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 №4

Выполнил: Иванов Н. С.

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 6](#_Toc180325128)

[КОД ПРОГРАММЫ 6](#_Toc180325129)

[ВЫВОД 10](#_Toc180325130)

# **ПОСТАНОВКА ЗАДАЧИ**

Создать интерфейс Описание железнодорожного переезда, описывающий оборудование переезда. Создать класс, описывающий простой не регулируемый переезд. Создать декоратор, добавляющий в описание светофор на переезде и декоратор, добавляющий в описание автоматически срабатывающие препятствия. Продемонстрировать описание с различным набором декораторов.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В этой лабораторной работе реализован аналогичный подход к предыдущей задаче, но вместо пиццы декорируется описание железнодорожного переезда. Здесь также используется шаблон "Декоратор" и механизмы CDI.

Основной интерфейс IRailwayCrossingDescription задает метод getDescription, который возвращает описание железнодорожного переезда.

Базовый класс SimpleRailwayCrossing представляет простой нерегулируемый переезд. Этот класс помечен аннотацией @Default, что позволяет CDI использовать его как основную реализацию интерфейса, если не указаны декораторы.

Класс RailwayCrossingDescription реализует бизнес-логику. Он инъектирует реализацию IRailwayCrossingDescription и предоставляет метод getDescription, который возвращает результат декорированного описания.

Декораторы AutoActivatedBarriersRailwayCrossing и TrafficLightRailwayCrossing добавляют к базовому описанию информацию об инфраструктуре:

* AutoActivatedBarriersRailwayCrossing добавляет фразу " + Автоматически срабатывающие препятствия".
* TrafficLightRailwayCrossing добавляет фразу " + Светофор".

Эти классы помечены аннотацией @Decorator и используют аннотацию @Delegate для декорирования базового объекта.

Конфигурация декораторов задается в файле beans.xml, где указано, что сначала применяется AutoActivatedBarriersRailwayCrossing, а затем TrafficLightRailwayCrossing. Порядок декораторов определяет последовательность добавления элементов инфраструктуры.

Класс Laboratory2 является точкой входа программы. В CDI-контейнере создается объект RailwayCrossingDescription, и вызов метода getDescription возвращает итоговое описание с примененными декораторами. Итог выводится на консоль.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

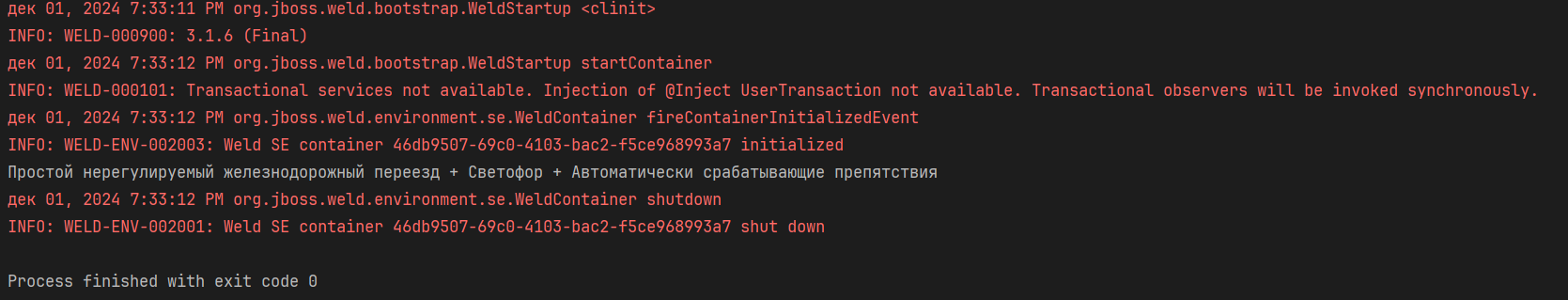


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

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

package Lab2\_Decorator.Component;  
  
import Lab2\_Decorator.Abstraction.IRailwayCrossingDescription;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.enterprise.inject.Default;  
  
@Default  
@AllArgsConstructor  
public class SimpleRailwayCrossing implements IRailwayCrossingDescription {  
  
 @Override  
 public String getDescription() {  
 return "Простой нерегулируемый железнодорожный переезд";  
 }  
}

package Lab2\_Decorator.Decorator;  
  
import Lab2\_Decorator.Abstraction.IRailwayCrossingDescription;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
  
@Decorator  
@NoArgsConstructor  
@AllArgsConstructor  
public class AutoActivatedBarriersRailwayCrossing implements IRailwayCrossingDescription {  
  
 private String additionalInfrastructure = " + Автоматически срабатывающие препятствия";  
  
 @Inject  
 @Delegate  
 private IRailwayCrossingDescription railwayCrossingDescription;  
  
  
 @Override  
 public String getDescription() {  
 String baseInfrastructure = railwayCrossingDescription.getDescription();  
  
 return baseInfrastructure + additionalInfrastructure;  
 }  
}

package Lab2\_Decorator.Decorator;  
  
import Lab2\_Decorator.Abstraction.IRailwayCrossingDescription;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
  
@Decorator  
@NoArgsConstructor  
@AllArgsConstructor  
public class TrafficLightRailwayCrossing implements IRailwayCrossingDescription {  
  
 private String additionalInfrastructure = " + Светофор";  
  
 @Inject  
 @Delegate  
 private IRailwayCrossingDescription railwayCrossingDescription;  
  
  
 @Override  
 public String getDescription() {  
 String baseInfrastructure = railwayCrossingDescription.getDescription();  
  
 return baseInfrastructure + additionalInfrastructure;  
 }  
}

package Lab2\_Decorator;  
  
import Lab2\_Decorator.Component.RailwayCrossingDescription;  
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 {  
  
 RailwayCrossingDescription railwayCrossingDescription = container.select(RailwayCrossingDescription.class).get();  
  
 System.out.println(railwayCrossingDescription.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.AutoActivatedBarriersRailwayCrossing</class>  
 <class>Lab2\_Decorator.Decorator.TrafficLightRailwayCrossing</class>  
 </decorators>  
</beans>

Lab2\_Decorator.Component;  
  
  
import Lab2\_Decorator.Abstraction.IRailwayCrossingDescription;  
  
import javax.inject.Inject;  
  
public class RailwayCrossingDescription {  
  
 @Inject  
 private IRailwayCrossingDescription railwayCrossingDescription;  
  
 public String getDescription() {  
 return railwayCrossingDescription.getDescription();  
 }  
  
}

# **ВЫВОД**

Работа демонстрирует использование шаблона "Декоратор" для динамического добавления инфраструктурных элементов к описанию железнодорожного переезда. Использование CDI и декораторов упрощает модификацию и расширение функционала без изменения исходного кода. Это улучшает читаемость, модульность и повторное использование компонентов. Такой подход особенно удобен для добавления новых функций в будущем.