МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №3  
по дисциплине «Корпоративные информационные системы»

Тема: «События»

Вариант №14

Выполнил: Сафонов Г.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180330916)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180330917)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 5](#_Toc180330918)

[КОД ПРОГРАММЫ 6](#_Toc180330919)

[ВЫВОД 11](#_Toc180330920)

# **ПОСТАНОВКА ЗАДАЧИ**

Создать систему взаимодействующих классов: класс, хранящий список товаров (Наименование, цена за единицу); класс, рассчитывающий количество каждого наименования товара; класс, рассчитывающий цену каждого наименования товара; класс, рассчитывающий общую цену всех товаров. Классы расчёта, должны срабатывать при каждом добавлении или удалении товаров.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В рамках выполнения лабораторной работы была разработана система для управления товарами с использованием событийной модели. В начале работы были созданы аннотации @AddedProduct и @RemovedProduct, которые используются для маркировки событий добавления и удаления товаров соответственно. Эти аннотации позволяют отслеживать изменения в списке товаров и инициировать соответствующие реакции в других компонентах системы.

Основным классом системы является ProductManager, который управляет списком товаров. Он содержит методы для добавления и удаления товаров, а также инициирует события при изменении списка. При добавлении нового товара вызывается событие productAddedEvent, которое передает обновленный список товаров другим компонентам системы. Аналогично, при удалении товара вызывается событие productRemovedEvent, которое также оповещает другие компоненты.

Кроме того, были созданы три класса для обработки этих событий:

1. **ProductPriceCalculator** — обновляет и выводит цены каждого товара при добавлении или удалении товаров.
2. **ProductQuantityCalculator** — подсчитывает и выводит количество каждого типа товара после изменения списка.
3. **TotalPriceCalculator** — вычисляет и выводит общую стоимость всех товаров в списке.

В заключительной части программы создается несколько объектов товаров, которые затем добавляются и удаляются из списка. Система в реальном времени обрабатывает эти изменения и выводит соответствующую информацию о количестве товаров, их ценах и общей стоимости. Вся обработка событий осуществляется автоматически с использованием CDI-контейнера.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

