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 №15

Выполнил: Дронов А. А.

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 4](#_Toc180325220)

[КОД ПРОГРАММЫ 7](#_Toc180325221)

[ВЫВОД 10](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс Университет. С помощью стандартного API валидации с использованием стандартных аннотаций произвести валидацию класса Университет. 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 этой лабораторной работе реализована валидация объектов, представляющих университеты, с использованием Jakarta Validation (Bean Validation) и Hibernate Validator. Объекты CodeValidatedUniversity и XmlValidatedUniversity проходят проверку на соответствие бизнес-правилам через программные и декларативные ограничения.

Класс CodeValidatedUniversity использует аннотации для указания правил валидации. Каждое поле имеет ограничения: название должно содержать только буквы и пробелы, адрес не может быть пустым, год основания должен быть в диапазоне от 1800 до 2024, а тип университета — одним из заранее определенных значений. Такой подход интегрирует правила валидации непосредственно в код, обеспечивая читаемость и простоту сопровождения.

Класс XmlValidatedUniversity полагается на декларативное описание ограничений в XML-файле constraints.xml. Этот файл содержит аналогичные ограничения для полей: проверку на null, соответствие регулярным выражениям и диапазону значений. Такой способ позволяет изменять правила валидации без необходимости вносить изменения в код программы.

Основной класс Laboratory4 создает экземпляр валидатора через ValidatorFactory, используя Hibernate Validator. Затем он выполняет валидацию объектов XmlValidatedUniversity и CodeValidatedUniversity. Объект XmlValidatedUniversity демонстрирует пример некорректного значения года основания, нарушающего ограничения. Для объекта CodeValidatedUniversity задаются некорректные значения сразу для нескольких полей, что приводит к выявлению множества нарушений валидации.

Метод validateAndPrint проверяет объекты и выводит на консоль список нарушений, если таковые имеются. Каждое нарушение сопровождается сообщением, указывающим поле и причину ошибки. Если объект соответствует всем ограничениям, выводится сообщение о том, что валидация прошла успешно.

Декларативный подход к валидации через XML-конфигурацию указан в файле validation-config. Он связывает валидатор с описанными ограничениями, что позволяет отделить бизнес-логику валидации от реализации объектов. Это повышает гибкость и расширяемость системы.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

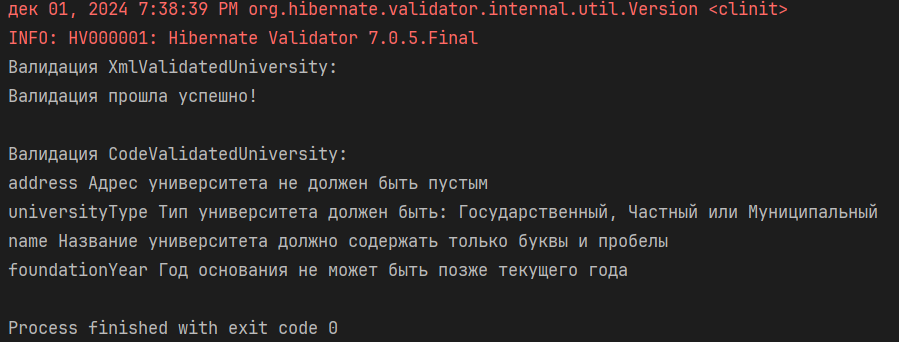


Рисунок 1 – Результат работы программы

# **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 CodeValidatedUniversity {  
  
 @NotNull(message = "Название университета не должно быть пустым")  
 @Pattern(regexp = "^[A-Za-zА-Яа-яЁё\\s]+$", message = "Название университета должно содержать только буквы и пробелы")  
 private String name;  
  
 @NotNull(message = "Адрес университета не должен быть пустым")  
 private String address;  
  
 @Min(value = 1800, message = "Год основания не может быть раньше 1800")  
 @Max(value = 2024, message = "Год основания не может быть позже текущего года")  
 private int foundationYear;  
  
 @NotNull(message = "Тип университета не должен быть пустым")  
 @Pattern(regexp = "^(Государственный|Частный|Муниципальный)$", message = "Тип университета должен быть: Государственный, Частный или Муниципальный")  
 private String universityType;  
}

package Lab4\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class XmlValidatedUniversity {  
  
 private String name;  
 private String address;  
 private int foundationYear;  
 private String universityType;  
}

package Lab4\_Validation;  
  
import Lab4\_Validation.Model.XmlValidatedUniversity;  
import Lab4\_Validation.Model.CodeValidatedUniversity;  
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();  
  
 XmlValidatedUniversity xmlUniversity = new XmlValidatedUniversity();  
 xmlUniversity.setName("МГУ имени Ломоносова");  
 xmlUniversity.setAddress("г. Москва, Ленинские Горы, 1");  
 xmlUniversity.setFoundationYear(1800); // Ошибка: меньше 1800  
 xmlUniversity.setUniversityType("Государственный");  
  
 System.out.println("Валидация XmlValidatedUniversity:");  
 validateAndPrint(validator, xmlUniversity);  
  
 CodeValidatedUniversity codeUniversity = new CodeValidatedUniversity();  
 codeUniversity.setName("123"); // Некорректное имя  
 codeUniversity.setAddress(null); // Поле не должно быть пустым  
 codeUniversity.setFoundationYear(2100); // Некорректное значение  
 codeUniversity.setUniversityType("Некоммерческий"); // Некорректный тип  
  
 System.out.println("\nВалидация CodeValidatedUniversity:");  
 validateAndPrint(validator, codeUniversity);  
 }  
  
 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());  
 }  
 }  
 }  
}

<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>

<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.XmlValidatedUniversity" ignore-annotations="false">  
 <field name="name">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Название университета не должно быть пустым</message>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Pattern">  
 <message>Название университета должно содержать только буквы и пробелы</message>  
 <element name="regexp">^[A-Za-zА-Яа-яЁё\s]+$</element>  
 </constraint>  
 </field>  
 <field name="address">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Адрес университета не должен быть пустым</message>  
 </constraint>  
 </field>  
 <field name="foundationYear">  
 <constraint annotation="jakarta.validation.constraints.Min">  
 <message>Год основания не может быть раньше 1800</message>  
 <element name="value">1800</element>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Max">  
 <message>Год основания не может быть позже текущего года</message>  
 <element name="value">2024</element>  
 </constraint>  
 </field>  
 <field name="universityType">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Тип университета не должен быть пустым</message>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Pattern">  
 <message>Тип университета должен быть: Государственный, Частный или Муниципальный</message>  
 <element name="regexp">^(Государственный|Частный|Муниципальный)$</element>  
 </constraint>  
 </field>  
 </bean>  
</constraint-mappings>

# **ВЫВОД**

Работа демонстрирует два подхода к валидации данных: программный (аннотации) и декларативный (XML). Оба подхода обеспечивают высокую гибкость и читаемость кода. Декларативный способ удобен для вынесения правил валидации из кода, что позволяет изменить их без перекомпиляции программы. Программный подход, в свою очередь, обеспечивает тесную связь ограничений с логикой объекта. Такой подход к валидации гарантирует, что данные, используемые в приложении, соответствуют установленным бизнес-правилам, минимизируя вероятность ошибок.