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 №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 6](#_Toc180325220)

[КОД ПРОГРАММЫ 7](#_Toc180325221)

[ВЫВОД 10](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс Поезд. Класс поезд содержит поле Расписание и поле Маршрут, каждое из этих полей содержит список объектов, включающих в качестве поля ЕСР стации. Описать ограничение, проверяющие соответствие расписания поезда его маршруту. Продемонстрировать работу с ограничением.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В ходе выполнения лабораторной работы была разработана система проверки соответствия расписания поезда его маршруту с использованием Java и Jakarta Bean Validation. Работа началась с создания аннотации @ValidRoute, которая служит для валидации объектов класса Train. Аннотация связана с валидатором RouteValidator, реализующим логику проверки.

Затем был создан класс Station, описывающий объекты станций с полями name и ESR. Для этого класса были добавлены базовые методы equals и hashCode для корректного сравнения объектов, а также аннотации для проверки обязательности заполнения полей.

После этого был описан класс Train, содержащий два поля: schedule (расписание) и route (маршрут), которые являются списками объектов Station. На уровень класса была добавлена аннотация @ValidRoute, а сами списки аннотированы @Valid для каскадной проверки корректности станций.

Ключевая часть работы — реализация валидатора RouteValidator. Этот класс проверяет, что все станции в расписании присутствуют в маршруте, и что порядок следования станций в расписании не нарушает порядок, заданный в маршруте. Для проверки используются методы Java Collections, такие как containsAll и indexOf.

Для демонстрации работы валидатора был создан основной класс Laboratory5. В нем сгенерированы тестовые данные: валидный поезд с корректным расписанием и маршрутом, а также поезд с ошибочным расписанием, где нарушен порядок станций. С помощью фабрики валидаторов был настроен объект Validator, который выполняет проверку поездов на соответствие требованиям.

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

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

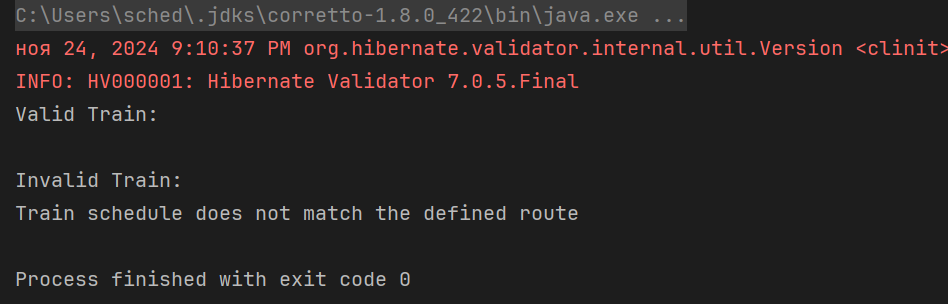


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab5\_Annotation\_Validation.Annotation;  
  
import Lab5\_Annotation\_Validation.Validator.RouteValidator;  
import jakarta.validation.Constraint;  
import jakarta.validation.Payload;  
  
import java.lang.annotation.\*;  
  
@Documented  
@Constraint(validatedBy = RouteValidator.class)  
@Target({ElementType.TYPE})  
@Retention(RetentionPolicy.RUNTIME)  
public @interface ValidRoute {  
 String message() default "Train schedule does not match the defined route";  
 Class<?>[] groups() default {};  
 Class<? extends Payload>[] payload() default {};  
}

package Lab5\_Annotation\_Validation.Model;  
  
import jakarta.validation.constraints.NotNull;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.Objects;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
public class Station {  
 @NotNull(message = "Station name must not be null")  
 private String name;  
  
 @NotNull(message = "Station ESR must not be null")  
 private String ESR; // ЕСР станции  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Station station = (Station) o;  
 return name.equals(station.name) && ESR.equals(station.ESR);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name, ESR);  
 }  
  
}

package Lab5\_Annotation\_Validation.Model;  
  
  
import Lab5\_Annotation\_Validation.Annotation.ValidRoute;  
import jakarta.validation.Valid;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.List;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@ValidRoute  
public class Train {  
 @Valid  
 private List<Station> schedule; // Расписание  
  