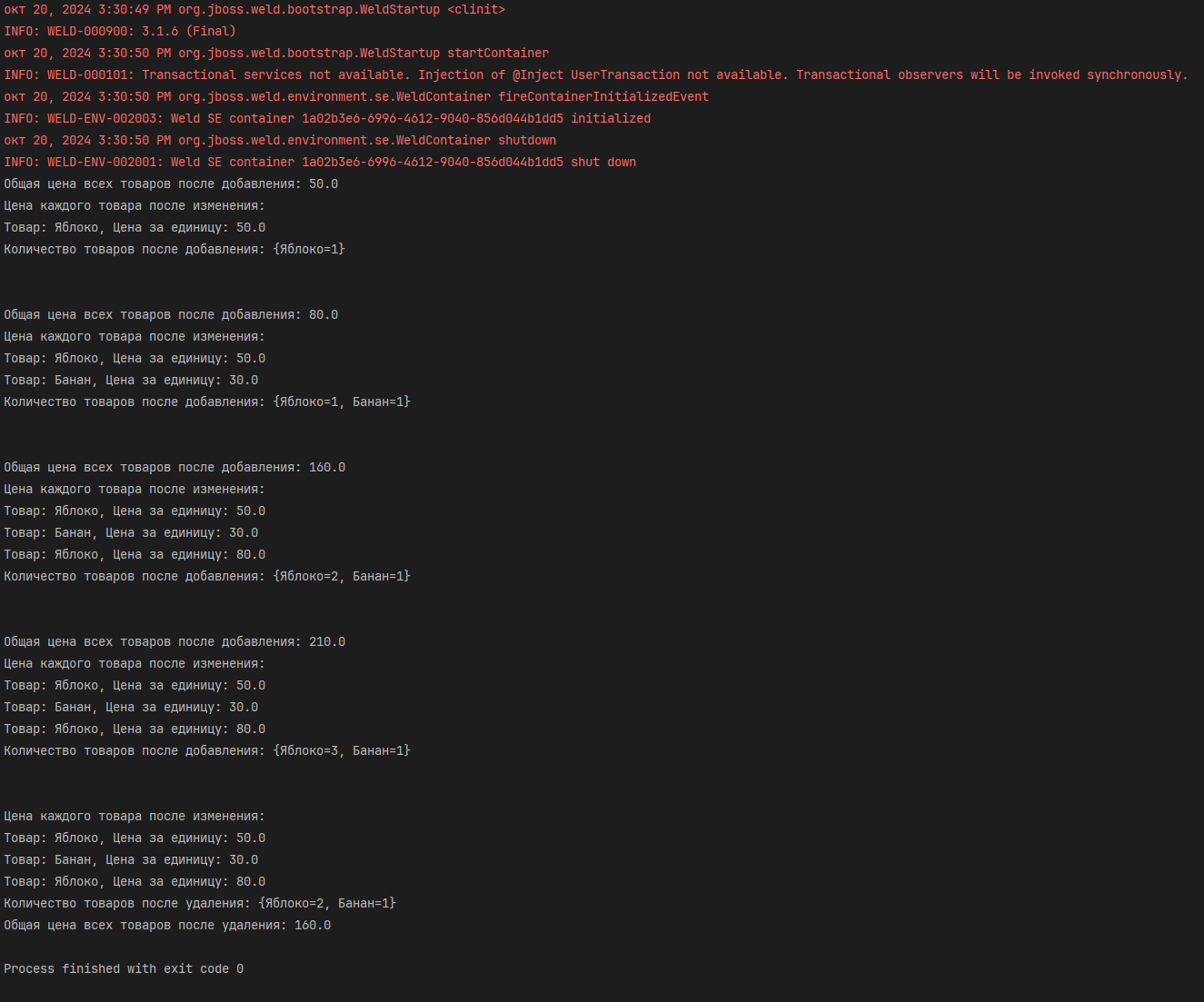


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab3\_Event.Annotation;  
  
import javax.inject.Qualifier;  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Qualifier  
@Retention(RetentionPolicy.RUNTIME)  
@Target({ElementType.FIELD, ElementType.TYPE, ElementType.METHOD, ElementType.PARAMETER})  
public @interface AddedProduct {  
}

package Lab3\_Event.Annotation;  
  
import javax.inject.Qualifier;  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Qualifier  
@Retention(RetentionPolicy.RUNTIME)  
@Target({ElementType.FIELD, ElementType.TYPE, ElementType.METHOD, ElementType.PARAMETER})  
public @interface RemovedProduct {  
}

package Lab3\_Event.Util;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
  
@Getter  
@AllArgsConstructor  
@NoArgsConstructor  
public class Product {  
 private String name; // Наименование товара  
 private double pricePerUnit; // Цена за единицу товара  
  
 @Override  
 public String toString() {  
 return "Товар: " + name + ", Цена за единицу: " + pricePerUnit;  
 }  
}

package Lab3\_Event.Util;  
  
import Lab3\_Event.Annotation.AddedProduct;  
import Lab3\_Event.Annotation.RemovedProduct;  
import lombok.Getter;  
  
import javax.enterprise.event.Event;  
import javax.inject.Inject;  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class ProductManager {  
  
 @Getter  
 private List<Product> productList = new ArrayList<>(); // Список товаров  
  
 @Inject  
 @AddedProduct  
 private Event<List<Product>> productAddedEvent; // Событие добавления  
  
 @Inject  
 @RemovedProduct  
 private Event<List<Product>> productRemovedEvent; // Событие удаления  
  
 public void addProduct(Product product) {  
 productList.add(product); // Добавление товара в список  
 productAddedEvent.fire(new ArrayList<>(productList)); // Вызов события добавления  
 }  
  
 public void removeProduct(Product product) {  
 if (productList.contains(product)) {  
 productList.remove(product); // Удаление товара из списка  
 productRemovedEvent.fire(new ArrayList<>(productList)); // Вызов события удаления  
 } else {  
 System.out.println("Товар " + product.getName() + " не найден в списке.");  
 }  
 }  
}

package Lab3\_Event.Util;  
  
import Lab3\_Event.Annotation.AddedProduct;  
import Lab3\_Event.Annotation.RemovedProduct;  
  
import javax.enterprise.event.Observes;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class ProductPriceCalculator {  
  
 private final Map<Product, Double> productPrices = new HashMap<>(); // Карта с уникальными экземплярами товаров и их ценами  
  
 // Метод для обработки события добавления товара  
 public void calculatePriceOnAdd(@Observes @AddedProduct List<Product> productList) {  
 updatePrices(productList);  
 printProductPrices();  
 }  
  
 // Метод для обработки события удаления товара  
 public void calculatePriceOnRemove(@Observes @RemovedProduct List<Product> productList) {  
 updatePrices(productList);  
 printProductPrices();  
 }  
  
