Table Of Content

Package org.sidiff.core.compare.comparefunctions

Class Summary

ChildrenCF

Compare function for comparing two nodes' children.

CompareFunction

Represents the abstract superclass for each compare function.

CompareFunction.Policy

The compare function's policy.

MaximumSimilarityCF

If both nodes are not `null`, which is checked by an assert statement, this compare functions returns `1` in all cases.

MinimumSimilarityCF

If both nodes are not `null`, which is checked by an assert statement, this compare functions returns `0` in all cases.

NeighboursMultipleReferenceCF

This is a compare function for comparing two nodes' neighbours.

NeighboursSingleReferenceCF

This is a compare function for comparing two nodes' neighbours.

ParentCF

Compare function for comparing two nodes' parents.

RemoteNodesCF

This is a compare function for comparing two nodes based on the comparison of remote nodes.

SelfCF

Compare function for comparing local properties of two nodes.

ShouldNotOccurDummy

This function is a dummy to ensure the not occur warranty.

SingleNeighbourCF

This is a compare function for comparing two nodes' neighbours.

org.sidiff.core.compare.comparefunctions

Class ChildrenCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.ChildrenCF
```

< Constructors > < Methods >

---

public class **ChildrenCF**
extends CompareFunction

Compare function for comparing two nodes' children.

This compare function expects exactly one parameter fragment: The comparator that should be used to compare the two collections "children of A" and "children of B".

All types of comparators are allowed for this compare function.

Author:

Pit Pietsch

Constructors

ChildrenCF

```
public  ChildrenCF(EClass dedicatedClass,
                   CompareFunction.Policy policy,
                   float weight,
                   java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - The compare function's meta-model
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter

This compare function expects exactly one parameter fragment: the comparator that should be used to compare the two collections "children of A" and "children of B".

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
              throws null
```

Calculates the similarity of two nodes according to the following proceeding:

(1) If one of the nodes is null, the calculation is aborted. This is checked by an assertion.

(2) Else if both nodes do not have any children, a NothingToCompareException is thrown.

(3) Else, similarity calculation is delegated to the comparator specified as parameter.

Parameters:

nodeInA - The node in model A
nodeInB - The node in model B

Returns:

The calculated similarity

Overrides:

compare in class CompareFunction

Throws:

null - if none of both nodes does have any children.

---

setContext

```
public void setContext(ServiceContext context)
```

Overrides:

setContext in class CompareFunction

---

org.sidiff.core.compare.comparefunctions

Class CompareFunction

```
java.lang.Object
    |
    +--org.sidiff.core.compare.comparefunctions.CompareFunction
```

Direct Known Subclasses:

ChildrenCF, MaximumSimilarityCF, MinimumSimilarityCF, NeighboursMultipleReferenceCF, NeighboursSingleReferenceCF, ParentCF, RemoteNodesCF, SelfCF, ShouldNotOccurDummy, SingleNeighbourCF

< Fields > < Constructors > < Methods >

---

public abstract class **CompareFunction**
extends java.lang.Object

Represents the abstract superclass for each compare function.

Fields

context

```
protected ServiceContext context
```

Holds the ServiceContext for this Compare Function.

---

paramItems

```
protected java.lang.String[] paramItems
```

Holds the parameter string fragments and can be used by subclasses.

Constructors

CompareFunction

```
protected  CompareFunction(EClass dedicatedClass,
                           CompareFunction.Policy policy,
                           float weight,
                           java.lang.String parameter)
```

The Constructor that has to be used by subclasses.

Initializes attribute values, extracts parameter segments and checks parameter segment correctness via assertion.

Parameters:

dedicatedClass - The dedicated meta-class (out of the document type specific meta-model) on whose instances this compare function operates on.
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter

Methods

compare

```
public abstract float compare(EObject nodeInA,
                              EObject nodeInB)
```

Compare two nodes using the specific implementation

Parameters:

nodeInA - The node in model A
nodeInB - The node in model B

Returns:

The calculated similarity

---

getEClass

```
public EClass getEClass()
```

Get the compare function's Meta-Model

Returns:

Returns the compare function's weight

---

getPolicy

