МИНИСТЕРСТВО ТРАНСПОРТА РФ

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

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №2  
по дисциплине «Корпоративные информационные системы»

Тема: «Декоратор»

Вариант №15

Выполнил: Дронов А. А.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180325126)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180325127)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 5](#_Toc180325128)

[КОД ПРОГРАММЫ 6](#_Toc180325129)

[ВЫВОД 9](#_Toc180325130)

# **ПОСТАНОВКА ЗАДАЧИ**

Создать интерфейс Пицца. Создать класс ПростаяПицца, содержащий основные свойства пиццы. Создать декоратор ПиццаССыром, дополняющий класс ПростаяПицца сыром. Создать декоратор ПиццаСГрибами, дополняющий класс ПростаяПицца грибами. Создать декоратор ПиццаСПерцем, дополняющий класс ПростаяПицца перцем.  Продемонстрировать операции с различным набором декораторов.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Эта лабораторная работа демонстрирует использование шаблона "Декоратор" в сочетании с механизмами CDI (Contexts and Dependency Injection). Работа построена на концепции добавления различных ингредиентов к базовой пицце с помощью декораторов.

Код начинается с интерфейса Pizza, который определяет метод getDescription. Этот метод возвращает описание пиццы и служит базовым контрактом для реализации декораторов.

Класс SimplePizza реализует интерфейс Pizza и представляет собой базовую пиццу с описанием "Простая пицца". Эта реализация помечена аннотацией @Default, что делает её основной реализацией для CDI.

Класс PizzaBean использует внедрение зависимостей для получения объекта Pizza. Метод getDescription этого класса просто возвращает описание текущего объекта Pizza.

Три декоратора (PizzaWithCheese, PizzaWithMushrooms, PizzaWithPepper) реализуют интерфейс Pizza и оборачивают базовый объект пиццы, добавляя к описанию дополнительный ингредиент: сыр, грибы или перец соответственно. Эти декораторы помечены аннотацией @Decorator и используют аннотацию @Delegate для указания базового объекта, который они декорируют.

Конфигурационный файл beans.xml определяет порядок применения декораторов. В данном случае сначала применяется декоратор PizzaWithCheese, затем PizzaWithPepper, и, наконец, PizzaWithMushrooms. CDI автоматически применяет декораторы в указанной последовательности.

Класс Laboratory2 является точкой входа в программу. Он создает CDI-контейнер с помощью библиотеки Weld и получает объект PizzaBean. Когда вызывается метод getDescription у PizzaBean, CDI подставляет объект Pizza с применёнными декораторами. В результате на консоль выводится описание пиццы, включающее все добавленные ингредиенты.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

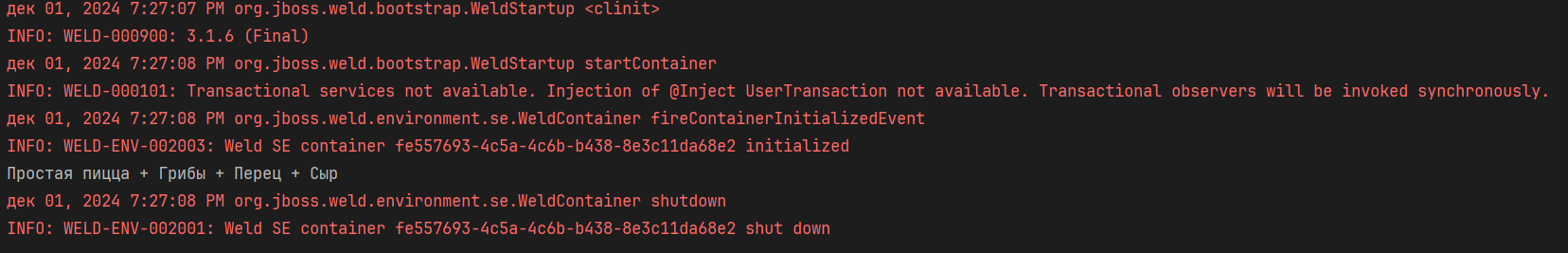


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab2\_Decorator.Abstraction;  
  
