

**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 запросов: В  
Вариант предметной области: 4

Выполнил:  
студентка группы ИУ5-33Б  
Беспалова Виктория

Проверил:  
преподаватель каф. ИУ5  
Гапанюк Ю. Е.

Москва, 2023 г.

## Решение варианта В для предметной области 4

1. «Дисплейный класс» и «Компьютер» связаны соотношением один-ко-многим. Выведите список всех компьютеров, у которых название производителя начинается с буквы «А», и номера дисплейных классов, в которых они находятся.
2. «Дисплейный класс» и «Компьютер» связаны соотношением один-ко-многим. Выведите список дисплейных классов с наиболее ранним годом выпуска компьютеров, отсортированный по наиболее раннему году выпуска.
3. «Дисплейный класс» и «Компьютер» связаны соотношением многие-ко-многим. Выведите список всех связанных компьютеров и дисплейных классов, отсортированный по компьютерам, сортировка по дисплейным классам произвольная.

### Задание

- 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

#### Файл main.py

```
class Computer:
    """Компьютер"""

    def __init__(self, id: int, manufacturer_name: str, manufacture_year: int, class_id: int):
        self._id = id
        self._manufacturer_name = manufacturer_name
        self._manufacture_year = manufacture_year
        self._class_id = class_id

    @property
    def id(self) -> int:
        return self._id

    @property
    def class_id(self) -> int:
        return self._class_id

    @property
    def manufacturer_name(self) -> str:
        return self._manufacturer_name

    @property
    def manufacture_year(self) -> int:
        return self._manufacture_year
```

```

class ComputerClassroom:
    """Дисплейный класс"""

    def __init__(self, id: int, number: str):
        self._id = id
        self._number = number

    @property
    def id(self) -> int:
        return self._id

    @property
    def number(self) -> str:
        return self._number


class ClassroomsComputers:
    """
    'Компьютеры класса' для реализации
    связи многие-ко-многим
    """

    def __init__(self, computer_id: int, classroom_id: int):
        self._computer_id = computer_id
        self._classroom_id = classroom_id

    @property
    def computer_id(self) -> int:
        return self._computer_id

    @property
    def classroom_id(self) -> int:
        return self._classroom_id


def generate_data():
    # Дисплейные классы
    computer_classrooms = [
        ComputerClassroom(1, "254л"),
        ComputerClassroom(2, "253л"),
        ComputerClassroom(3, "306э"),
        ComputerClassroom(4, "362"),
        ComputerClassroom(5, "107л")
    ]

    # Компьютеры
    computers = [
        Computer(1, "Lenovo", 2022, 1),
        Computer(2, "Lenovo", 2020, 1),
        Computer(3, "Acer", 2019, 2),
        Computer(4, "Lenovo", 2021, 3),
        Computer(5, "Acer", 2020, 3),
        Computer(6, "Asus", 2017, 4),
        Computer(7, "Apple", 2020, 5)
    ]

    classrooms_computers = [
        ClassroomsComputers(1, 1),
        ClassroomsComputers(2, 1),
        ClassroomsComputers(3, 2),
        ClassroomsComputers(4, 3),

```

```

    ClassroomsComputers(5, 3),
    ClassroomsComputers(6, 4),
    ClassroomsComputers(7, 5)
]
return computer_classrooms, computers, classrooms_computers

def task1(computer_classrooms: list[ComputerClassroom], computers: list[Computer]):
    data = [(computer, classroom) for computer in computers for classroom in computer_classrooms if
             computer.class_id == classroom.id and computer.manufacturer_name.startswith("A")]
    return [(computer.manufacturer_name, classroom.number) for computer, classroom in data]

def task2(computer_classrooms: list[ComputerClassroom], computers: list[Computer]):
    data = {}
    for computer_classroom in computer_classrooms:
        classroom_manufacture_years = [computer.manufacture_year for computer in computers for classroom in
                                         computer_classrooms if
                                         computer.class_id == classroom.id and classroom.id == computer_classroom.id]
        data[computer_classroom.number] = min(classroom_manufacture_years)

    data_items = list(data.items())
    data_items.sort(key=lambda x: x[1])
    return [(classroom, earliest_year) for classroom, earliest_year in data_items]

def task3(computer_classrooms: list[ComputerClassroom], computers: list[Computer],
           classrooms_computers: list[ClassroomsComputers]):
    data = [(computer, computer_classroom) for cc in classrooms_computers for computer in computers for
             computer_classroom in computer_classrooms if
             cc.computer_id == computer.id and cc.classroom_id == computer_classroom.id]

    data.sort(key=lambda x: x[0].manufacturer_name)
    return [(computer.manufacturer_name, classroom.number) for computer, classroom in data]

def execute_tasks(computer_classrooms, computers, classrooms_computers):
    # Задача(запрос) 1
    print("Запрос № 1.\nСписок компьютеров, у которых название производителя начинается с буквы \"A\", \n"
          "и номеров дисплейных классов в которых они находятся.")
    for computer_manufacturer_name, classroom_number in task1(computer_classrooms, computers):
        print(computer_manufacturer_name, classroom_number)
    print()

    # Задача(запрос) 2
    print("Запрос № 2.\nСписок дисплейных классов с наиболее ранним годом выпуска компьютеров, \nотсортированный
по "
          "наиболее раннему году выпуска.")
    for (classroom, earliest_year) in task2(computer_classrooms, computers):
        print(classroom, earliest_year)
    print()

    # Задача(запрос) 3
    print("Запрос № 3.\nСписок всех связанных компьютеров и дисплейных классов, отсортированный по компьютерам, "
          "\nсортировка по дисплейным классам произвольная.")
    for (computer_manufacturer_name, classroom_number) in task3(computer_classrooms, computers, classrooms_computers):
        print(computer_manufacturer_name, classroom_number)
    print()

def main():
    # Генерация данных
    computer_classrooms, computers, classrooms_computers = generate_data()

