами
        """
        # Связи один-ко-многим
        one_to_many = self.create_one_to_many_relation()
        all_connections = list(one_to_many)

        # Добавляем связи многие-ко-многим
        for cdc in self.computer_display_classes:
            computer = next((c for c in self.computers if c.id == cdc.computer_id), None)
            display_class = next((dc for dc in self.display_classes
                                  if dc.id == cdc.display_class_id), None)

            if computer and display_class:
                connection_exists = any(
                    conn[0].id == computer.id and conn[1].id == display_class.id
                    for conn in all_connections
                )
                if not connection_exists:
                    all_connections.append((computer, display_class))

        # Инициализируем результат ВСЕМИ классами (даже пустыми)
        result = {dc.name: [] for dc in self.display_classes}

        # Заполняем компьютерами
        for computer, display_class in all_connections:
            result[display_class.name].append(computer)

        # Сортируем компьютеры внутри каждого класса по убыванию цены
        for class_name in result:
            result[class_name].sort(key=lambda x: x.price, reverse=True)

        # Сортируем по названию класса
        return dict(sorted(result.items()))


def create_sample_data():
    """Создает тестовые данные"""
    display_classes = [
        DisplayClass(1, 'А-101'),
        DisplayClass(2, 'Б-202'),
        DisplayClass(3, 'А-303'),
        DisplayClass(4, 'В-404'),
        DisplayClass(5, 'А-505'),
    ]

    computers = [
        Computer(1, 'Dell Optiplex', 50000, 1),
        Computer(2, 'HP ProDesk', 45000, 2),
```

```
        Computer(2, 'HP ProDesk', 45000, 2),
        Computer(3, 'Lenovo ThinkCentre', 60000, 3),
        Computer(4, 'Acer Veriton', 40000, 3),
        Computer(5, 'Asus ExpertCenter', 55000, 3),
        Computer(6, 'Fujitsu Esprimo', 48000, 5),
        Computer(7, 'MSI Pro', 52000, 5),
    ]

    computers_display_classes = [
        ComputerDisplayClass(1, 3),
        ComputerDisplayClass(2, 5),
    ]
                    (variable) display_classes: list[DisplayClass]
    return computers, display_classes, computers_display_classes


def main():
    """Основная функция программы"""
    computers, display_classes, computers_display_classes = create_sample_data()
    service = ComputerService(computers, display_classes, computers_display_classes)

    print('Задание Г1')
    print('Список всех дисплейных классов, у которых название начинается с буквы «А», и список компьютеров в них:')

    result_1 = service.get_computers_in_class_a()
    for class_name, comp_list in result_1.items():
        print(f"\n{class_name}:")
        for comp in comp_list:
            print(f"  - {comp.model} (цена: {comp.price} руб.)")

    print('\n' + '='*60)
    print('Задание Г2')
    print('Список дисплейных классов с максимальной ценой компьютеров в каждом классе, отсортированный по максимальной цене:')

    result_2 = service.get_max_price_per_class()
    for class_name, max_price in result_2:
        print(f"{class_name}: {max_price} руб.")

    print('\n' + '='*60)
    print('Задание Г3')
    print('Список всех связанных компьютеров и дисплейных классов, отсортированный по классам:')

    result_3 = service.get_all_computer_display_connections()
    for class_name, comp_list in result_3.items():
        print(f"\n{class_name}:")
        if comp_list:
            for comp in comp_list:
                print(f"  - {comp.model} (цена: {comp.price} руб.)")
        else:
            print(f"  (нет компьютеров)")


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

```
kirill@WIN-OHQ18823BK3:~/Programming_Paradigms/RK2$ python3 main.py
Задание Г1
Список всех дисплейных классов, у которых название начинается с буквы «А», и список компьютеров в них:

А-101:
  - Dell Optiplex (цена: 50000 руб.)

А-303:
  - Lenovo ThinkCentre (цена: 60000 руб.)
  - Acer Veriton (цена: 40000 руб.)
  - Asus ExpertCenter (цена: 55000 руб.)

А-505:
  - Fujitsu Esprimo (цена: 48000 руб.)
  - MSI Pro (цена: 52000 руб.)

============================================================
Задание Г2
Список дисплейных классов с максимальной ценой компьютеров в каждом классе, отсортированный по максимальной цене:
А-303: 60000 руб.
А-505: 52000 руб.
А-101: 50000 руб.
Б-202: 45000 руб.

============================================================
Задание Г3
Список всех связанных компьютеров и дисплейных классов, отсортированный по классам:

А-101:
  - Lenovo ThinkCentre (цена: 60000 руб.)
  - Dell Optiplex (цена: 50000 руб.)

А-303:
  - Lenovo ThinkCentre (цена: 60000 руб.)
  - Asus ExpertCenter (цена: 55000 руб.)
  - Lenovo ThinkCentre (цена: 60000 руб.)
  - Dell Optiplex (цена: 50000 руб.)

А-303:
  - Lenovo ThinkCentre (цена: 60000 руб.)
  - Asus ExpertCenter (цена: 55000 руб.)
  - Acer Veriton (цена: 40000 руб.)

А-505:
  - MSI Pro (цена: 52000 руб.)
  - Fujitsu Esprimo (цена: 48000 руб.)

Б-202:
  - Asus ExpertCenter (цена: 55000 руб.)
  - HP ProDesk (цена: 45000 руб.)

В-404:
  (нет компьютеров)
kirill@WIN-OHQ18823BK3:~/Programming_Paradigms/RK2$ python3 test.py
test_get_all_computer_display_connections (__main__.TestComputerService.test_get_all_computer_display_connections)
Тест для задания Г3: ... ok
test_get_computers_in_class_a (__main__.TestComputerService.test_get_computers_in_class_a)
Тест для задания Г1: ... ok
test_get_max_price_per_class (__main__.TestComputerService.test_get_max_price_per_class)
Тест для задания Г2: ... ok
test_empty_data (__main__.TestComputerServiceEdgeCases.test_empty_data)
Тест с пустыми данными ... ok
test_no_class_a (__main__.TestComputerServiceEdgeCases.test_no_class_a)
Тест когда нет классов на букву 'А' ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
```