**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: B

Вариант предметной области: 4

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студентка группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Беспалова Виктория |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 г.

**Решение варианта B для предметной области 4**

# «Дисплейный класс» и «Компьютер» связаны соотношением один-ко-многим. Выведите список всех компьютеров, у которых название производителя начинается с буквы «A», и номера дисплейных классов, в которых они находятся.

# «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.

# «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 \"А\",\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 класслв, 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 реаизации связи многие-ко-многим "Компютеры 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**

