Nolix

Nolix Validator

2019-02-01

Table of contents

1 Introduction

1.1 What the Nolix Validator is

The Nolix Validator is a feature for Java that provides methods to validate **arguments**.

1.2 Why to use the Nolix Validator

- The Nolix Validator can validate arguments on **many different** properties.
- The Nolix Validator produces **consistent** error messages.
- The calls of the Nolix Validator can be written in **very legible** code.

1.3 Where the Nolix Validator is

The Nolix Validator is defined in the Nolix library. To use the Nolix Validator, import the Nolix library into your project.

2 How to import the Nolix Validator class

```
import ch.nolix.core.validator2.Validator;

…

var validatorClass = Validator.class;

…
```

The Nolix validator can be found in the package ‘ch.nolix.core.validator2’.

3 How to validate that an object is not null

```
public void setContent(Object content) {

  Validator.suppose(content).isNotNull();

  …
}
```

If the given content is null, the Validator will throw a NullArgumentException. The error message of the NullArgumentException will be:

“The given argument is null.”

4 How to include the argument's name in the error message

```
public void setContent(Object content) {

  Validator.suppose(content).thatIsNamed("content").isNotNull();

  …
}
```

If the given content is null, the Validator will throw a NullArgumentException. The error message of the NullArgumentException will be:

"The given content is null."

5 Where to find common argument names as constants

```
import ch.nolix.core.constants.VariableNameCatalogue;

…

public void setContent(Object content) {

  Validator
  .suppose(content)
  .thatIsNamed(VariableNameCatalogue.CONTENT)
  .isNotNull();

  …
}

…
```

If the given content is null, the Validator will throw a NullArgumentException. The error message of the NullArgumentException will be:

"The given content is null."

The VariableNameCatalogue provides constants of common argument names and can be found in the package 'ch.nolix.core.constants'. These constants are just strings.

6 How to validate that an object is of a given type

```
public void setContent(Object content) {

  Validator.suppose(content).isOfType(Person.class);

  …
}
```

If the given content is null, the Validator will throw a NullArgumentException.

If the given content is not a Person, the Validator will throw an InvalidArgumentException. The message of the InvalidArgumentException will be:

”The given argument is not a Person.”

7 How to validate that a number is not negative

```
public void setAmount(int amount) {

  Validator.suppose(amount).isNotNegative();

  …
}
```

If the given amount negative, the Validator will throw a NegativeArgumentException. If the given amount is e.g. -25, the error message of the NegativeArgumentException will be:

"The given argument '-25' is negative."

8 How to validate that a number is in a given range

```
public void buyPencils(int amount) {

  Validator.suppose(amount).isBetween(100, 10000);

  …
}
```

If the given amount is not in the given range, the Validator will throw an OutOfRangeArgumentException. If the given amount is e.g. 50, the error message of the OutOfRangeArgumentException will be:

"The given argument '50' is not in [100, 10000]."

9 How to validate that a String is not empty

```
public void setName(String name) {

  Validator.suppose(name).isNotEmpty();

  …
}
```

If the given name is null, the Validator will throw a NullArgumentException.

If the given name is empty, the Validator will throw an EmptyArgumentException. The error message of the EmptyArgumentException will be:

"The given String is empty."

10 How to validate that a container is not empty

```
public void saveMeasuredValues(double[] measuredValues) {

  Validator.suppose(measuredValues).isNotEmpty();

  …
}
```

If the given measured values array is null, the Validator will throw a NullArgumentException.

If the given measured values array is empty, the Validator will throw an EmptyArgumentException. The error message of the EmptyArgumentException will be:

"The given array is empty."

11 How to validate that the floating point numbers in a container are not negative

```
public void saveMeasuredValues(double[] measuredValues) {

  Validator.supposeTheDoubles(measuredValues).areNotNegative();

  …
}
```

If the given measured values array is null, the Validator will throw a NullArgumentException.

If one or severals of the given measured values are negative, the Validator will throw a NegativeArgumentException. If e.g. the 5th measured value is -10, the error message of the NegativeArgumentException will be:

"The given 5th argument '-10' is negative."

12 How to validate that the Strings in a container are not empty

```
public void addCities(String[] cityNames) {

  Validator.supposeTheStrings(cityNames).areNotEmpty();

  …
}
```

If the given city names array is null, the Validator will throw a NullArgumentException.

If one or several of the given city names are empty, the Validator will throw an EmptyArgumentException. If e.g. the 5th city name is empty, the error message of the EmptyArgumentException will be:

"The given 5th argument is empty."