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 №13

Выполнил: Веселков М.Ю.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc183377068)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc183377069)

[ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ 6](#_Toc183377070)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 7](#_Toc183377071)

[КОД ПРОГРАММЫ 8](#_Toc183377072)

[ВЫВОД 11](#_Toc183377073)

# **ПОСТАНОВКА ЗАДАЧИ**

Создать систему взаимодействующих классов: класс, хранящий текст и информацию о длине строки и количестве строк на странице; класс, рассчитывающий количество слов в тексте; класс, рассчитывающий количество строк в тексте; класс, рассчитывающий количество страниц. Классы расчёта, должны срабатывать при каждом изменении текста.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В ходе выполнения лабораторной работы была разработана система для управления текстом, в которой реализованы механизмы изменения текста с использованием событий. В начале работы были созданы две аннотации: @Added и @Removed, предназначенные для квалификации событий добавления и удаления текста. Эти аннотации позволяют отслеживать изменения текста и взаимодействовать с другими компонентами системы.

Затем был реализован класс TextManager, который содержит текст и информацию о его структуре, включая длину строки и количество строк на странице. В этом классе происходят инъекции событий textAddedEvent и textRemovedEvent, которые вызываются при изменении текста. При добавлении нового текста метод addText вызывает событие textAddedEvent, передавая текущее состояние текста, что позволяет другим компонентам реагировать на это изменение. Аналогично, метод removeText вызывает событие textRemovedEvent при удалении части текста. В обоих случаях выводится состояние текста для демонстрации изменений.

Далее был создан класс WordCountCalculator, который отслеживает изменения текста и подсчитывает количество слов. Этот класс использует метод calculateWordCount для анализа текста, определяя количество слов. Реакция на события добавления и удаления текста обеспечивается методами countWordsOnAdd и countWordsOnRemove, которые обрабатывают соответствующие события, инициированные классом TextManager.

Также была реализована функциональность подсчета количества строк в тексте через класс LineCountCalculator. При изменении текста этот класс обновляет количество строк, используя метод calculateLineCount. Методы countLinesOnAdd и countLinesOnRemove отслеживают изменения в тексте и выводят текущее количество строк после каждой операции.

Кроме того, был создан класс PageCountCalculator, который рассчитывает количество страниц на основе длины текста и заданного максимального количества строк на одной странице. При изменении текста метод calculatePageCount пересчитывает количество страниц, а результаты выводятся для отображения обновлений.

Наконец, в классе Laboratory3 была организована основная логика работы программы. Используя контейнер CDI (Contexts and Dependency Injection), инициализируется экземпляр TextManager, через который происходит изменение текста. Программа демонстрирует добавление нового текста, обновление существующего и удаление части текста, чтобы продемонстрировать взаимодействие компонентов и обработку событий. В завершение работы контейнер CDI корректно завершает свою работу, освобождая все ресурсы.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – 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 TextChanged {  
}

package Lab3\_Event.Util;  
  
public class LineCountCalculator {  
 public int calculateLineCount(String text) {  
 if (text == null || text.isEmpty()) {  
 return 0;  
 }  
 return text.split("\n").length; // Подсчет строк  
 }  
}

package Lab3\_Event.Util;  
  
public class PageCountCalculator {  
 private final TextContent textContent;  
  
 public PageCountCalculator(TextContent textContent) {  
 this.textContent = textContent;  
 }  
  
 public int calculatePageCount() {  
 if (textContent.getContent() == null || textContent.getContent().isEmpty()) {  
 return 0;  
 }  
 int lineCount = textContent.getContent().split("\n").length;  
 int maxLinesPerPage = textContent.getMaxLinesPerPage();  
 return (int) Math.*ceil*((double) lineCount / maxLinesPerPage);  
 }  
}

package Lab3\_Event.Util;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
  
@Getter  
@AllArgsConstructor  
@NoArgsConstructor  
public class TextContent {  
 private String content; // Текст  
 private int maxLinesPerPage; // Максимальное количество строк на странице  
  
 @Override  
 public String toString() {  
 return "Текст: " + content + ", Максимальное количество строк на странице: " + maxLinesPerPage;  
 }  
}

package Lab3\_Event.Util;  
  
import Lab3\_Event.Annotation.TextChanged;  
import javax.enterprise.event.Event;  
import javax.inject.Inject;  
import java.util.ArrayList;  
import java.util.List;  
  
public class TextManager {  
  
 private String text;  
 private final int maxLinesPerPage = 25; // Примерное количество строк на странице  
  
