Nolix

Nolix Exceptions

2025-07-04

Table of contents

1 Introduction

1.1 What Nolix Exceptions are

Nolix Exceptions are for some general situations and **invalid arguments**.

1.2 Why to use Nolix Exceptions

- There exist **suitable** types of Nolix Exceptions for most situations.
- Nolix Exceptions provide **consistent** error messages.
- Nolix Exceptions have creation methods for carrying most of the available information.

1.3 Where Nolix Exceptions are

The Nolix Exceptions are in the Nolix library. To use Nolix Exceptions, import the Nolix library into your project.

2 InvalidArgumentExceptions

2.1 General

InvalidArgumentExceptions are Exceptions for invalid arguments. For example, for incoming negative amounts or empty names.

2.2 Import InvalidArgumentExceptions

```
import ch.nolix.core.errorcontrol.invalidargumentexception.…;
```

All InvalidArgumentExceptions are in the ch.nolix.core.errorcontrol.invalidargumentexception package.

2.3 Throw an InvalidArgumentException

```
public void setDeliveryAmount(int deliveryAmount) {

  if (deliveryAmount < 0) {
    throw NegativeArgumentException.forArgument(deliveryAmount);
  }
  …
}
```

The forArgument static method of NegativeArgumentException creates a new NegativeArgumentException for the given argument.

For example, if the given argument is -25, the error message of the NegativeArgumentException will be:

The given argument '-25' is negative.

2.4 Include the argument's name in an InvalidArgumentExcetpion

```
public void setDeliveryAmount(int deliveryAmount) {

  if (deliveryAmount < 0) {
    throw
    NegativeArgumentException.forArgumentAndArgumentName(
      deliveryAmount,
      "delivery amount"
    );
  }
  …
}
```

The forArgumentAndArgumentName static method of NegativeArgumentException creates a new NegativeArgumentException for the given argument and argument name.

For example, if the given delivery amount is -25, the error message of the NegativeArgumentException will be:

The given deliveryAmount '-25' is negative.

2.5 Use constants for common argument names

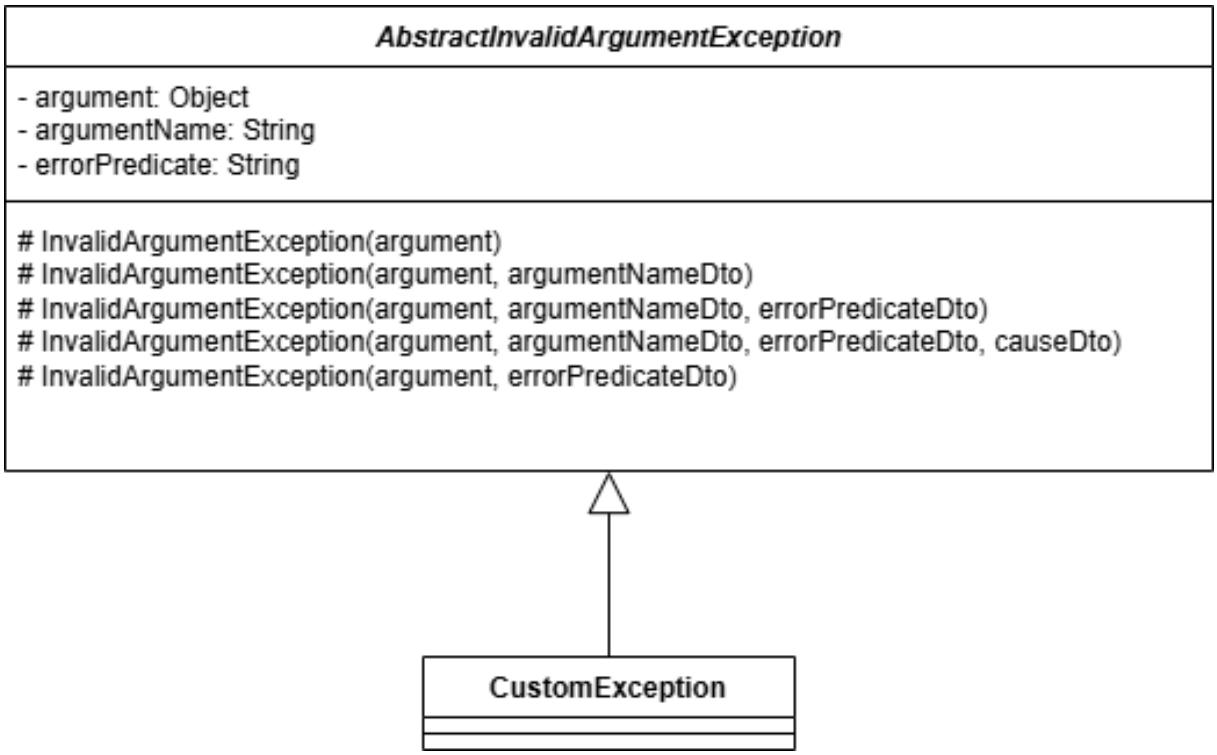
```
import
ch.nolix.coreapi.programatomatomapi.variableapi.LowerCaseVariableCatalog;
…
public void setAmount(int amount) {

  if (amount < 0) {
    throw
    NegativeArgumentException.forArgumentNameAndArgument(
      amount,
      LowerCaseVariableCatalog.AMOUNT
    );
  }
  …
}
```

