Міністерство освіти і науки України

Харківський національний університет ім. В. Н. Каразіна

Факультет комп’ютерних наук

Кафедра безпеки інформаційних систем і технологій

Лабораторна робота №10

*з навчальної дисципліни*

«Спеціалізовані мови програмування та проектування електронних елементів і систем»

Виконав:

Студент групи КБ-41

Кравченко Є.М.

Перевірив:

Олійников Р.В.

Харків – 2020 р.

1.Разработать класс “Товар”, в котором предусмотреть стоимость, срок реализации и уникальный код (буквенно-цифровой, обычно называемый “артикул”). Все поля объекта делать immutable (неизменяемые; они инициализируются только конструктором и внутри класса объявлены как private final). Реализовать очередь на основе ArrayDeque<> (https://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html), в которую внести несколько наименований товара на реализацию. Сформировать еще несколько наименований товара, добавить в начало или в конец очереди в зависимости от срока реализации. Распечатать содержимое очереди, извлекая все элементы, начиная с начала или с конца.

**public class** Product **implements** Comparable<Product>, Serializable {  
 **private final** String **article**;  
 **private final int cost**;  
 **private final int realize**;  
  
 **public** Product(String article, **int** cost, **int** realize) {  
 **this**.**article** = article;  
 **this**.**cost** = cost;  
 **this**.**realize** = realize;  
 }  
  
 **public** String getArticle() {  
 **return article**;  
 }  
  
 **public int** getCost() {  
 **return cost**;  
 }  
  
 **public int** getRealize() {  
 **return realize**;  
 }  
  
 @Override  
 **public int** compareTo(Product o) {  
 **return realize** - o.getRealize();  
 }  
  
 @Override  
 **public** String toString() {  
 **return new** StringJoiner(**", "**, Product.**class**.getSimpleName() + **"["**, **"]"**)  
 .add(**"article='"** + **article** + **"'"**)  
 .add(**"cost="** + **cost**)  
 .add(**"realize="** + **realize**)  
 .toString();  
 }  
}

**private static void** task1() {  
 ArrayDeque<Product> products = **new** ArrayDeque<>();  
 products.add(**new** Product(**"123qwe"**, 123, 3));  
 products.addFirst(**new** Product(**"12qw12"**, 231, 1));  
 products.addLast(**new** Product(**"123qwe1"**, 23, 3));  
 products.addLast(**new** Product(**"123qwe3"**, 342, 4));  
 products.addFirst(**new** Product(**"123qwe2"**, 100, 2));  
 **for** (Product p : products) {  
 System.***out***.println(p);  
 }  
}

2.Реализовать предыдущее задание на основе PriorityQueue<> (https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html). Приоритет - срок реализации.

**private static void** task2() {  
 PriorityQueue<Product> products = **new** PriorityQueue<>();  
 products.add(**new** Product(**"123qwe"**, 123, 3));  
 products.add(**new** Product(**"12qw12"**, 231, 1));  
 products.add(**new** Product(**"123qwe1"**, 23, 3));  
 products.add(**new** Product(**"123qwe3"**, 342, 4));  
 products.add(**new** Product(**"123qwe2"**, 100, 2));  
 **for** (Product p : products) {  
 System.***out***.println(p);  
 }  
}

3.Для класса “Товар” из задания 1 разработать отображение (Map) артикула в сам товар (поиск товара по артикулу). Создать базу из нескольких товаров, вывести на экран результаты поиска одного товара по артикулу и саму базу.

**private static void** task3() {  
 Map<String, Product> map = **new** HashMap<>();  
 **for** (**int** i = 0; i < 10; i++) {  
 Product product = **new** Product(**"qwerty"** + i, 10 \* i, i);  
 map.put(product.getArticle(), product);  
 }  
 System.***out***.println(map.get(**"qwerty5"**));  
}

4.Для баз из двух разных магазинов (на основе задания 3) создать общую. Базы могут содержать коллизионные значения, когда одинаковым артикулам соответствуют разные товары. Проверить на конфликты, сделать общую базу для бесколлизионных артикулов. Для коллизионных создать отдельное мультимножество. Распечатать исходные базы, объединенную базу и мультимножество коллизионных товаров (артикул->ArrayList<Good>).