```
public CompareFunction.Policy getPolicy()
```

Get the compare function's policy

Returns:

Returns the compare function's policy

---

getSignature

```
public java.lang.String getSignature()
```

---

getWeight

```
public float getWeight()
```

Get the compare function's weight

Returns:

Returns the compare function's weight

---

setContext

```
public void setContext(ServiceContext context)
```

Sets the ServiceContext for this Compare Function

Parameters:

context - the Service Context

---

org.sidiff.core.compare.comparefunctions

Class CompareFunction.Policy

```
java.lang.Object
    |
    +--java.lang.Enum
        |
        +--org.sidiff.core.compare.comparefunctions.CompareFunction.Policy
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable

< Fields > < Methods >

---

public static final class **CompareFunction.Policy**
extends java.lang.Enum

The compare function's policy. It tells whether the function is mandatory, optional, or ignoreable.

If a mandatory function fails, the comparison will be aborted. If a scale function fails, the comparison will continue without this function and its weight will be extracted. If a zero function fails, the comparison will continue. The compared function will be taken as returning 0.0. If an abort function fails, the comparison will return 0.0 as overall similarity.

Fields

fulfilled

```
public static final CompareFunction.Policy fulfilled
```

---

mandatory

```
public static final CompareFunction.Policy mandatory
```

---

unfulfilled

```
public static final CompareFunction.Policy unfulfilled
```

Methods

valueOf

```
public static CompareFunction.Policy valueOf(java.lang.String name)
```

---

values

```
public static
org.sidiff.core.compare.comparefunctions.CompareFunction.Policy[] values()
```

---

org.sidiff.core.compare.comparefunctions

Class MaximumSimilarityCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.MaximumSimilarityCF
```

< Constructors > < Methods >

---

public class **MaximumSimilarityCF**
extends CompareFunction

If both nodes are not `null`, which is checked by an assert statement, this compare functions returns `1` in all cases.

This compare function has no parameters.

Constructors

MaximumSimilarityCF

```
public  MaximumSimilarityCF(EClass dedicatedClass,
                            CompareFunction.Policy policy,
                            float weight)
```

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
```

Overrides:

compare in class CompareFunction

---

org.sidiff.core.compare.comparefunctions

Class MinimumSimilarityCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.MinimumSimilarityCF
```

< Constructors > < Methods >

---

public class **MinimumSimilarityCF**
extends CompareFunction

If both nodes are not `null`, which is checked by an assert statement, this compare functions returns `0` in all cases.

This compare function has no parameters.

Constructors

MinimumSimilarityCF

```
public  MinimumSimilarityCF(EClass dedicatedClass,
                            CompareFunction.Policy policy,
                            float weight)
```

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
```

Overrides:

compare in class CompareFunction

org.sidiff.core.compare.comparefunctions

Class NeighboursMultipleReferenceCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.NeighboursMultipleReferenceCF
```

< Fields > < Constructors > < Methods >

public class **NeighboursMultipleReferenceCF**
extends CompareFunction

This is a compare function for comparing two nodes' neighbours.

The neighbours are either

(1) all local references (defined by ConstantValues.PARAM_LOCAL_REFERENCES) or

(2) all references (defined by ConstantValues.PARAM_ALL_REFERENCES).

This compare function takes two parameters:

(1) The comparator that should be used to compare the neighbours of node in model A with those in model B.

(2) A constant value (either ConstantValues.PARAM_LOCAL_REFERENCES or ConstantValues.PARAM_ALL_REFERENCES) that specifies the neighbours.

Attention: This compare function is expensive.

Author:
Pit Pietsch

Fields

referenceList

```
protected java.util.List referenceList
```
List of references to be compared

Constructors

NeighboursMultipleReferenceCF

```
public  NeighboursMultipleReferenceCF(EClass dedicatedClass,
                                      CompareFunction.Policy policy,
                                      float weight,
                                      java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - The compare function's meta-model
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter.