```

```

# Выполнение всех задач(запросов)
execute_tasks(computer_classrooms, computers, classrooms_computers)

if __name__ == "__main__":
    main()

```

## Файл TDDtests.py

```

import unittest
from main import *

# Тестирование класса "Компьютер"
class TestComputer(unittest.TestCase):

    def test_computer_creation(self):
        computer = Computer(1, "Lenovo", 2022, 1)
        self.assertEqual(computer.id, 1)
        self.assertEqual(computer.manufacturer_name, "Lenovo")
        self.assertEqual(computer.manufacture_year, 2022)
        self.assertEqual(computer.class_id, 1)

# Тестирование класса "Дисплейный класс"
class TestComputerClassroom(unittest.TestCase):

    def test_computer_classroom_creation(self):
        computer_classroom = ComputerClassroom(1, "254л")
        self.assertEqual(computer_classroom.id, 1)
        self.assertEqual(computer_classroom.number, "254л")

# Тестирование класса для реализации связи многие-ко-многим "Компьютеры класса"
class TestClassroomsComputers(unittest.TestCase):

    def test_classrooms_computers_creation(self):
        classrooms_computers = ClassroomsComputers(1, 1)
        self.assertEqual(classrooms_computers.computer_id, 1)
        self.assertEqual(classrooms_computers.classroom_id, 1)

#
class TestTaskExecution(unittest.TestCase):
    def setUp(self):
        self.computer_classrooms, self.computers, self.classrooms_computers = generate_data()

    # Тестирование запроса №1
    def test_task1(self):
        result = task1(self.computer_classrooms, self.computers)
        self.assertEqual(result, [("Acer", "253л"), ("Acer", "306э"), ("Asus", "362"), ("Apple", "107л")])

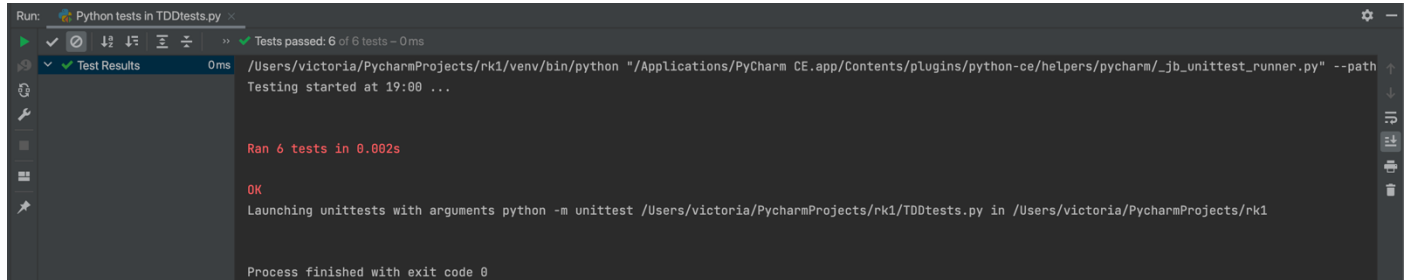
    # Тестирование запроса №2
    def test_task2(self):
        result = task2(self.computer_classrooms, self.computers)
        self.assertEqual(result, [("362", 2017), ("253л", 2019), ("254л", 2020), ("306э", 2020), ("107л", 2020)])

    # Тестирование запроса №3
    def test_task3(self):
        result = task3(self.computer_classrooms, self.computers, self.classrooms_computers)
        self.assertEqual(result,

```

```
[("Acer", "253л"), ("Acer", "306э"), ("Apple", "107л"), ("Asus", "362"), ("Lenovo", "254л"),  
("Lenovo", "254л"), ("Lenovo", "306э"))]  
  
if __name__ == "__main__":  
    unittest.main()
```

## Результат выполнения программы



The screenshot shows the PyCharm Run window for a Python test. The top status bar indicates "Tests passed: 6 of 6 tests - 0 ms". The "Test Results" tab is active, showing a list of 6 tests, all of which are passed. The output console shows the following text:

```
Testing started at 19:00 ...  
  
Ran 6 tests in 0.002s  
  
OK  
Launching unittests with arguments python -m unittest /Users/victoria/PycharmProjects/rk1/TDDtests.py in /Users/victoria/PycharmProjects/rk1  
  
Process finished with exit code 0
```