public interface Pizza {  
 String getDescription();  
}

package Lab2\_Decorator.Bean;  
  
import Lab2\_Decorator.Abstraction.Pizza;  
  
import javax.inject.Inject;  
  
public class PizzaBean {  
  
 @Inject  
 private Pizza pizza;  
  
 public String getDescription() {  
 return pizza.getDescription();  
 }  
}

package Lab2\_Decorator.Bean;  
  
import Lab2\_Decorator.Abstraction.Pizza;  
import lombok.NoArgsConstructor;  
  
import javax.enterprise.inject.Default;  
  
@Default  
@NoArgsConstructor  
public class SimplePizza implements Pizza {  
  
 @Override  
 public String getDescription() {  
 return "Простая пицца";  
 }  
}

package Lab2\_Decorator.Decorator;  
  
import Lab2\_Decorator.Abstraction.Pizza;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
  
@Decorator  
@AllArgsConstructor  
@NoArgsConstructor  
public class PizzaWithCheese implements Pizza {  
  
 private String additionalIngredient = " + Сыр";  
  
 @Inject  
 @Delegate  
 private Pizza pizza;  
  
 @Override  
 public String getDescription() {  
 String baseDescription = pizza.getDescription();  
  
 return baseDescription + additionalIngredient;  
 }  
}

package Lab2\_Decorator.Decorator;  
  
import Lab2\_Decorator.Abstraction.Pizza;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
  
@Decorator  
@AllArgsConstructor  
@NoArgsConstructor  
public class PizzaWithMushrooms implements Pizza {  
  
 private String additionalIngredient = " + Грибы";  
  
 @Delegate  
 @Inject  
 private Pizza pizza;  
  
 @Override  
 public String getDescription() {  
 String baseDescription = pizza.getDescription();  
  
 return baseDescription + additionalIngredient;  
 }  
}

package Lab2\_Decorator.Decorator;  
  
import Lab2\_Decorator.Abstraction.Pizza;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
  
@Decorator  
@AllArgsConstructor  
@NoArgsConstructor  
public class PizzaWithPepper implements Pizza {  
  
 private String additionalIngredient = " + Перец";  
  
 @Delegate  
 @Inject  
 private Pizza pizza;  
  
 @Override  
 public String getDescription() {  
 String baseDescription = pizza.getDescription();  
  
 return baseDescription + additionalIngredient;  
 }  
}

package Lab2\_Decorator;  
  
import Lab2\_Decorator.Bean.PizzaBean;  
import org.jboss.weld.environment.se.Weld;  
import org.jboss.weld.environment.se.WeldContainer;  
  
public class Laboratory2 {  
 protected static Weld weld;  
 protected static WeldContainer container;  
  
 public static void main(String[] args) {  
 weld = new Weld();  
 container = weld.initialize();  
  
 try {  
  
 PizzaBean pizzaBean = container.select(PizzaBean.class).get();  
  
 System.out.println(pizzaBean.getDescription());  
  
 } finally {  
 weld.shutdown();  
 }  
 }  
}

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee  
 http://xmlns.jcp.org/xml/ns/javaee/beans\_1\_1.xsd"  
 bean-discovery-mode="all">  
  
 <decorators>  
 <class>Lab2\_Decorator.Decorator.PizzaWithCheese</class>  
 <class>Lab2\_Decorator.Decorator.PizzaWithPepper</class>  
 <class>Lab2\_Decorator.Decorator.PizzaWithMushrooms</class>  
 </decorators>  
</beans>

# **ВЫВОД**

Работа демонстрирует реализацию гибкого подхода к добавлению функциональности с помощью шаблона "Декоратор". Использование CDI позволяет легко управлять зависимостями и расширять функционал без изменения исходного кода базовых классов. Такая структура кода упрощает добавление новых декораторов, улучшает модульность и способствует повторному использованию компонентов.