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
              throws null
```

Calculates the similarity of two nodes according to the following proceeding:

(1) If one of the nodes is null, the calculation is aborted. This is checked by an assertion.

(2) Else if both neighbour sets given by the parameter constant are empty, a NothingToCompareException is thrown.

(3) Else, similarity calculation is delegated to the comparator specified as parameter.

Compares the two nodes on the basis of their neigbour sets given by the reference name and using the specified comparator.

Parameters:

nodeInA - The node in model A
nodeInB - The node in model B

Returns:

The calculated similarity

Overrides:

compare in class CompareFunction

Throws:

null - if both neighbour sets given by the parameter constant are empty.

---

setContext

```
public void setContext(ServiceContext context)
```

Overrides:

setContext in class CompareFunction

org.sidiff.core.compare.comparefunctions

Class NeighboursSingleReferenceCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.NeighboursSingleReferenceCF
```

< Fields > < Constructors > < Methods >

public class **NeighboursSingleReferenceCF**
extends CompareFunction

This is a compare function for comparing two nodes' neighbours.

The neighbours are defined by the specific rolename of the reference.

This compare function takes two parameters:

(1) The comparator that should be used to compare the neighbours of node in model A with those in model B.

(2) A string value that specifies the role name of the reference in the meta-model referencing the neighbours.

Author:
Pit Pietsch

Fields

reference

```
protected EReference reference
```
The references to be compared

Constructors

NeighboursSingleReferenceCF

```
public  NeighboursSingleReferenceCF(EClass dedicatedClass,
                                    CompareFunction.Policy policy,
                                    float weight,
                                    java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - The compare function's meta-model
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter.

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
              throws null
```

Calculates the similarity of two nodes according to the following proceeding:

(1) If one of the nodes is null, the calculation is aborted. This is checked by an assertion.

(2) Else if both neighbour sets given by the reference name are empty, a NothingToCompareException is thrown.

(3) Else, similarity calculation is delegated to the comparator specified as parameter.

Compares the two nodes on the basis of their neigbour sets given by the reference name and using the specified comparator.

Parameters:

nodeInA - The node in model A
nodeInB - The node in model B

Returns:

The calculated similarity

Overrides:

compare in class CompareFunction

Throws:

null - if both neighbour sets given by the reference name are empty.

---

setContext

```
public void setContext(ServiceContext context)
```

Overrides:

setContext in class CompareFunction

org.sidiff.core.compare.comparefunctions

Class ParentCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.ParentCF
```

< Constructors > < Methods >

public class **ParentCF**
extends CompareFunction

Compare function for comparing two nodes' parents.

This compare function expects exactly one parameter fragment: The comparator that should be used to compare the parent of node in model A with the parent of node in model B.

All types of comparators are allowed for this compare function.

Author:

Pit Pietsch

Constructors

ParentCF

```
public  ParentCF(EClass dedicatedClass,
                 CompareFunction.Policy policy,
                 float weight,
                 java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - The compare function's meta-model.
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter.

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
              throws null
```

Extracts the parents and calculates the similarity of two nodes according to the following proceeding:

(1) If one of the nodes is null, the calculation is aborted. This is checked by an assertion.

(2) Else if both nodes do not have parents, a NothingToCompareException is thrown.

(3) Else, similarity calculation is delegated to the comparator specified as parameter.

Parameters:

context - The service context containing elements for fulfilling similarity calculation
nodeInA - The node in model A.
nodeInB - The node in model B.

Returns:

The calculated similarity.

Overrides:

compare in class CompareFunction

Throws:

null - if both nodes do not have parents

---

setContext

```
public void setContext(ServiceContext context)
```

Overrides:

setContext in class CompareFunction

---

org.sidiff.core.compare.comparefunctions

Class RemoteNodesCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.RemoteNodesCF
```

< Fields > < Constructors > < Methods >

---

public class **RemoteNodesCF**
extends CompareFunction

This is a compare function for comparing two nodes based on the comparison of remote nodes.

The remote nodes are defined by a path expression based on the document type specific meta-model.

This compare function takes two parameters:

(1) The comparator that should be used to compare the remote nodes collected by the given path expression.

(2) A string value that specifies the a path expression identifying the remote nodes.

Attention: this compare function is expensive!