 @Valid  
 private List<Station> route; // Маршрут  
}

package Lab5\_Annotation\_Validation.Validator;  
  
import Lab5\_Annotation\_Validation.Annotation.ValidRoute;  
import Lab5\_Annotation\_Validation.Model.Station;  
import Lab5\_Annotation\_Validation.Model.Train;  
import jakarta.validation.ConstraintValidator;  
import jakarta.validation.ConstraintValidatorContext;  
  
import java.util.HashSet;  
import java.util.List;  
  
public class RouteValidator implements ConstraintValidator<ValidRoute, Train> {  
  
 @Override  
 public boolean isValid(Train train, ConstraintValidatorContext context) {  
 if (train.getSchedule() == null || train.getRoute() == null) {  
 return true; // Null-поля проверяются другими аннотациями  
 }  
  
 List<Station> schedule = train.getSchedule();  
 List<Station> route = train.getRoute();  
  
 // Проверяем, что расписание — это подмножество маршрута  
 if (!new HashSet<>(route).containsAll(schedule)) {  
 return false;  
 }  
  
 // Проверяем порядок станций в расписании относительно маршрута  
 int lastIndex = -1;  
 for (Station station : schedule) {  
 int currentIndex = route.indexOf(station);  
 if (currentIndex < 0 || currentIndex < lastIndex) {  
 return false;  
 }  
 lastIndex = currentIndex;  
 }  
 return true;  
 }  
}

package Lab5\_Annotation\_Validation;  
  
import Lab5\_Annotation\_Validation.Model.Station;  
import Lab5\_Annotation\_Validation.Model.Train;  
import jakarta.validation.Validation;  
import jakarta.validation.Validator;  
import jakarta.validation.ValidatorFactory;  
import org.hibernate.validator.messageinterpolation.ParameterMessageInterpolator;  
  
import java.util.Arrays;  
import java.util.List;  
  
public class Laboratory5 {  
 public static void main(String[] args) {  
 Station station1 = new Station("Station A", "ESR1");  
 Station station2 = new Station("Station B", "ESR2");  
 Station station3 = new Station("Station C", "ESR3");  
  
 // Валидный маршрут и расписание  
 List<Station> route = Arrays.asList(station1, station2, station3);  
 List<Station> schedule = Arrays.asList(station1, station2);  
  
 Train validTrain = new Train(schedule, route);  
  
 // Невалидное расписание (не по порядку)  
 List<Station> invalidSchedule = Arrays.asList(station2, station1);  
 Train invalidTrain = new Train(invalidSchedule, route);  
  
 ValidatorFactory factory = Validation.byDefaultProvider()  
 .configure()  
 .messageInterpolator(new ParameterMessageInterpolator())  
 .buildValidatorFactory();  
 Validator validator = factory.getValidator();  
  
 System.out.println("Valid Train:");  
 validator.validate(validTrain).forEach(violation -> System.out.println(violation.getMessage()));  
  
 System.out.println("\nInvalid Train:");  
 validator.validate(invalidTrain).forEach(violation -> System.out.println(violation.getMessage()));  
 }  
}

# **ВЫВОД**

В ходе выполнения лабораторной работы была успешно реализована система проверки корректности расписания поездов относительно их маршрутов с использованием аннотаций и валидации в Java. Создана пользовательская аннотация @ValidRoute, валидатор RouteValidator, а также модели Train и Station, обеспечивающие корректное представление данных.

Проверка валидатора на различных тестовых данных подтвердила его работоспособность. Для валидного поезда, где расписание соответствует маршруту, ошибок обнаружено не было, а для невалидного поезда с ошибочным порядком станций в расписании было выведено сообщение: "Train schedule does not match the defined route". 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 расписания или поддержка более сложных маршрутов. Работа продемонстрировала практическое применение Jakarta Bean Validation для задач валидации данных и научила эффективно использовать аннотации и каскадные проверки в Java-приложениях.