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 №19

Выполнил: Любченкова О.А.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180330916)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180330917)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 6](#_Toc180330918)

[КОД ПРОГРАММЫ 7](#_Toc180330919)

[ВЫВОД 11](#_Toc180330920)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс ЖелезнаяДорога. Поле СписокРайоновУправления содержит объекты типа РайонУправленияЖД. Описать ограничение, проверяющие что все районы управления объединены между собой смежными станциями или перегонами.  Продемонстрировать работу с ограничением.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Код начинается с объявления пользовательской аннотации @ValidRailway, которая используется для валидации объекта Railway. Эта аннотация связывается с валидатором ConnectedRailwayValidator, который реализует логику проверки связности железнодорожной системы. В аннотации задаются сообщение об ошибке, группы и полезная нагрузка для дальнейшей обработки нарушений.

Далее идет описание модели данных. Класс Railway представляет всю железнодорожную сеть и аннотирован пользовательской аннотацией @ValidRailway. Он содержит список районов управления, каждый из которых описан в классе RailwayManagementArea. Эти районы включают станции и перегоны, которые моделируются классами Station и TrackSection соответственно. Вся модель структурирована так, чтобы легко было сопоставить логические связи между объектами.

Основная логика валидации описана в классе ConnectedRailwayValidator. В методе isValid происходит построение графа из станций и перегонов. Если станций или перегонов нет, считается, что объект валиден. При наличии данных проверяется связность графа методом обхода в ширину (BFS). Если все станции достижимы из одной точки, железнодорожная сеть считается связной.

Логика обхода графа представлена в методе isGraphConnected. Она использует очередь и множество для хранения посещенных станций. После завершения обхода сверяется, все ли станции были посещены. Если хотя бы одна станция недостижима, сеть считается разорванной.

В методе main создаются тестовые данные для демонстрации работы валидатора. Определяются станции, перегоны и районы управления. Затем создаются два объекта Railway: первый — валидный, с полностью связной сетью, второй — невалидный, где одна станция находится вне общего графа.

Для проверки используется стандартный API валидации Jakarta Validation. Создается Validator через фабрику, после чего вызывается метод validate. Для каждого из объектов выводится результат проверки. Если объект не прошел валидацию, отображается сообщение, заданное в аннотации.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

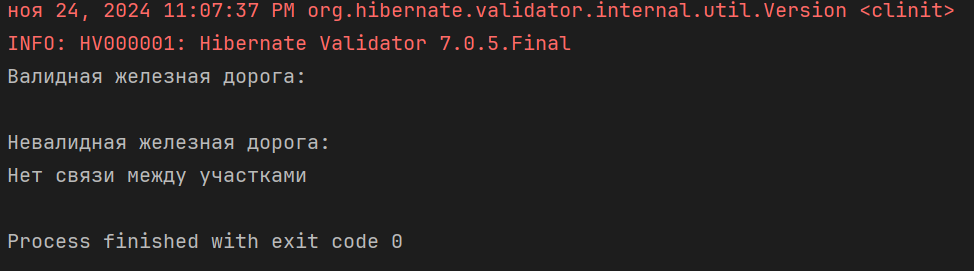


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab5\_Annotation\_Validation.Annotation;  
  
import Lab5\_Annotation\_Validation.Validator.ConnectedRailwayValidator;  
import jakarta.validation.Constraint;  
import jakarta.validation.Payload;  
  
import java.lang.annotation.\*;  
  
@Documented  
@Constraint(validatedBy = ConnectedRailwayValidator.class)  
@Target({ElementType.TYPE})  
@Retention(RetentionPolicy.RUNTIME)  
public @interface ValidRailway {  
 String message() default "Нет связи между участками";  
 Class<?>[] groups() default {};  
 Class<? extends Payload>[] payload() default {};  
}

package Lab5\_Annotation\_Validation.Model;  
  
import Lab5\_Annotation\_Validation.Annotation.ValidRailway;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.List;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@ValidRailway  
public class Railway {  
 private List<RailwayManagementArea> managementAreas; // Список районов управления  
}

package Lab5\_Annotation\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.List;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
public class RailwayManagementArea {  
 private List<Station> stations; // Список станций в районе  
 private List<TrackSection> trackSections; // Список перегонов в районе  
}

package Lab5\_Annotation\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
public class Station {  
 private String name; // Название станции  
}

package Lab5\_Annotation\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
public class TrackSection {  
 private Station stationFrom; // Станция отправления  
 private Station stationTo; // Станция прибытия  
}

package Lab5\_Annotation\_Validation.Validator;  
  
import Lab5\_Annotation\_Validation.Annotation.ValidRailway;  
import Lab5\_Annotation\_Validation.Model.Railway;  
import Lab5\_Annotation\_Validation.Model.RailwayManagementArea;  
import Lab5\_Annotation\_Validation.Model.Station;  
import Lab5\_Annotation\_Validation.Model.TrackSection;  
  
import jakarta.validation.ConstraintValidator;  
import jakarta.validation.ConstraintValidatorContext;  
  
import java.util.\*;  
  
public class ConnectedRailwayValidator implements ConstraintValidator<ValidRailway, Railway> {  
  