Author:

Pit Pietsch

Fields

path

```
protected EMFPath path
```

The path to the remote elements

Constructors

RemoteNodesCF

```
public  RemoteNodesCF(EClass dedicatedClass,
                      CompareFunction.Policy policy,
                      float weight,
                      java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - The compare function's meta-model
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter.

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
              throws null
```

Calculates the similarity of two nodes according to the following proceeding:

(1) If one of the nodes is null, the calculation is aborted. This is checked by an assertion.

(2) Else if both remote node sets given by the path expression are empty, a NothingToCompareException is thrown.

(3) Else, similarity calculation is delegated to the comparator specified as parameter.

Parameters:

nodeInA - The node in graph A
nodeInB - The node in graph B

Returns:

The calculated similarity

Overrides:

compare in class CompareFunction

Throws:

null - if both remote node sets given by the path expression are empty

---

setContext

```
public void setContext(ServiceContext context)
```

Overrides:

setContext in class CompareFunction

---

org.sidiff.core.compare.comparefunctions

Class SelfCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.SelfCF
```

< Constructors > < Methods >

---

public class **SelfCF**
extends CompareFunction

Compare function for comparing local properties of two nodes.

This compare function expects exactly one parameter fragment: The comparator that should be used to do the comparison.

All types of comparators are allowed for this compare function.

Author:

Pit Pietsch

Constructors

SelfCF

```
public  SelfCF(EClass dedicatedClass,
               CompareFunction.Policy policy,
               float weight,
               java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - The compare function's meta-model.
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter.

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
```

Delegates the comparison to the specified comparator.

If one of both nodes is `null`, calculation is aborted. This is checked via assertion.

Parameters:

context - The service context containing elements for fulfilling similarity calculation
nodeInA - The node in model A.
nodeInB - The node in model B.

Returns:

The calculated similarity.

Overrides:

compare in class CompareFunction

setContext

```
public void setContext(ServiceContext context)
```

Overrides:

setContext in class CompareFunction

---

org.sidiff.core.compare.comparefunctions

Class ShouldNotOccurDummy

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.ShouldNotOccurDummy
```

< Constructors > < Methods >

---

public class **ShouldNotOccurDummy**
extends CompareFunction

This function is a dummy to ensure the not occur warranty.

Constructors

ShouldNotOccurDummy

```
public  ShouldNotOccurDummy(EClass dedicatedClass,
                            CompareFunction.Policy policy,
                            float weight)
```

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
```

Overrides:

compare in class CompareFunction

---

toString

`public java.lang.String` **`toString`**`()`

Overrides:

toString in class java.lang.Object

---

org.sidiff.core.compare.comparefunctions

Class SingleNeighbourCF

```
java.lang.Object
    |
    +--CompareFunction
        |
        +--org.sidiff.core.compare.comparefunctions.SingleNeighbourCF
```

< Fields > < Constructors > < Methods >

---

public class **SingleNeighbourCF**
extends CompareFunction

This is a compare function for comparing two nodes' neighbours. It only works when the cardinality of the reference is exactly 1.

The two neighbour nodes are defined by the specific rolename of the reference.

This compare function takes two parameters:

(1) The comparator that should be used to compare the neighbours of node in model A with those in model B.

(2) A string value that specifies the role name of the reference in the meta-model referencing the neighbours.

Author:

Pit Pietsch

Fields

reference

`protected EReference` **`reference`**

The references to be compared

Constructors

SingleNeighbourCF

```
public  SingleNeighbourCF(EClass dedicatedClass,
                          CompareFunction.Policy policy,
                          float weight,
                          java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - The compare function's meta-model
policy - The compare function's policy.
weight - The compare function's weight.
parameter - The compare function's parameter.

Methods

compare

```
public float compare(EObject nodeInA,
                     EObject nodeInB)
              throws null
```

Calculates the similarity of two nodes according to the following proceeding:

(1) If one of the origin nodes is null, the calculation is aborted. This is checked by an assertion.

(2) If if both neighbour nodes are null, a NothingToCompareException is thrown.

