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
Вариант Д4

Выполнил:
Студент группы ИУ5-36Б
Канаев А.И.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

Листинг кода

main.py

```
import operator


class DisplayClass:
    """Display"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Computer:
    """Comp"""
    def __init__(self, id, name, cost, class_id):
        self.id = id
        self.name = name
        self.cost = cost
        self.class_id = class_id

class CompClass:
    """'Компьютеры в классах' для связи многие-ко-многим"""
    def __init__(self, comp_id, class_id):
        self.comp_id = comp_id
        self.class_id = class_id


classes = [
    DisplayClass(1, 'Аудитория 501'),
    DisplayClass(2, 'Лаборатория сетей'),
    DisplayClass(3, 'Аудитория 333'),
]

computers = [
    Computer(1, 'HP Pavilion 15s', 95000, 1),
    Computer(2, 'MacBook Pro M4', 250000, 2),
    Computer(3, 'Dell XPS 13', 150000, 1),
    Computer(4, 'Asus ROG Zephyrus', 180000, 3),
    Computer(5, 'Lenovo IdeaPad s340s', 80000, 3),
]

comps_classes = [
    CompClass(1, 1), CompClass(2, 2), CompClass(3, 1),
    CompClass(4, 3), CompClass(5, 3),
    CompClass(2, 1), # мас в 1 классе
]

def get_task_d1(computers, classes):
    """
    Список компьютеров, название которых заканчивается на 's', и их классов.
    Возвращает список кортежей (название_компа, название_класса).
    """
    return [
        (comp.name, cls.name)
        for comp in computers
        for cls in classes
        if comp.class_id == cls.id and comp.name.endswith('s')
    ]

def get_task_d2(computers, classes):
    """
    Список классов со средней стоимостью компьютеров, отсортированный по убыванию средней цены.
    Возвращает список кортежей (название_класса, средняя_цена).
    """
```

```
    res_d2 = []
    for cls in classes:
        # Фильтруем компы текущего класса
        class_comps = list(filter(lambda x: x.class_id == cls.id, computers))

        if len(class_comps) > 0:
            # Считаем среднюю цену
            avg_cost = sum([comp.cost for comp in class_comps]) / len(class_comps)
            res_d2.append((cls.name, int(avg_cost)))

    # Сортировка по убыванию цены
    return sorted(res_d2, key=lambda item: item[1], reverse=True)

def get_task_d3(computers, classes, comps_classes):
    """
    Список классов, начинающихся на 'A', и список компьютеров в них (многие-ко-многим).
    Возвращает словарь {класс: [список компьютеров]}.
    """
    # {id: Computer}
    comps_dict = {comp.id: comp for comp in computers}
    # {id: DisplayClass}
    classes_dict = {cls.id: cls for cls in classes}

    many_to_many = {}

    for cc in comps_classes:
        cls_obj = classes_dict.get(cc.class_id)
        comp_obj = comps_dict.get(cc.comp_id)

        # Защита от битых ссылок
        if not cls_obj or not comp_obj:
            continue

        class_name = cls_obj.name
        comp_name = comp_obj.name

        if class_name not in many_to_many:
            many_to_many[class_name] = []
        many_to_many[class_name].append(comp_name)

    # Фильтр по названию класса (начинается на 'A')
    return {key: val for key, val in many_to_many.items() if key.startswith('A')}

def main():
    print('Задание Д1')
    print(get_task_d1(computers, classes))
    print()

    print('Задание Д2')
    print(get_task_d2(computers, classes))
    print()

    print('Задание Д3')
    print(get_task_d3(computers, classes, comps_classes))

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

test_main.py

```
import unittest

from main import Computer, DisplayClass, CompClass, get_task_d1, get_task_d2, get_task_d3

class TestComputerTasks(unittest.TestCase):
    def setUp(self):
        """Создаем тестовые данные перед каждым тестом"""
```

```
        self.classes = [
            DisplayClass(1, 'Аудитория 100'),
            DisplayClass(2, 'Буфет'),
            DisplayClass(3, 'Актовый зал'), # Класс без компьютеров (для проверки Д2)
        ]

        self.computers = [
            Computer(1, 'TestComp_s', 1000, 1),  # Заканчивается на 's', класс 1
            Computer(2, 'TestComp_x', 2000, 1),  # НЕ заканчивается на 's', класс 1
            Computer(3, 'SuperPC_s', 5000, 2),   # Заканчивается на 's', класс 2
        ]

        self.comps_classes = [
            CompClass(1, 1),
            CompClass(2, 1),
            CompClass(3, 2),
            CompClass(3, 1), # Компьютер 3 также связан с классом 1 (многие-ко-многим)
        ]

    def test_task_d1(self):
        """
        Тест Д1: Проверяет фильтрацию по окончанию имени на 's'
        """
        result = get_task_d1(self.computers, self.classes)


        expected = [
            ('TestComp_s', 'Аудитория 100'),
            ('SuperPC_s', 'Буфет')
        ]


        self.assertCountEqual(result, expected) # пофиг на порядок

    def test_task_d2(self):
        """
        Тест Д2: Проверяет расчет средней цены и сортировку
        """
        result = get_task_d2(self.computers, self.classes)


        expected = [
            ('Буфет', 5000),         # 1 место (5000 > 1500)
            ('Аудитория 100', 1500)  # 2 место
        ]

        self.assertEqual(result, expected)

    def test_task_d3(self):
        """
        Тест Д3: Проверяет связь многие-ко-многим и фильтр по 'А'
        """
        result = get_task_d3(self.computers, self.classes, self.comps_classes)

        # 'Аудитория 100' (начинается на А):
        # - Связана с комп 1, 2, 3 (через comps_classes)
        # 'Буфет' (НЕ начинается на А):
        # - Игнорируем
        # 'Актовый зал':
        # - Нет связей

        # В self.comps_classes у нас:
        # (1,1) -> TestComp_s в Ауд.100
        # (2,1) -> TestComp_x в Ауд.100
        # (3,2) -> SuperPC_s в Буфет (игнор)
        # (3,1) -> SuperPC_s в Ауд.100

        expected_keys = ['Аудитория 100']
        self.assertEqual(list(result.keys()), expected_keys)

        # Проверим состав списка для Аудитории 100
```

```
        expected_comps = ['TestComp_s', 'TestComp_x', 'SuperPC_s']
        self.assertCountEqual(result['Аудитория 100'], expected_comps)

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

Результаты тестов

```
  pcpl/RK_2  on 🌱 main [?]
❯ python3 -m unittest test_main.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```