 // Метод для обновления цен товаров  
 private void updatePrices(List<Product> productList) {  
 productPrices.clear();  
 for (Product product : productList) {  
 productPrices.put(product, product.getPricePerUnit());  
 }  
 }  
  
 // Метод для вывода цен товаров  
 private void printProductPrices() {  
 System.out.println("Цена каждого товара после изменения:");  
 for (Map.Entry<Product, Double> entry : productPrices.entrySet()) {  
 Product product = entry.getKey();  
 System.out.println(product);  
 }  
 }  
}

package Lab3\_Event.Util;  
  
import Lab3\_Event.Annotation.AddedProduct;  
import Lab3\_Event.Annotation.RemovedProduct;  
  
import javax.enterprise.event.Observes;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class ProductQuantityCalculator {  
  
 private final Map<String, Integer> productQuantities = new HashMap<>(); // Количество товаров  
  
 // Метод для обработки события добавления товара  
 public void calculateQuantityOnAdd(@Observes @AddedProduct List<Product> productList) {  
 updateQuantities(productList);  
 System.out.println("Количество товаров после добавления: " + productQuantities);  
 }  
  
 // Метод для обработки события удаления товара  
 public void calculateQuantityOnRemove(@Observes @RemovedProduct List<Product> productList) {  
 updateQuantities(productList);  
 System.out.println("Количество товаров после удаления: " + productQuantities);  
 }  
  
 // Метод для обновления количества товаров  
 private void updateQuantities(List<Product> productList) {  
 productQuantities.clear();  
 for (Product product : productList) {  
 productQuantities.put(product.getName(), productQuantities.getOrDefault(product.getName(), 0) + 1);  
 }  
 }  
}

package Lab3\_Event.Util;  
  
import Lab3\_Event.Annotation.AddedProduct;  
import Lab3\_Event.Annotation.RemovedProduct;  
  
import javax.enterprise.event.Observes;  
import java.util.List;  
  
public class TotalPriceCalculator {  
  
 private double totalPrice = 0.0; // Общая цена всех товаров  
  
 // Метод для обработки события добавления товара  
 public void calculateTotalPriceOnAdd(@Observes @AddedProduct List<Product> productList) {  
 totalPrice = calculateTotal(productList);  
 System.out.println("Общая цена всех товаров после добавления: " + totalPrice);  
 }  
  
 // Метод для обработки события удаления товара  
 public void calculateTotalPriceOnRemove(@Observes @RemovedProduct List<Product> productList) {  
 totalPrice = calculateTotal(productList);  
 System.out.println("Общая цена всех товаров после удаления: " + totalPrice);  
 }  
  
 // Метод для вычисления общей цены товаров  
 private double calculateTotal(List<Product> productList) {  
 double sum = 0.0;  
 for (Product product : productList) {  
 sum += product.getPricePerUnit();  
 }  
 return sum;  
 }  
}

package Lab3\_Event;  
  
import Lab3\_Event.Util.Product;  
import Lab3\_Event.Util.ProductManager;  
import org.jboss.weld.environment.se.Weld;  
import org.jboss.weld.environment.se.WeldContainer;  
  
public class Laboratory3 {  
 protected static Weld weld;  
  
 protected static WeldContainer container;  
  
 public static void main(String[] args) {  
 weld = new Weld();  
 container = weld.initialize();  
  
 try {  
 ProductManager productManager = container.select(ProductManager.class).get();  
  
 Product product1 = new Product("Яблоко", 50.0);  
 Product product2 = new Product("Банан", 30.0);  
 Product product3 = new Product("Яблоко", 80.0);  
  
  
 productManager.addProduct(product1);  
 System.out.println("\n");  
 productManager.addProduct(product2);  
 System.out.println("\n");  
 productManager.addProduct(product3); // Добавляем еще одно яблоко с другой ценой  
 System.out.println("\n");  
 productManager.addProduct(product1); // Добавляем еще одно яблоко  
 System.out.println("\n");  
  
 productManager.removeProduct(product1); // Удаляем одно яблоко  
  
 } finally {  
 weld.shutdown();  
 }  
 }  
}

# **ВЫВОД**

В ходе выполнения работы была успешно реализована система управления товарами с использованием событийной модели на базе технологии CDI. События, инициируемые при изменении списка товаров, позволяют автоматически обновлять и выводить информацию о ценах, количестве и общей стоимости товаров. Это демонстрирует гибкость и расширяемость системы, что является важным аспектом при проектировании корпоративных информационных систем.