(3) Else if only one of the neighbour nodes is null, a similarity value if 0 is returned

(3) Else a similarity calculation is delegated to the comparator specified as parameter.

Compares the two nodes on the basis of their neigbour sets given by the reference name and using the specified comparator.

Parameters:

nodeInA - The node in model A
nodeInB - The node in model B

Returns:

The calculated similarity

Overrides:

compare in class CompareFunction

Throws:

null - if both neighbour sets given by the reference name are empty.

setContext

```
public void setContext(ServiceContext context)
```

Overrides:

setContext in class CompareFunction

Package org.sidiff.core.compare.comparefunctions.comp

Class Summary

CEqualType

This comparator compares two object based on their class.

CEquals

This comparator compares two object based on their .equals()-method.

CHeaviside

This comparator mimics the characteristics of a heaviside-function.

CHeavisideReverse

This comparator compares two object based on the specified inner comparator.

ECAnnotation

This comparator compares a specified annotation of two elements based on the specified inner comparator.

ECAttributeDynamic

This comparator compares the values of the specified attributes of two elements based on the given inner comparator.

ECAttributeStatic

This comparator compares the attributes of two elements based on the specified inner comparator.

ECEqualIndex

This comparator compares two object based on their class.

ECMatched

This comparator checks if two elements are matched.

ECMatchedOrSimilar

This comparator checks whether two elements are matched.

ECSimilarity

This comparator looks up and returns the similarity-value of two elements.

LCAlignedList

This comparator compares two aligned list of the same size (this constrained is tested by assertion) element by element.

LCLongestCommonSubsequence

This comparator compares two aligned list based on an LongestCommonSubsequence-algorithm.

SCGreedyMatchedOrSimilar

This comparator calculates a similarity value between two sets of elements based on matchings

and similarities.

SCGreedySimilarity

The calculation of the similarity value is separated in two parts:

(1) In a first step the algorithm looks tries to match the elements as best as possible based on their similarity.

SCMatched

This comparator counts the number of matched elements between to sets and normalize the sum by dividing through the number of elements in the bigger set.

SCSize

This comparator compares two collections based on their .size()-method.

VCCharacterEqualsCI

This comparator performs an case-insensitive .equals()-comparison between two Character.

VCGauss

This comparator compares two numerical values based on a gauss-calculation.

VCStringEMFCompare

The suggested string similarity measure by Xing/Stroulia This code is blatantly stolen from org.eclipse.emf.compare.match.internal.statistic.NameSimilarity

VCStringEqualsCI

This comparator performs an case-insensitive .equals()-comparison between two Strings.

VCStringIndexOf

This comparator compares two string values based on a their index-of similarity.

VCStringLCS

This comparator compares two string values based on a longest common subsequence calculation.

---

org.sidiff.core.compare.comparefunctions.comparators

Class CEqualType

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.CEqualType
```

< Constructors > < Methods >

---

public class **CEqualType**
extends AbstractValueComparator

This comparator compares two object based on their class. The returned similarity is

- 1f in case the given objects are of equal-type or
- 0f in case they differ.

No additional parameters are necessary to use this comparator.

The following example-configuration shows how to use the CEqualType comparator to compare two Elements based on their type. In case they are of equal type, 1f is returned, 0f otherwise.

<CompareFunction class="SelfCF" weight="1" parameter="CEqualType"/>

Author:

Pit Pietsch

Constructors

CEqualType

```
public  CEqualType(EClass dedicatedClass,
                   java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

Compares two elements based on their .equals()-method. In case the elements are equal a similarity-value of 1f is returned, 0f otherwise.

