МИНИСТЕРСТВО ТРАНСПОРТА РФ

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

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №4  
по дисциплине «Корпоративные информационные системы»

Тема: «Валидация с использованием стандартных аннотаций»

Вариант №7

Выполнил: Шедания В. М.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180325218)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180325219)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 5](#_Toc180325220)

[КОД ПРОГРАММЫ 6](#_Toc180325221)

[ВЫВОД 9](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс Поезд. С помощью стандартного API валидации с использованием стандартных аннотаций произвести валидацию класса Поезд. Валидацию провести с использованием аннотаций при описании класса и с указанием ограничений в XML. Обязательно использовать аннотацию @Pattern с проверкой, определяемой регулярным выражением.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В начале лабораторной работы настраивается среда для валидации данных, в которой используется Jakarta Validation с провайдером Hibernate Validator. Для этого создается фабрика валидаторов ValidatorFactory, настраивается интерполятор сообщений, и затем создается валидатор Validator для проверки объектов.

Далее в проекте реализуются две модели данных — XmlValidatedTrain и CodeValidatedTrain. В классе CodeValidatedTrain используется аннотационная валидация полей с помощью Jakarta Validation, чтобы задать ограничения на значения каждого поля: например, проверка на пустое значение, соответствие регулярным выражениям для имени и номера поезда, а также минимальное и максимальное значение для количества вагонов. В XmlValidatedTrain валидация не задается аннотациями — вместо этого ограничения прописаны в XML-файле constraints.xml.

В файле конфигурации валидации validation-config.xml подключается XML-мэппинг ограничений constraints.xml. В этом XML-файле указаны правила валидации для полей класса XmlValidatedTrain: ограничения на непустые значения, соответствие формату, а также минимальные и максимальные значения. Это позволяет задать все ограничения без использования аннотаций в коде, только через XML.

В основном классе Laboratory4 создается экземпляр XmlValidatedTrain с корректными данными и вызывается метод validateAndPrint, который проверяет объект на соответствие установленным правилам. Поскольку данные валидны, вывод подтверждает успешное прохождение валидации.

Затем создается объект CodeValidatedTrain с некорректными данными, где нарушены правила валидации, такие как формат имени, номера поезда, наличие маршрута и количество вагонов. После запуска метода validateAndPrint выводится информация о всех нарушениях валидации с соответствующими сообщениями об ошибках для каждого поля.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

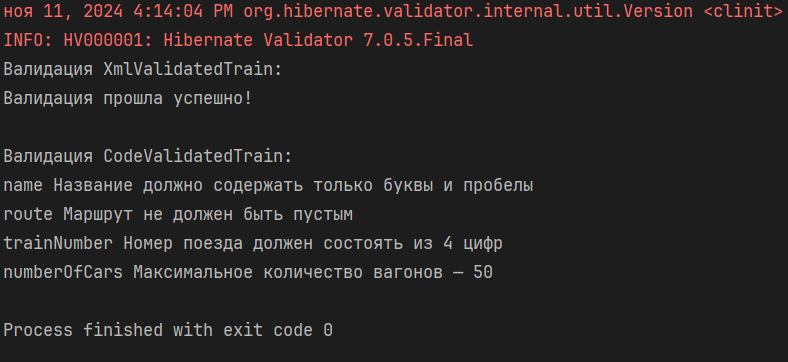


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab4\_Validation.Model;  
  
import jakarta.validation.constraints.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class CodeValidatedTrain {  
  
 @NotNull(message = "Название не должно быть пустым")  
 @Pattern(regexp = "^[A-Za-zА-Яа-яЁё\\s]+$", message = "Название должно содержать только буквы и пробелы")  
 private String name;  
  
 @NotNull(message = "Номер поезда не должен быть пустым")  
 @Pattern(regexp = "^\\d{4}$", message = "Номер поезда должен состоять из 4 цифр")  
 private String trainNumber;  
  
 @NotNull(message = "Маршрут не должен быть пустым")  
 private String route;  
  
 @Min(value = 1, message = "Минимальное количество вагонов должно быть 1")  
 @Max(value = 50, message = "Максимальное количество вагонов — 50")  
 private int numberOfCars;  
}

package Lab4\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class XmlValidatedTrain {  
  
 private String name;  
 private String trainNumber;  
 private String route;  
 private int numberOfCars;  
}

package Lab4\_Validation;  
  
import Lab4\_Validation.Model.XmlValidatedTrain;  
import Lab4\_Validation.Model.CodeValidatedTrain;  
import jakarta.validation.ConstraintViolation;  
import jakarta.validation.Validation;  
import jakarta.validation.Validator;  
import jakarta.validation.ValidatorFactory;  
import org.hibernate.validator.HibernateValidator;  
import org.hibernate.validator.messageinterpolation.ParameterMessageInterpolator;  
  
import java.util.Set;  
  
public class Laboratory4 {  
  
 public static void main(String[] args) {  
 ValidatorFactory factory = Validation.byProvider(HibernateValidator.class)  
 .configure()  
 .messageInterpolator(new ParameterMessageInterpolator())  
 .buildValidatorFactory();  
 Validator validator = factory.getValidator();  
  
