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 №12

Выполнил: Бугаева А. Л.

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 5](#_Toc180330918)

[КОД ПРОГРАММЫ 6](#_Toc180330919)

[ВЫВОД 9](#_Toc180330920)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс Компьютер. Поле ТипПроцессора и СокетМатеринскойПлаты должны проверяться на совместимость.  Описать ограничение, проверяющие совместимость. Продемонстрировать работу с ограничением.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Сначала была создана пользовательская аннотация @CompatibleComponents. Она предназначена для проверки совместимости процессора и материнской платы. Аннотация помечается мета-аннотацией @Constraint, указывающей валидатор CompatibilityValidator, который содержит логику проверки. Аннотация применяется на уровне класса, поскольку проверяемая логика касается двух связанных полей: процессора и материнской платы.

Далее был реализован класс CompatibilityValidator, который имплементирует интерфейс ConstraintValidator. Этот класс проверяет, совпадают ли сокеты процессора и материнской платы. Если сокеты не совпадают, объект считается недопустимым. Также были предусмотрены проверки на null, чтобы избежать ошибок в случаях, когда поля не инициализированы.

В рамках работы были созданы модели для процессора, материнской платы и компьютера. Все они снабжены аннотациями Lombok для упрощения работы с кодом (генерации геттеров, сеттеров и конструкторов).

После этого в главном классе Laboratory5 была реализована логика проверки объектов на соответствие правилам аннотации. Для этого использовалась фабрика валидаторов, которая инициализирует объект Validator. Создавались различные сценарии проверки:

1. Компьютер с совместимыми процессором и материнской платой (оба используют сокет LGA1200).
2. Компьютер с несовместимыми компонентами (процессор LGA1200 и материнская плата AM5).
3. Другой компьютер с совместимыми компонентами (оба используют сокет AM4).

Код выполняет валидацию каждого созданного объекта с помощью вызова метода validator.validate(...) и выводит сообщения о нарушениях.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

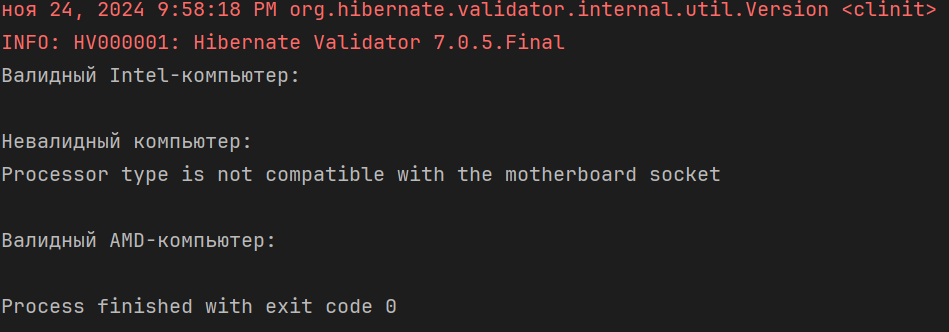


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab5\_Annotation\_Validation.Annotation;  
  
import Lab5\_Annotation\_Validation.Validator.CompatibilityValidator;  
import jakarta.validation.Constraint;  
import jakarta.validation.Payload;  
  
import java.lang.annotation.\*;  
  
@Documented  
@Constraint(validatedBy = CompatibilityValidator.class)  
@Target({ElementType.TYPE})  
@Retention(RetentionPolicy.RUNTIME)  
public @interface CompatibleComponents {  
 String message() default "Processor type is not compatible with the motherboard socket";  
 Class<?>[] groups() default {};  
 Class<? extends Payload>[] payload() default {};  
}

package Lab5\_Annotation\_Validation.Model;  
  
import Lab5\_Annotation\_Validation.Annotation.CompatibleComponents;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@CompatibleComponents  
public class Computer {  
 private Processor processor;  
 private Motherboard motherboard;  
}

package Lab5\_Annotation\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
public class Motherboard {  
 private String socket;  
}

package Lab5\_Annotation\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
public class Processor {  
 private String type;  
 private String socket;  
}

package Lab5\_Annotation\_Validation.Validator;  
  
import Lab5\_Annotation\_Validation.Annotation.CompatibleComponents;  
import Lab5\_Annotation\_Validation.Model.Computer;  
import jakarta.validation.ConstraintValidator;  
import jakarta.validation.ConstraintValidatorContext;  
  
public class CompatibilityValidator implements ConstraintValidator<CompatibleComponents, Computer> {  
  
 @Override  
 public boolean isValid(Computer computer, ConstraintValidatorContext context) {  
 if (computer.getProcessor() == null || computer.getMotherboard() == null) {  
 return true; // Проверка на null проводится другими аннотациями  
 }  
  
 String processorSocket = computer.getProcessor().getSocket();  
 String motherboardSocket = computer.getMotherboard().getSocket();  
  
 return processorSocket != null && processorSocket.equals(motherboardSocket);  
 }  
}

package Lab5\_Annotation\_Validation;  
  
import Lab5\_Annotation\_Validation.Model.Computer;  
import Lab5\_Annotation\_Validation.Model.Motherboard;  
import Lab5\_Annotation\_Validation.Model.Processor;  
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) {  
 Processor intelProcessor = new Processor("Intel", "LGA1200");  
 Motherboard intelMotherboard = new Motherboard("LGA1200");  
  
 Processor amdProcessor = new Processor("AMD", "AM4");  
 Motherboard amdMotherboard = new Motherboard("AM4");  
  
 Motherboard incompatibleMotherboard = new Motherboard("AM5");  
  
 Computer validComputer = new Computer(intelProcessor, intelMotherboard);  
 Computer invalidComputer = new Computer(intelProcessor, incompatibleMotherboard);  
 Computer validAmdComputer = new Computer(amdProcessor, amdMotherboard);  
  
 ValidatorFactory factory = Validation.byDefaultProvider()  
 .configure()  
 .messageInterpolator(new ParameterMessageInterpolator())  
 .buildValidatorFactory();  
 Validator validator = factory.getValidator();  
  
 System.out.println("Valid Intel Computer:");  
 validator.validate(validComputer).forEach(violation -> System.out.println(violation.getMessage()));  
  
 System.out.println("\nInvalid Computer:");  
 validator.validate(invalidComputer).forEach(violation -> System.out.println(violation.getMessage()));  
  
 System.out.println("\nValid AMD Computer:");  
 validator.validate(validAmdComputer).forEach(violation -> System.out.println(violation.getMessage()));  
 }  
}

# **ВЫВОД**

Лабораторная работа успешно демонстрирует использование аннотаций и пользовательских валидаторов в Jakarta Validation. Реализованный подход позволяет элегантно и удобно проводить проверки совместимости в рамках объектной модели. Использование этого механизма улучшает читаемость кода и централизует логику проверки, что является важным аспектом в проектировании масштабируемых приложений.