---

init

```
public void init(ServiceContext serviceContext)
```

org.sidiff.core.compare.comparefunctions.comparators

Class CEquals

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.CEquals
```

< Constructors > < Methods >

---

public class **CEquals**
extends AbstractValueComparator

This comparator compares two object based on their .equals()-method. The returned similarity is

- 1f in case the given objects are equal or
- 0f in case they differ.

No additional parameters are necessary to use this comparator.

The following example-configuration shows how to use the CEquals comparator to compare two (Operation-)elements based on their "isStatic"-Attribute. In case the values of the isStatic-attributes are equal, 1f is returned, 0f otherwise.

<CompareFunction class="SelfCF" weight="1"
parameter="ECAttributeStatic[CEquals;Operation;isStatic]"/>

org.sidiff.core.compare.comparefunctions.comparators.ECAttributeStatic

Author:

Pit Pietsch

Constructors

CEquals

```
public  CEquals(EClass dedicatedClass,
                java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

Compares two elements based on their .equals()-method. In case the elements are equal a similarity-value of 1f is returned, 0f otherwise.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class CHeaviside

```
java.lang.Object
    |
    +--AbstractComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.CHeaviside
```

< Constructors > < Methods >

---

public class **CHeaviside**
extends AbstractComparator

This comparator mimics the characteristics of a heaviside-function. It compares two object based on the specified inner comparator and in case the similarity-value returned by the inner comparator is below the given threshold 0f is returned, otherwise 1f.

The parameter of this comparefunction specifies at position

- 0: The inner comparator and it's configuration
- 1: The threshold used for the heaviside-decision.

The following example-configuration shows how to use the CHeviside comparator in combination with ECSimilarity to compare two elements based on their similarity. In case the similarity-value is above the specified threshold (here 0.5f) a similarity of 1f is returned, 0f otherwise.

<CompareFunction class="SelfCF" weight="1" parameter="CHeaviside[ECSimilarity;0.5]"/>

org.sidiff.core.compare.comparefunctions.comparators.ECSimilarity

Author:

Pit Pietsch

Constructors

CHeaviside

```
public  CHeaviside(EClass dedicatedClass,
                   java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

Compares two Elements based on the specified inner comparator. If the returned similarity-value is below the given threshold, 0f is returned, otherwise 1f.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class CHeavisideReverse

```
java.lang.Object
    |
    +--AbstractComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.CHeavisideReverse
```

< Constructors > < Methods >

---

public class **CHeavisideReverse**
extends AbstractComparator

This comparator compares two object based on the specified inner comparator. In case the similarity-value returned by the inner comparator is below the given threshold 0f is returned, otherwise the calculated similarity value is passed. This comparator is similar to the comparator CHeaviside.

The parameter of this comparefunction specifies at position

- 0: The inner comparator and it's configuration

- 1: The threshold used for the heaviside-decision.

The following example-configuration shows how to use the CHeavisideReverse comparator in combination with ECSimilarity to compare two elements based on their similarity. In case the similarity-value is above the specified 0.5f-threshold 1f is returned, the similarity-value.

<CompareFunction class="SelfCF" weight="1" parameter="CHeavisideReverse[ECSimilarity;0.5]"/>

org.sidiff.core.compare.comparefunctions.comparators.CHeaviside

org.sidiff.core.compare.comparefunctions.comparators.ECSimilarity

Author:

Pit Pietsch

Constructors

CHeavisideReverse

```
public  CHeavisideReverse(EClass dedicatedClass,
                          java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

Compares two Elements based on the specified inner comparator. If the returned similarity-value is below the given threshold, 0f is returned, otherwise the similarity-value.

---

init

```
public void init(ServiceContext serviceContext)
```

org.sidiff.core.compare.comparefunctions.comparators

Class ECAnnotation

```
java.lang.Object
    |
    +--AbstractElementComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.ECAnnotation
```

< Constructors > < Methods >

---

public class **ECAnnotation**
extends AbstractElementComparator

This comparator compares a specified annotation of two elements based on the specified inner comparator. It is tested by assertion whether the annotations exists or not. The parameter of this comparefunction specifies at position

- 0: The inner comparator and it's configuration
- 1: The name of the annotation to be compared

The following example-configuration shows how to use the ECAttributeDynamic comparator to compare two elements based on the equality of their TypePath-annotation. In case both attribute-values are equal 1f is returned, 0f otherwise.