 @Override  
 public boolean isValid(Railway railway, ConstraintValidatorContext context) {  
 if (railway == null || railway.getManagementAreas() == null || railway.getManagementAreas().isEmpty()) {  
 return true; // Если объект пустой, пропускаем проверку  
 }  
  
 // Собираем граф станций  
 Set<Station> allStations = new HashSet<>();  
 Map<Station, List<Station>> graph = new HashMap<>();  
  
 for (RailwayManagementArea area : railway.getManagementAreas()) {  
 if (area.getStations() != null) {  
 allStations.addAll(area.getStations());  
 }  
  
 if (area.getTrackSections() != null) {  
 for (TrackSection track : area.getTrackSections()) {  
 graph.computeIfAbsent(track.getStationFrom(), k -> new ArrayList<>()).add(track.getStationTo());  
 graph.computeIfAbsent(track.getStationTo(), k -> new ArrayList<>()).add(track.getStationFrom());  
 }  
 }  
 }  
  
 if (allStations.isEmpty()) {  
 return true; // Если станций нет, объект считается валидным  
 }  
  
 // Проверяем связность графа (поиск в глубину/ширину)  
 return isGraphConnected(graph, allStations);  
 }  
  
 private boolean isGraphConnected(Map<Station, List<Station>> graph, Set<Station> allStations) {  
 if (graph.isEmpty()) return false;  
  
 Set<Station> visited = new HashSet<>();  
 Queue<Station> queue = new LinkedList<>();  
  
 // Начинаем с любой станции  
 Station startStation = allStations.iterator().next();  
 queue.add(startStation);  
  
 while (!queue.isEmpty()) {  
 Station current = queue.poll();  
 if (!visited.add(current)) {  
 continue; // Уже посещена  
 }  
 List<Station> neighbors = graph.getOrDefault(current, Collections.emptyList());  
 queue.addAll(neighbors);  
 }  
  
 // Если все станции были посещены, граф связен  
 return visited.containsAll(allStations);  
 }  
}

package Lab5\_Annotation\_Validation;  
  
import Lab5\_Annotation\_Validation.Model.Railway;  
import Lab5\_Annotation\_Validation.Model.RailwayManagementArea;  
import Lab5\_Annotation\_Validation.Model.TrackSection;  
import Lab5\_Annotation\_Validation.Model.Station;  
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.Collections;  
  
public class Laboratory5 {  
 public static void main(String[] args) {  
 // Станции  
 Station stationA = new Station("Station A");  
 Station stationB = new Station("Station B");  
 Station stationC = new Station("Station C");  
 Station stationD = new Station("Station D");  
  
 // Перегоны  
 TrackSection trackSection1 = new TrackSection(stationA, stationB);  
 TrackSection trackSection2 = new TrackSection(stationB, stationC);  
 TrackSection trackSection3 = new TrackSection(stationC, stationD);  
  
 // Районы управления  
 RailwayManagementArea area1 = new RailwayManagementArea(  
 Arrays.asList(stationA, stationB, stationC),  
 Arrays.asList(trackSection1, trackSection2)  
 );  
  
 RailwayManagementArea area2 = new RailwayManagementArea(  
 Arrays.asList(stationC, stationD),  
 Collections.singletonList(trackSection3)  
 );  
  
 // Валидная железная дорога  
 Railway validRailway = new Railway(Arrays.asList(area1, area2));  
  
 // Невалидная железная дорога  
 RailwayManagementArea disconnectedArea = new RailwayManagementArea(  
 Collections.singletonList(new Station("Station E")),  
 Collections.emptyList()  
 );  
 Railway invalidRailway = new Railway(Arrays.asList(area1, disconnectedArea));  
  
 // Валидация  
 ValidatorFactory factory = Validation.byDefaultProvider()  
 .configure()  
 .messageInterpolator(new ParameterMessageInterpolator())  
 .buildValidatorFactory();  
 Validator validator = factory.getValidator();  
  
 System.out.println("Валидная железная дорога:");  
 validator.validate(validRailway).forEach(violation -> System.out.println(violation.getMessage()));  
  
 System.out.println("\nНевалидная железная дорога:");  
 validator.validate(invalidRailway).forEach(violation -> System.out.println(violation.getMessage()));  
 }  
}

# **ВЫВОД**

В ходе выполнения лабораторной работы была разработана пользовательская аннотация @ValidRailway для проверки корректности модели данных, описывающей железнодорожную сеть. Данная аннотация предназначена для валидации структуры сети, чтобы гарантировать наличие связности между станциями.

Модель данных была построена с учетом иерархии железнодорожной системы. Станции, перегоны и районы управления представлены отдельными классами, что позволяет представить железнодорожную сеть в виде графа. Такое представление упростило реализацию проверки связности.

Основная логика валидации была реализована в классе ConnectedRailwayValidator. С помощью алгоритма обхода графа в ширину проверялась достижимость всех станций. Если хотя бы одна станция была недоступна из других, сеть считалась невалидной. Это позволило реализовать универсальную и надежную проверку.

Работа валидатора была протестирована на примерах с валидной и невалидной железнодорожной сетью. Для валидной сети проверка успешно завершилась без ошибок, а в случае невалидной сети было выведено сообщение о нарушении связности. Это подтвердило корректность работы реализованного алгоритма и аннотации.