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 №12

Выполнил: Шедания В. М.

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 7](#_Toc180325129)

[ВЫВОД 11](#_Toc180325130)

# **ПОСТАНОВКА ЗАДАЧИ**

Создать интерфейс Преобразование одномерного массива. Создать класс, выполняющий удаление заданных элементов из массива. Создать декоратор, выполняющий замену заданных элементов массива, на заданный элемент и декоратор, выполняющий медианную фильтрацию массива. Продемонстрировать операции с различным набором декораторов.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В методе main класса Laboratory2 происходит инициализация контейнера CDI (Contexts and Dependency Injection) с помощью библиотеки Weld. Этот контейнер управляет зависимостями между объектами и их жизненным циклом. В результате создается объект WeldContainer, который будет использоваться для получения экземпляров других классов, отмеченных аннотациями CDI.

Создается одномерный массив целых чисел: int[] array = {1, 2, 100, 3, 4, 5, 6, 7, 200, 9};. Этот массив содержит десять элементов, некоторые из которых будут удалены или изменены в процессе трансформации. Массив служит исходной последовательностью данных, над которой будут выполняться преобразования.

Далее с помощью контейнера CDI выбирается экземпляр класса TransformerBean, который инкапсулирует логику преобразования массива. Этот класс содержит метод transform, который отвечает за вызов метода преобразования на объекте, реализующем интерфейс ArrayTransformer. Это позволяет использовать разные реализации преобразования в зависимости от текущих требований.

При вызове метода transformerBean.transform(array) вызывается метод transform на объекте TransformerBean, который, в свою очередь, вызывает метод transform на объекте, реализующем интерфейс ArrayTransformer. В данном случае это будет класс RemoveElementsTransformer, который отвечает за удаление заданных элементов из массива.

В классе RemoveElementsTransformer проверяется, содержится ли элемент массива в списке для удаления, который определён как elementsToRemove = {1, 2}. Если элемент массива совпадает с одним из этих значений, он не включается в результирующий массив. В результате формируется новый массив, в котором отсутствуют указанные элементы.

После выполнения трансформации выводится исходный и результат преобразования. Затем, в зависимости от конфигурации, могут быть добавлены декораторы, такие как ReplaceElementDecorator и MedianFilterDecorator. Эти декораторы могут быть применены для дальнейшей обработки массива, что позволяет гибко изменять поведение системы, добавляя новые функции без изменения существующего кода.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

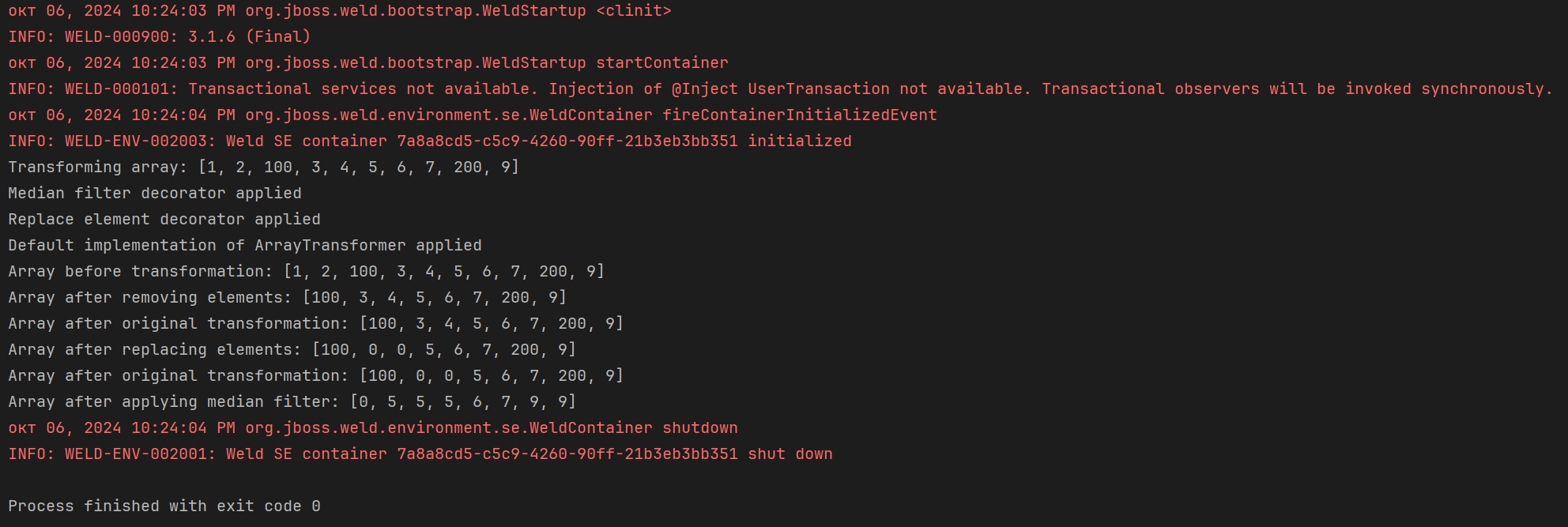


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab2\_Decorator.Decorator;  
  
import Lab2\_Decorator.Trasnformer.ArrayTransformer;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
import java.util.Arrays;  
  
// Decorator for applying a median filter to an array transformer  
@Decorator  
@AllArgsConstructor  
@NoArgsConstructor  
public class MedianFilterDecorator implements ArrayTransformer {  
  
 @Inject  
 @Delegate  
 private ArrayTransformer arrayTransformer;  
  