<CompareFunction class="SelfCF" weight="1" parameter="ECAnnotation[CEquals;TypePath]"/>

org.sidiff.core.compare.comparefunctions.comparators.CEquals

Author:

Pit Pietsch

Constructors

ECAnnotation

```
public  ECAnnotation(EClass dedicatedClass,
                     java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        EObject elementA,
                        EObject elementB)
```

The specified annotation is extracted from both elements and a similarity value is calculated by the inner comparator. It is assured by assertion that the annotation exist in both elements.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class ECAttributeDynamic

```
java.lang.Object
    |
    +--AbstractElementComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.ECAttributeDynamic
```

< Constructors > < Methods >

---

public class **ECAttributeDynamic**
extends AbstractElementComparator

This comparator compares the values of the specified attributes of two elements based on the given inner comparator.

Warning: This comparefunction is expensive!

This comparator compares the values of the specified attributes of two elements based on the given inner comparator. If the attribute does not exist in at least one of the elements a AttributeDoesNotExist-Exception is thrown. Because it is tested for every call of the comparator if the attributed exists in both elements, this comparator is expensive and it's use is discouraged. The comparator ECAttributeStatic should be used whenever possible instead. The parameter of this comparefunction specifies at position

- 0: The inner comparator and it's configuration
- 1: The name of the attribute to be compared

The following example-configuration shows how to use the ECAttributeDynamic comparator to compare two elements based on the equality of their name-attribute. In case both attribute-values are equal 1f is returned, 0f otherwise.

<CompareFunction class="SelfCF" weight="1" parameter="ECAttributeDynamic[CEquals;;name]"/>

org.sidiff.core.compare.comparefunctions.comparators.ECAttributeStatic

AttributeNotExistsException

Author:

Pit Pietsch

Constructors

ECAttributeDynamic

```
public  ECAttributeDynamic(EClass dedicatedClass,
                           java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        EObject elementA,
                        EObject elementB)
```

The given elements are compared based on the specified attribute and the inner comparator. In case the attribute does not exist in at least on of the elements an AttributeNotExistsExcpetion is thrown. Otherwise the similarity-value calculated by the inner comparator is returned.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class ECAttributeStatic

```
java.lang.Object
    |
    +--AbstractElementComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.ECAttributeStatic
```

< Constructors > < Methods >

public class **ECAttributeStatic**
extends AbstractElementComparator

This comparator compares the attributes of two elements based on the specified inner comparator. If the attribute does not exist in at least one of the elements a AttributeDoesNotExist-Exception is thrown. The parameter of this comparefunction specifies at position

- 0: The inner comparator and it's configuration
- 1: The name of the target class
- 2: The name of the attribute to be compared

The following example-configuration shows how to use the ECAttributeDynamic comparator to compare two (Operation-)elements based on the equality of their name-attribute. In case both attribute-values are equal 1f is returned, 0f otherwise.

<CompareFunction class="SelfCF" weight="1" parameter="ECAttributeStatic[CEquals;operation;name]"/>

org.sidiff.core.compare.comparefunctions.comparators.ECAttributeDynamic

AttributeNotExistsException

Author:
Pit Pietsch

Constructors

ECAttributeStatic

```
public  ECAttributeStatic(EClass dedicatedClass,
                          java.lang.String parameter)
```

Instantiates the comparator used to do the comparison and extracts the {@link EAttribute} which should be compared from the metamodel. In case the specified attribute does not exist in the specified target class an AttributeNotExistsException is thrown. With respect to the parameters `context` and `parameter`, please refer to the constructor in base class {@link AbstractComparator}.

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        EObject nodeInA,
                        EObject nodeInB)
```

The given elements are compared based on the specified attribute and the inner comparator and the calculated similarity-value is returned.

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class ECEqualIndex

```
java.lang.Object
    |
    +--AbstractElementComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.ECEqualIndex
```

< Constructors > < Methods >

---

public class **ECEqualIndex**
extends AbstractElementComparator

This comparator compares two object based on their class. The returned similarity is

- 1f in case the given objects have the same index in their container or
- 0f in case the indices differ.

No additional parameters are necessary to use this comparator.

However, the optional parameter flag "includesize" forces a comparison of the size of the container in which the compared element is contained.

The following example-configuration shows how to use the ECEqualIndex comparator to compare two Elements based on their index. In case they have the same index, 1f is returned, 0f otherwise.

<CompareFunction class="SelfCF" weight="1" parameter="ECEqualIndex"/>

Author:

Sven Wenzel

Constructors

ECEqualIndex

