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: Д
Вариант предметной области: 4

Выполнил:
студент группы ИУ5-35Б
Вопияшин Никита

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

Решение варианта Д для предметной области 4

1. «Компьютер» и «Дисплейный класс» связаны соотношением один-ко-многим. Выведите список всех компьютеров, у которых название фирмы заканчивается на «о», и номера аудиторий, в которых они находятся.
2. «Компьютер» и «Дисплейный класс» связаны соотношением один-ко-многим. Выведите список аудиторий со средней годом выпуска компьютеров в каждой аудитории, отсортированный по среднему возрасту (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений*).
3. «Компьютер» и «Дисплейный класс» связаны соотношением многие-ко-многим. Выведите список всех аудиторий, у которых номер начинается с цифры «2», и список названий фирм находящихся в них компьютеров.

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:
    def __init__(self, _id, _name, _year, _classroom_id):
        self.id = _id
        self.name = _name
        self.year = _year
        self.classroom_id = _classroom_id


class Classroom:
    def __init__(self, id_number, _classroom_number):
        self.id = id_number
        self.classroom_number = _classroom_number


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


def generate_data():
    # Аудитории
    classrooms = [
        Classroom(1, "254л"),
        Classroom(2, "253л"),
        Classroom(3, "306э"),
        Classroom(4, "362"),
        Classroom(5, "107л")
    ]

    # Компьютеры
```

```python
    computers = [
        Computer(1, "Lenovo", 2022, 1),
        Computer(2, "Cisco", 2020, 1),
        Computer(3, "Acer", 2019, 2),
        Computer(4, "Lenovo", 2021, 3),
        Computer(5, "Cisco", 2018, 3),
        Computer(6, "Asus", 2017, 4),
        Computer(7, "Apple", 2020, 5)
    ]

    # Связь многие ко многим
    classrooms_computers = [
        ClassroomsComputers(1, 1),
        ClassroomsComputers(2, 1),
        ClassroomsComputers(3, 2),
        ClassroomsComputers(4, 3),
        ClassroomsComputers(5, 3),
        ClassroomsComputers(6, 4),
        ClassroomsComputers(7, 5)
    ]
    return classrooms, computers, classrooms_computers


def task1(classrooms: list, computers: list):
    one_to_many = [(c.name, c.year, cc.classroom_number)
                   for cc in classrooms
                   for c in computers
                   if c.classroom_id == cc.id]
    res_11 = []
    for computer_name, year, classroom_num in one_to_many:
        if computer_name.endswith('o'):
            res_11.append((computer_name, classroom_num))
    return res_11


def task2(classrooms: list, computers: list):
    one_to_many = [(c.name, c.year, cc.classroom_number)
                   for cc in classrooms
                   for c in computers
                   if c.classroom_id == cc.id]
    res_12 = {}
    for cc in classrooms:
        cc_computers = list(filter(lambda i: i[2] == cc.classroom_number, one_to_many))
        if len(cc_computers) > 0:
            l_books_years = [x for _, x, _ in cc_computers]
            res_12[cc.classroom_number] = int(sum(l_books_years) / len(l_books_years))
    return sorted(res_12.items(), key=lambda item: item[1])


def task3(classrooms: list, computers: list,
          classrooms_computers: list):
    many_to_many_temp = [(cc.classroom_number, ccs.classroom_id, ccs.computer_id)
                         for cc in classrooms
                         for ccs in classrooms_computers
                         if cc.id == ccs.classroom_id]

    many_to_many = [(c.name, c.year, classroom_name)
                    for classroom_name, classroom_id, computer_id in many_to_many_temp
                    for c in computers if c.id == computer_id]

    res_13 = {}
    for cc in classrooms:
        if cc.classroom_number[0] == '2':
```

```
        cc_computers = list(filter(lambda i: i[2] == cc.classroom_number, many_to_many))
        cc_computers_names = [x for x, _, _ in cc_computers]
        res_13[cc.classroom_number] = cc_computers_names
    return res_13


def main():
    classrooms, computers, classrooms_computers = generate_data()

    # Фирмы компьютеров, заканчивающиеся на 'о', и их аудитории
    print('Запрос 1')
    print(task1(classrooms, computers))
    print('\n')

    # средний год создания компьютера в аудитории
    print('Запрос 2')
    print(task2(classrooms, computers))
    print('\n')

    # кабинеты и их компьютеры, начинающиеся с цифры 2
    print('Запрос 3')
    print(task3(classrooms, computers, classrooms_computers))


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

Файл test.py

```
import unittest
from main import *


class TestComputer(unittest.TestCase):
    def test_computer_creation(self):
        computer = Computer(1, "Lenovo", 2077, 1)
        self.assertEqual(computer.id, 1)
        self.assertEqual(computer.name, "Lenovo")
        self.assertEqual(computer.year, 2077)
        self.assertEqual(computer.classroom_id, 1)


class TestClassroom(unittest.TestCase):
    def test_classroom_creation(self):
        classroom = Classroom(1, "254л")
        self.assertEqual(classroom.id, 1)
        self.assertEqual(classroom.classroom_number, "254л")


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
```

```python
    def test_task1(self):
        result = task1(self.computer_classrooms, self.computers)
        self.assertEqual(result, [("Lenovo", "254л"), ("Cisco", "254л"), ("Lenovo", "306э"), ("Cisco", "306э")])

    # Тестирование запроса №2
    def test_task2(self):
        result = task2(self.computer_classrooms, self.computers)
        self.assertEqual(result, [("362", 2017), ("253л", 2019), ("306э", 2019), ("107л", 2020), ("254л", 2021)])

    # Тестирование запроса №3
    def test_task3(self):
        result = task3(self.computer_classrooms, self.computers, self.classrooms_computers)
        self.assertEqual(result,
                         {'254л': ['Lenovo', 'Cisco'], '253л': ['Acer']})


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

Результат выполнения программы

```
Testing started at 23:32 ...
Launching unittests with arguments python -m unittest C:/Users/nikiv/P



Ran 6 tests in 0.041s

OK

Process finished with exit code 0
```