**private static void** task4() {  
 Map<String, Product> map1 = **new** HashMap<>();  
 **for** (**int** i = 0; i < 10; i++) {  
 Product product = **new** Product(**"qwerty"** + i, 10 \* i, i);  
 map1.put(product.getArticle(), product);  
 }  
 Map<String, Product> map2 = **new** HashMap<>();  
 **for** (**int** i = 5; i < 15; i++) {  
 Product product = **new** Product(**"qwerty"** + i, 10 \* i, i);  
 map2.put(product.getArticle(), product);  
 }  
 Map<String, List<Product>> map = **new** HashMap<>();  
 **for** (Map.Entry<String, Product> entry : map1.entrySet()) {  
 **if** (!map.containsKey(entry.getKey())) {  
 List<Product> arr = **new** ArrayList<>();  
 arr.add(entry.getValue());  
 map.put(entry.getKey(), arr);  
 } **else** {  
 map.get(entry.getKey()).add(entry.getValue());  
 }  
 }  
 **for** (Map.Entry<String, Product> entry : map2.entrySet()) {  
 **if** (!map.containsKey(entry.getKey())) {  
 List<Product> arr = **new** ArrayList<>();  
 arr.add(entry.getValue());  
 map.put(entry.getKey(), arr);  
 } **else** {  
 map.get(entry.getKey()).add(entry.getValue());  
 }  
 }  
 System.***out***.println(map);  
}

5.Объединенную базу из предыдущего задания распечатать по возрастанию срока годности товара.

**private static void** task5() {  
 Map<String, Product> map1 = **new** HashMap<>();  
 **for** (**int** i = 0; i < 10; i++) {  
 Product product = **new** Product(**"qwerty"** + i, 10 \* i, i + 1);  
 map1.put(product.getArticle(), product);  
 }  
 Map<String, Product> map2 = **new** HashMap<>();  
 **for** (**int** i = 15; i > 5; i--) {  
 Product product = **new** Product(**"qwerty"** + i, 10 \* i, i);  
 map2.put(product.getArticle(), product);  
 }  
 Map<String, List<Product>> map = **new** HashMap<>();  
 **for** (Map.Entry<String, Product> entry : map1.entrySet()) {  
 **if** (!map.containsKey(entry.getKey())) {  
 List<Product> arr = **new** ArrayList<>();  
 arr.add(entry.getValue());  
 map.put(entry.getKey(), arr);  
 } **else** {  
 map.get(entry.getKey()).add(entry.getValue());  
 }  
 }  
 **for** (Map.Entry<String, Product> entry : map2.entrySet()) {  
 **if** (!map.containsKey(entry.getKey())) {  
 List<Product> arr = **new** ArrayList<>();  
 arr.add(entry.getValue());  
 map.put(entry.getKey(), arr);  
 } **else** {  
 map.get(entry.getKey()).add(entry.getValue());  
 }  
 }  
 System.***out***.println(map);  
 **for** (List<Product> l : map.values()) {  
 l.sort((Comparator.*comparingInt*(Product::getRealize)));  
 }  
 System.***out***.println(map);  
}

6.Разработать класс “Магазин”, в котором предусмотреть методы: получения полного списка товаров (объектов класса из задания 1) и их количества в наличии (хранение количества реализовать через WeakHashMap); поиск товара по артикулу, вывод их количества в магазине.

Все поля класса делать исключительно private. Если предоставляется внешний доступ к внутренней коллекции (по полному списку товаров), выдавать их только через немодифицируемые коллекции.

**private static void** task6() {  
 Market market = **new** Market();  
 **for** (**int** i = 0; i < 20; i++) {  
 market.addProduct(**new** Product(**"qwerty"** + i, i \* 5 + 1, i + 3));  
 }  
 **for** (**int** i = 0; i < 5; i++) {  
 market.addProduct(**new** Product(**"qwerty"** + i, i \* 5 + 1, i + 3));  
 }  
 System.***out***.println(market);  
}

**public class** Market {  
 **private** Map<String, Product> **products**;  
 **private** Map<String, Integer> **counts**;  
  
 **public** Market() {  
 **products** = **new** HashMap<>();  
 **counts** = **new** WeakHashMap<>();  
 }  
  
 **public void** addProduct(Product product) {  
 **products**.put(product.getArticle(), product);  
 **counts**.merge(product.getArticle(), 1, Integer::*sum*);  
  
 }  
  
 **public int** getCount(String article) {  
 **return counts**.get(article);  
 }  
  
 **public** Product getProduct(String article) {  
 **return products**.get(article);  
 }  
  
 **public** Collection<Product> getProducts() {  
 Collection<Product> collection = **products**.values();  
 **return** Collections.*unmodifiableCollection*(collection);  
 }  
  
 @Override  
 **public** String toString() {  
 StringJoiner stringJoiner = **new** StringJoiner(**";"** + System.*lineSeparator*(), **""**, **""**);  
 **for** (Product p : **products**.values()) {  
 stringJoiner.add(p.toString() + **" count: "** + **counts**.get(p.getArticle()).toString());  
 }  
 **return** stringJoiner.toString();  
 }  
}