 @Override  
 public int[] transform(int[] array) {  
 System.out.println("Median filter decorator applied");  
  
 int[] transformedArray = arrayTransformer.transform(array);  
 System.out.println("Array after original transformation: " + Arrays.toString(transformedArray));  
  
 int[] filteredArray = applyMedianFilter(transformedArray);  
 System.out.println("Array after applying median filter: " + Arrays.toString(filteredArray));  
  
 return filteredArray;  
 }  
  
 private int[] applyMedianFilter(int[] array) {  
 int[] result = new int[array.length];  
 for (int i = 0; i < array.length; i++) {  
 result[i] = medianOfNeighbors(array, i);  
 }  
 return result;  
 }  
  
 private int medianOfNeighbors(int[] array, int index) {  
 int start = Math.max(0, index - 2);  
 int end = Math.min(array.length - 1, index + 2);  
 int[] neighbors = Arrays.copyOfRange(array, start, end + 1);  
 Arrays.sort(neighbors);  
 return neighbors[neighbors.length / 2];  
 }  
}

package Lab2\_Decorator.Decorator;  
  
import Lab2\_Decorator.Trasnformer.ArrayTransformer;  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
import java.util.Arrays;  
  
// Decorator for replacing specified elements in an array transformer  
@Decorator  
@AllArgsConstructor  
@NoArgsConstructor  
public class ReplaceElementDecorator implements ArrayTransformer {  
  
 @Inject  
 @Delegate  
 private ArrayTransformer arrayTransformer;  
  
 private int[] elementsToReplace = {3, 4};  
 private int newElement = 0;  
  
 @Override  
 public int[] transform(int[] array) {  
 System.out.println("Replace element decorator applied");  
  
 int[] transformedArray = arrayTransformer.transform(array);  
 System.out.println("Array after original transformation: " + Arrays.toString(transformedArray));  
  
 // Replace specified elements in the transformed array  
 for (int i = 0; i < transformedArray.length; i++) {  
 if (shouldReplace(transformedArray[i])) {  
 transformedArray[i] = newElement; // Replace element  
 }  
 }  
  
 System.out.println("Array after replacing elements: " + Arrays.toString(transformedArray));  
 return transformedArray;  
 }  
  
 private boolean shouldReplace(int element) {  
 for (int e : elementsToReplace) {  
 if (e == element) {  
 return true;  
 }  
 }  
 return false;  
 }  
}

package Lab2\_Decorator.Trasnformer;  
  
public interface ArrayTransformer {  
 int[] transform(int[] array);  
}

package Lab2\_Decorator.Trasnformer;  
  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.enterprise.inject.Default;  
import java.util.Arrays;  
  
// Transformer for removing specified elements from an array  
@Default  
@AllArgsConstructor  
@NoArgsConstructor  
public class RemoveElementsTransformer implements ArrayTransformer {  
  
 private int[] elementsToRemove = {1, 2};  
  
 private boolean contains(int value) {  
 for (int element : elementsToRemove) {  
 if (element == value) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public int[] transform(int[] array) {  
 System.out.println("Default implementation of ArrayTransformer applied");  
 System.out.println("Array before transformation: " + Arrays.toString(array));  
  
 int[] tempArray = new int[array.length];  
 int count = 0;  
  
 for (int value : array) {  
 if (!contains(value)) {  
 tempArray[count++] = value;  
 }  
 }  
  
 int[] resultArray = Arrays.copyOf(tempArray, count);  
 System.out.println("Array after removing elements: " + Arrays.toString(resultArray));  
 return resultArray;  
 }  
}

package Lab2\_Decorator.Trasnformer;  
  
import javax.inject.Inject;  
import java.util.Arrays;  
  
public class TransformerBean {  
  
 @Inject  
 private ArrayTransformer arrayTransformer;  
  
 public int[] transform(int[] array) {  
 System.out.println("Transforming array: " + Arrays.toString(array));  
 return arrayTransformer.transform(array);  
 }  
}

package Lab2\_Decorator;  
  
import Lab2\_Decorator.Trasnformer.TransformerBean;  
import org.jboss.weld.environment.se.Weld;  
import org.jboss.weld.environment.se.WeldContainer;  
  
import java.util.Arrays;  
  
public class Laboratory2 {  
 protected static Weld weld;  
 protected static WeldContainer container;  
  
 public static void main(String[] args) {  
 weld = new Weld();  
 container = weld.initialize();  
  
 try {  
 int[] array = {1, 2, 100, 3, 4, 5, 6, 7, 200, 9};  
  
 TransformerBean transformerBean = container.select(TransformerBean.class).get();  
  
 int[] modifiedArray = transformerBean.transform(array);  
  
 } finally {  
 weld.shutdown();  
 }  
 }  
  
}

# **ВЫВОД**

В ходе выполнения лабораторной работы №2 была реализована система для преобразования одномерного массива с использованием декораторов в рамках CDI-контейнера. Основной задачей было создание интерфейса и классов для удаления заданных элементов массива, а также применение декораторов для замены элементов и медианной фильтрации.

Программа успешно выполняла удаление элементов из массива, замену определённых элементов и применение медианного фильтра, что позволило продемонстрировать гибкость использования декораторов для динамического изменения функциональности. Лабораторная работа показала, как можно расширять возможности программы, добавляя новые функции без изменения базового кода, что соответствует принципам модульности и инверсии зависимостей.

Таким образом, работа помогла закрепить навыки работы с декораторами и контейнерами CDI, а также продемонстрировала возможности гибкого изменения поведения программы.