МИНИСТЕРСТВО ТРАНСПОРТА РФ

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

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №5  
по дисциплине «Корпоративные информационные системы»

Тема: «Создание аннотации для валидации»

Вариант №13

Выполнил: Веселков М.Ю.

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 6](#_Toc180325220)

[КОД ПРОГРАММЫ 7](#_Toc180325221)

[ВЫВОД 10](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс Магазин. Поле КоличествоЕдиницПродуктаПоставленого содержит количество единиц определённого продукта, доставленного в магазин на реализацию, а поле КоличествоЕдиницПродуктаПродано содержит количество проданных единиц определённого продукта. Описать ограничение, проверяющие что КоличествоЕдиницПродуктаПродано не превышает КоличествоЕдиницПродуктаПоставленого.  Продемонстрировать работу с ограничением.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В ходе выполнения лабораторной работы была разработана система проверки соответствия количества проданных единиц продукта количеству поставленных с использованием Java и Jakarta Bean Validation. Работа началась с создания аннотации @ValidProductQuantity, которая служит для валидации объектов класса Shop. Аннотация связана с валидатором ProductQuantityValidator, реализующим логику проверки.

Затем был создан класс Shop, описывающий объекты магазина с полями deliveredProductUnits (количество поставленных единиц продукта) и soldProductUnits (количество проданных единиц продукта). Для данного класса были добавлены базовые аннотации для проверки корректности полей и применена созданная аннотация @ValidProductQuantity на уровне класса для проверки бизнес-логики.

Ключевая часть работы — реализация валидатора ProductQuantityValidator. Этот класс проверяет, что количество проданных единиц продукта не превышает количество поставленных. Логика проверки учитывает стандартные методы работы с числами в Java, а для демонстрации работы использован механизм валидации Jakarta Bean Validation.

Для демонстрации работы валидатора был создан основной класс Laboratory5. В нем сгенерированы тестовые данные: валидный объект магазина, где количество проданных единиц продукта не превышает поставленные, а также невалидный объект, где это условие нарушено. С помощью фабрики валидаторов был настроен объект Validator, который выполняет проверку магазинов на соответствие требованиям.

Запуск программы подтвердил корректность работы валидатора: для валидного объекта магазина не выводится никаких сообщений об ошибках, а для невалидного отображается сообщение, заданное в аннотации. Таким образом, все задачи лабораторной работы выполнены, а результаты соответствуют ожиданиям.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab5\_Annotation\_Validation.Annotation;  
  
import Lab5\_Annotation\_Validation.Validator.ProductQuantityValidator;  
import jakarta.validation.Constraint;  
import jakarta.validation.Payload;  
  
import java.lang.annotation.\*;  
  
@Documented  
@Constraint(validatedBy = ProductQuantityValidator.class)  
@Target({ElementType.*TYPE*})  
@Retention(RetentionPolicy.*RUNTIME*)  
public @interface ValidProductQuantity {  
 String message() default "Количество проданных единиц не может превышать количество поставленных единиц.";  
 Class<?>[] groups() default {};  
 Class<? extends Payload>[] payload() default {};  
}

package Lab5\_Annotation\_Validation.Model;  
  
import Lab5\_Annotation\_Validation.Annotation.ValidProductQuantity;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@ValidProductQuantity  
public class Shop {  
 private int deliveredProductUnits; // Number of delivered product units  
 private int soldProductUnits; // Number of sold product units  
}

package Lab5\_Annotation\_Validation.Validator;  
  
import Lab5\_Annotation\_Validation.Model.Shop;  
import Lab5\_Annotation\_Validation.Annotation.ValidProductQuantity;  
import jakarta.validation.ConstraintValidator;  
import jakarta.validation.ConstraintValidatorContext;  
  
public class ProductQuantityValidator implements ConstraintValidator<ValidProductQuantity, Shop> {  
  
 @Override  
 public boolean isValid(Shop shop, ConstraintValidatorContext context) {  
 return shop.getSoldProductUnits() <= shop.getDeliveredProductUnits();  
 }  
}

package Lab5\_Annotation\_Validation;  
  
import Lab5\_Annotation\_Validation.Model.Shop;  
import jakarta.validation.Validation;  
import jakarta.validation.Validator;  
import jakarta.validation.ValidatorFactory;  
import org.hibernate.validator.messageinterpolation.ParameterMessageInterpolator;  
  
public class Laboratory5 {  
 public static void main(String[] args) {  
 Shop validShop = new Shop(100, 50); // 50 sold, 100 delivered  
 Shop invalidShop = new Shop(100, 150); // 150 sold, 100 delivered  
  
 ValidatorFactory factory = Validation.*byDefaultProvider*()  
 .configure()  
 .messageInterpolator(new ParameterMessageInterpolator())  
 .buildValidatorFactory();  
 Validator validator = factory.getValidator();  
  
 System.*out*.println("Valid shop:");  
 validator.validate(validShop).forEach(violation -> System.*out*.println(violation.getMessage()));  
  
 System.*out*.println("\nInvalid shop:");  
 validator.validate(invalidShop).forEach(violation -> System.*out*.println(violation.getMessage()));  
 }  
}

# **ВЫВОД**

В ходе выполнения лабораторной работы была успешно реализована система проверки корректности количества проданных единиц продукта относительно поставленных с использованием аннотаций и валидации в Java. Создана пользовательская аннотация @ValidProductQuantity, валидатор ProductQuantityValidator, а также модель Shop, обеспечивающая корректное представление данных.

Проверка валидатора на различных тестовых данных подтвердила его работоспособность. Для валидного магазина, где количество проданных единиц продукта не превышает количество поставленных, ошибок обнаружено не было, а для невалидного магазина, где проданное количество превышает поставленное, было выведено сообщение: "The number of sold units cannot exceed the number of delivered units." Это свидетельствует о корректной работе валидатора в условиях, предусмотренных лабораторной задачей.

Данная реализация может быть расширена для более сложных сценариев в реальных приложениях, таких как добавление дополнительных критериев проверки количества товара или поддержка учета возвратов и списаний. Работа продемонстрировала практическое применение Jakarta Bean Validation для задач валидации данных и научила эффективно использовать аннотации и валидационные механизмы в Java-приложениях.