 @Inject  
 @TextChanged  
 private Event<String> textChangedEvent; // Событие изменения текста  
  
 public TextManager(String initialText) {  
 this.text = initialText;  
 textChangedEvent.fire(text); // Инициализация с отправкой события  
 }  
  
 public void setText(String newText) {  
 this.text = newText;  
 textChangedEvent.fire(newText); // Отправка события при изменении текста  
 }  
  
 public String getText() {  
 return text;  
 }  
  
 public int getLength() {  
 return text.length(); // Длина строки  
 }  
  
 public int getLinesCount() {  
 return text.split("\n").length; // Количество строк  
 }  
  
 public int getMaxLinesPerPage() {  
 return maxLinesPerPage;  
 }  
}

package Lab3\_Event.Util;  
  
public class WordCountCalculator {  
 public int calculateWordCount(String text) {  
 if (text == null || text.isEmpty()) {  
 return 0;  
 }  
 return text.split("\\s+").length; // Подсчет слов  
 }  
}

package Lab3\_Event;  
  
import Lab3\_Event.Util.TextContent;  
import Lab3\_Event.Util.WordCountCalculator;  
import Lab3\_Event.Util.LineCountCalculator;  
import Lab3\_Event.Util.PageCountCalculator;  
  
public class Laboratory3 {  
  
 public static void main(String[] args) {  
 // Создаем объект TextContent  
 TextContent textContent = new TextContent("Это пример текста.\nЭтот текст состоит из нескольких строк.", 25);  
  
 // Вывод информации о тексте  
 System.*out*.println("Исходный текст:");  
 System.*out*.println(textContent.getContent());  
 System.*out*.println("Максимальное количество строк на странице: " + textContent.getMaxLinesPerPage());  
  
 // Создаем объекты калькуляторов  
 WordCountCalculator wordCountCalculator = new WordCountCalculator();  
 LineCountCalculator lineCountCalculator = new LineCountCalculator();  
 PageCountCalculator pageCountCalculator = new PageCountCalculator(textContent);  
  
 // Подсчитываем и выводим количество слов  
 int wordCount = wordCountCalculator.calculateWordCount(textContent.getContent());  
 System.*out*.println("\nКоличество слов: " + wordCount);  
  
 // Подсчитываем и выводим количество строк  
 int lineCount = lineCountCalculator.calculateLineCount(textContent.getContent());  
 System.*out*.println("Количество строк: " + lineCount);  
  
 // Подсчитываем и выводим количество страниц  
 int pageCount = pageCountCalculator.calculatePageCount();  
 System.*out*.println("Количество страниц: " + pageCount);  
  
 // Изменяем текст через сеттеры (если потребуется их добавить)  
 textContent = new TextContent("Новый текст для демонстрации работы.\nЭтот текст содержит несколько строк.", 30);  
  
 // Вывод обновленной информации  
 System.*out*.println("\nОбновленный текст:");  
 System.*out*.println(textContent.getContent());  
 System.*out*.println("Максимальное количество строк на странице: " + textContent.getMaxLinesPerPage());  
  
 // Пересчитываем метрики для обновленного текста  
 wordCount = wordCountCalculator.calculateWordCount(textContent.getContent());  
 System.*out*.println("\nКоличество слов: " + wordCount);  
  
 lineCount = lineCountCalculator.calculateLineCount(textContent.getContent());  
 System.*out*.println("Количество строк: " + lineCount);  
  
 pageCount = pageCountCalculator.calculatePageCount();  
 System.*out*.println("Количество страниц: " + pageCount);  
 }  
}

# **ВЫВОД**

В ходе выполнения лабораторной работы №3 была разработана система для управления текстовым содержимым, которая реагирует на изменения текста с использованием событий. Для реализации взаимодействия компонентов были использованы CDI-аннотации и события, что обеспечило реакцию системы на каждое добавление или удаление текста.

Система включала три ключевых компонента:

TextManager — класс, управляющий текстом и генерирующий события при изменении текста.

WordCountCalculator — компонент, рассчитывающий количество слов в тексте при каждом его изменении.

LineCountCalculator — класс, определяющий количество строк в тексте.

PageCountCalculator — компонент, вычисляющий количество страниц в тексте с учётом заданного ограничения на количество строк на странице.

Была реализована обработка событий, таких как добавление и удаление текста, и продемонстрировано взаимодействие между компонентами через публикацию и обработку событий. Лабораторная работа позволила понять, как с помощью событий и CDI можно реализовать гибкую и модульную систему, реагирующую на изменения данных.