The LowerCaseVariableCatalog provides variable names in lower case. The LowerCaseVariableCatalog is in the ch.nolix.coreapi.programatomapi.variableapi package.

For example, if the given amount is -25, the error message of the NegativeArgumentException will be:

The given amount ‘-25’ is negative.

2.6 Define a custom InvalidArgumentException




```
public class CustomException extends AbstractInvalidArgumentException {
  …
}
```

A custom InvalidArgumentException can be defined by inheriting from AbstractInvalidArgumentException. The constructors of the custom InvalidArgumentException must call one of the super constructors of the AbstractInvalidArgumentException.

The argument of an AbstractInvalidArgumentException can be null.

2.7 Types of InvalidArgumentExceptions

2.7.1 For general cases

Exception	Suppose
ArgumentIsNullException	Supposed to be thrown when a given argument is undesirably **null**.
ClosedArgumentException	Supposed to be thrown when an argument is undesirably **closed**.
EqualArgumentException	Supposed to be thrown when a given argument undesirably **equals** a given value.
InvalidArgumentException	Supposed to be thrown when a given argument is **not valid**.
UnequalArgumentException	Supposed to be thrown when a given argument does undesirably **not equal** a given value.
UnrepresentingArgumentException	Supposed to be thrown when a given argument **does not represent** an object of a wanted type.

2.7.2 For numbers

Exception	Suppose
ArgumentIsInRangeException	Supposed to be thrown when a given argument is **in an unwanted range**.
ArgumentIsOutOfRangeException	Supposed to be thrown when a given argument is **not in a wanted range**.
BiggerArgumentException	Supposed to be thrown when a given argument is undesirably **bigger** than a given maximum.
NegativeArgumentException	Supposed to be thrown when a given argument is undesirably **negative**.
NonNegativeArgumentException	Supposed to be thrown when a given argument is undesirably **not negative**.
NonPositiveArgumentException	Supposed to be thrown when a given argument is undesirably **not positive**.
PositiveArgumentException	Supposed to be thrown when a given argument is undesirably **positive**.
SmallerArgumentException	Supposed to be thrown when a given argument is undesirably **smaller** than a given minimum.

2.7.3 For structure

Exception	Suppose
ArgumentBelongsToParentException	Supposed to be thrown when a given argument **belongs** undesirably to a parent.
ArgumentContainsElementException	Supposed to be thrown when a given argument **contains** undesirably a given element.
ArgumentDoesNotBelongToParentException	Supposed to be thrown when a given argument does undesirably **not belong** to a parent.
ArgumentDoesNotContainElementException	Supposed to be thrown when a given argument does undesirably **not contain** a given element.
EmptyArgumentException	Supposed to be thrown when a given argument is undesirably **empty**.
NonEmptyArgumentException	Supposed to be thrown when a given argument is undesirably **not empty**.

2.7.4 For attributes

Exception	Suppose
ArgumentDoesNotHaveAttributeException	Supposed to be thrown when a given argument does undesirably **not have a given attribute**.
ArgumentHasAttributeException	Supposed to be thrown when a given argument has undesirably a given **attribute**.

2.7.5 For flow control

Exception	Suppose
ArgumentDoesNotSupportMethodException	Supposed to be thrown when a given argument does undesirably **not support a given method**.
UnsupportedCaseException	Supposed to be thrown when a given case is undesirably **not supported**.