**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: |  | Проверил: |
| студент группы ИУ5-35Б |  | преподаватель каф. ИУ5 |
| Вопияшин Никита |  | 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 «o», 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  
 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  
 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**

