

Übung Java Lambda Validator

Ausgangslage

Die folgenden Listings zeigen eine Validator Lösung mit enum Typ klassisch ohne Lambda

```
Expressions:package ch.std.jup8.lambda.validator.classic;&#xA;&#xA;public enum
StringValidatorEnum {&#xA;&#xA; STRING_8 {&#xA; @Override&#xA; public void validate(String
s) throws ValidatorException {&#xA; if (s instanceof String) {&#xA; System.out.println();&#xA;
}&#xA; if (s.length() &gt; 8) {&#xA; throw new ValidatorException(this.name());&#xA;
}&#xA; } &#xA; },&#xA; STRING_16 {&#xA; @Override&#xA; public void validate(String s) throws
ValidatorException {&#xA; if (s.length() &gt; 16) {&#xA; throw new
ValidatorException(this.name());&#xA; }&#xA; } &#xA; },&#xA; STRING_32 {&#xA;
@Override&#xA; public void validate(String s) throws ValidatorException {&#xA; if (s.length()
&gt; 32) {&#xA; throw new ValidatorException(this.name());&#xA; }&#xA; } &#xA; };&#xA;
private StringValidatorEnum() {&#xA; &#xA; }&#xA; public abstract void validate(String s) throws
ValidatorException;&#xA;}package ch.std.jup8.lambda.validator.classic;&#xA;&#xA;public class
ValidatorException extends Exception {&#xA;&#xA; private static final long serialVersionUID =
3493994521286475516L;&#xA;&#xA; public ValidatorException() {&#xA; super();&#xA;
}&#xA;&#xA; public ValidatorException(String message) {&#xA; super(message);&#xA;
}&#xA;&#xA;}package ch.std.jup8.lambda.validator.classic;&#xA;&#xA;public class ValidatorDemo
{&#xA;&#xA; public static void main(String[] args) {&#xA; String s = &#34;1234567890&#34;;&#xA;
try {&#xA; StringValidatorEnum.STRING_32.validate(s);&#xA;
StringValidatorEnum.STRING_16.validate(s);&#xA;
StringValidatorEnum.STRING_8.validate(s);&#xA; } catch (ValidatorException e) {&#xA;
e.printStackTrace();&#xA; }&#xA; }&#xA; }
```

Vorlage

Die Dateien der Vorlage finden Sie auch hier

Aufgabe Supplier

In einem ersten Schritt möchten wir die Vorlage mit dem Supplier FunctionalInterface anpassen, so dass die Validierung über solches erfolgen kann. Das folgende Listing zeigt das angepasste Demo

```
Programm:package ch.std.jup8.lambda.validator.lambda.supplier;&#xA;&#xA;public class
ValidatorDemo {&#xA;&#xA; public static void main(String[] args) {&#xA; String s =
&#34;1234567890&#34;;&#xA; try {&#xA; StringValidator.STRING_32.validate() -&gt;
s);&#xA; StringValidator.STRING_16.validate() -&gt; s);&#xA;
StringValidator.STRING_8.validate() -&gt; s);&#xA; } catch (ValidatorException e) {&#xA;
e.printStackTrace();&#xA; }&#xA; }&#xA; },
```

Lösung Supplier FunctionalInterface

Eine mögliche Lösung mit Supplier FunctionalInterface finden Sie hier

Aufgabe Validator FunctionalInterface

Jetzt wollen wir die Validierungsregeln über ein eigenen Validator FunctionalInterface erweitern. Das folgende Listing zeigt die Enum Klasse:

```
package
ch.std.jup8.lambda.validator.lambda;&#xA;&#xA;&#xA;import
java.util.function.Supplier;&#xA;&#xA;public enum StringValidatorEnum {&#xA;&#xA; STRING_8(s
-&gt; { &#xA; if (s.get().toString().length() &gt; 8) {&#xA; throw new
ValidatorException(&#34;min length is 8&#34;);&#xA; }&#xA; } ),&#xA; STRING_16(s -&gt; {
&#xA; if (s.get().toString().length() &gt; 16) {&#xA; throw new ValidatorException(&#34;min
length is 16&#34;);&#xA; }&#xA; } ),&#xA; STRING_32(s -&gt; { &#xA; if
(s.get().toString().length() &gt; 32) {&#xA; throw new ValidatorException(&#34;min length is
32&#34;);&#xA; }&#xA; } );&#xA;&#xA; &#xA; private Validator&lt;String&gt; v;&#xA;
&#xA; private StringValidatorEnum(Validator&lt;String&gt; v) {&#xA; this.v = v;&#xA;
}&#xA; &#xA; public void validate(Supplier&lt;String&gt; s) throws ValidatorException
{&#xA; v.validate(s);&#xA; }&#xA;&#xA;}&#xA;Programmieren Sie das Validator
FunctionalInterface, so dass der enum Typ kompiliert. Das Demo Programm sollte identisch
```

bleiben. Das folgende Listing zeigt den Einsatz eines allgemeinen AnyValidators: // check null

try {
 AnyValidator.validate(supplier -> {
 if (supplier.get() == null) {

throw new ValidatorException();
 }
 }, null);
 } catch (ValidatorException e)
{
 e.printStackTrace();
 }
 // check is number
 try {

AnyValidator.validate(supplier -> {
 if (supplier.get() == null) {
 throw new
ValidatorException();
 }
 Integer.parseInt(supplier.get());
 },
"123");
 } catch (ValidatorException e) {
 e.printStackTrace();

} Programmieren Sie den AnyValidator (generic Interface), so dass das Listing kompiliert als Teil des
Demo Programms.

Lösung Validator FunctionalInterface

Eine mögliche Lösung mit Callable FunctionalInterface finden Sie hier

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsausschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

<https://www.simtech-ag.ch/klassisch>