 XmlValidatedTrain xmlTrain = new XmlValidatedTrain();  
 xmlTrain.setName("Ласточка");  
 xmlTrain.setTrainNumber("5678");  
 xmlTrain.setRoute("Сочи - Адлер");  
 xmlTrain.setNumberOfCars(5);  
  
 System.out.println("Валидация XmlValidatedTrain:");  
 validateAndPrint(validator, xmlTrain);  
  
 CodeValidatedTrain codeTrain = new CodeValidatedTrain();  
 codeTrain.setName("1234"); // Некорректное имя (должно содержать только буквы и пробелы)  
 codeTrain.setTrainNumber("12A4"); // Некорректный номер (должен состоять только из 4 цифр)  
 codeTrain.setRoute(null); // Поле не должно быть пустым  
 codeTrain.setNumberOfCars(100); // Некорректное значение (должно быть от 1 до 50)  
  
 System.out.println("\nВалидация CodeValidatedTrain:");  
 validateAndPrint(validator, codeTrain);  
 }  
  
 private static <T> void validateAndPrint(Validator validator, T object) {  
 Set<ConstraintViolation<T>> violations = validator.validate(object);  
 if (violations.isEmpty()) {  
 System.out.println("Валидация прошла успешно!");  
 } else {  
 for (ConstraintViolation<T> violation : violations) {  
 System.out.println(violation.getPropertyPath() + " " + violation.getMessage());  
 }  
 }  
 }  
}

<constraint-mappings xmlns="http://jboss.org/xml/ns/javax/validation/mapping"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/mapping validation-mapping-1.1.xsd"  
 version="1.1">  
 <bean class="Lab4\_Validation.Model.XmlValidatedTrain" ignore-annotations="false">  
 <field name="name">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Название не должно быть пустым</message>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Pattern">  
 <message>Название должно содержать только буквы и пробелы</message>  
 <element name="regexp">^[A-Za-zА-Яа-яЁё\\s]+$</element>  
 </constraint>  
 </field>  
 <field name="trainNumber">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Номер поезда не должен быть пустым</message>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Pattern">  
 <message>Номер поезда должен состоять из 4 цифр</message>  
 <element name="regexp">^\d{4}$</element>  
 </constraint>  
 </field>  
 <field name="route">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Маршрут не должен быть пустым</message>  
 </constraint>  
 </field>  
 <field name="numberOfCars">  
 <constraint annotation="jakarta.validation.constraints.Min">  
 <message>Минимальное количество вагонов должно быть 1</message>  
 <element name="value">1</element>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Max">  
 <message>Максимальное количество вагонов — 50</message>  
 <element name="value">50</element>  
 </constraint>  
 </field>  
 </bean>  
</constraint-mappings>

<validation-config xmlns="http://jboss.org/xml/ns/javax/validation/configuration"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/configuration validation-configuration-1.1.xsd"  
 version="1.1">  
 <constraint-mapping>META-INF/constraints.xml</constraint-mapping>  
</validation-config>

# **ВЫВОД**

В ходе выполнения лабораторной работы были изучены принципы валидации данных в Java с использованием Jakarta Validation и Hibernate Validator. Были реализованы два подхода к валидации: аннотационная валидация в классе CodeValidatedTrain и XML-настройки валидации для класса XmlValidatedTrain. Это позволило продемонстрировать гибкость Jakarta Validation и возможность отделения логики валидации от кода с помощью XML-файлов.

Результаты экспериментов показали, что валидация корректно обрабатывает различные типы ограничений: проверку на пустые значения, соответствие регулярным выражениям и диапазоны числовых значений. В случае некорректных данных валидатор выводит сообщения об ошибках, что облегчает диагностику и исправление введенных данных.

Работа подтвердила эффективность Jakarta Validation для обеспечения корректности данных на уровне моделей. Использование аннотаций позволяет сократить объем кода и сосредоточить правила валидации непосредственно рядом с полями. В то же время, конфигурация в XML дает возможность гибко управлять валидацией без изменения исходного кода, что может быть полезно при изменениях требований.