```
public  ECEqualIndex(EClass dedicatedClass,
                     java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        EObject elementA,
                        EObject elementB)
```

Compares two elements based on their index in their container. In case the indices are equal a similarity-value of 1f is returned, 0f otherwise.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class ECMatched

```
java.lang.Object
    |
    +--AbstractElementComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.ECMatched
```

< Constructors > < Methods >

---

public class **ECMatched**
extends AbstractElementComparator

This comparator checks if two elements are matched. If they are matched 1f is returned, otherwise 0f. This comparator does not need a parameter.

Author:

Pit Pietsch

Constructors

ECMatched

```
public  ECMatched(EClass dedicatedClass,
                  java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        EObject elementA,
                        EObject elementB)
```

Checks if the two elements to be compared are already matched. If they are matched 1f is returned, otherwise 0f

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class ECMatchedOrSimilar

```
java.lang.Object
    |
    +--AbstractElementComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.ECMatchedOrSimilar
```

< Constructors > < Methods >

---

public class **ECMatchedOrSimilar**
extends AbstractElementComparator

This comparator checks whether two elements are matched. If they are matched 1f is returned. If they are not corresponding the similarity-value between them is looked up and returned.

This comparator does not need a parameter.

Author:

Pit Pietsch

Constructors

ECMatchedOrSimilar

```
public  ECMatchedOrSimilar(EClass dedicatedClass,
                           java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     EObject elementA,
                     EObject elementB)
```

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class ECSimilarity

```
java.lang.Object
    |
    +--AbstractElementComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.ECSimilarity
```

< Constructors > < Methods >

---

public class **ECSimilarity**
extends AbstractElementComparator

This comparator looks up and returns the similarity-value of two elements.

This comparator does not need a parameter.

Author:

Pit Pietsch

Constructors

ECSimilarity

```
public  ECSimilarity(EClass dedicatedClass,
                     java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     EObject elementA,
                     EObject elementB)
```

The similarity-value of the two elements is returned.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class LCAlignedList

```
java.lang.Object
    |
    +--AbstractListComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.LCAlignedList
```

< Constructors > < Methods >

---

public class **LCAlignedList**
extends AbstractListComparator

This comparator compares two aligned list of the same size (this constrained is tested by assertion) element by element. How the single elements are to be compared is specified by the inner comparator. The element-wise calculated similarity values are added and normalized by the size of the lists.

The parameter of this comparefunction specifies at position

- 0: The inner comparator and it's configuration

The following example-configuration shows how to use the LCAlignedList comparator in combination with ECMatched. The children of the dedicated class are compared based on whether they are already matched or not. To use LCAlignedList the the children have to be ordered and of the same number.

<CompareFunction class="ChildrenCF" weight="1" parameter="LCAlignedList[ECMatched]"/>

org.sidiff.core.compare.comparefunctions.comparators.ECMatched

Author:

Pit Pietsch

Constructors

LCAlignedList

```
public  LCAlignedList(EClass dedicatedClass,
                      java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        java.util.List collectionA,
                        java.util.List collectionB)
```

The passed lists of elements are compared element by element based on the inner comparator. The element-wise calculated similarity-values are added and normalized by the size of the lists.

---

init

```
public void init(ServiceContext serviceContext)
```

org.sidiff.core.compare.comparefunctions.comparators

Class LCLongestCommonSubsequence

```
java.lang.Object
    |
    +--AbstractListComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.LCLongestCommonSubs
```

< Constructors > < Methods >

---

public class **LCLongestCommonSubsequence**
extends AbstractListComparator

This comparator compares two aligned list based on an LongestCommonSubsequence-algorithm. The condition whether a sequence is discontinued or not is made based on the inner comparator and a threshold. If the similarity-value calculated by the inner comparator is below the threshold, the sequence is deemed as interrupted. The parameter of this comparefunction specifies at position

- 0: The inner comparator and it's configuration
- 1: The threshold for LCS-calculation

The following example-configuration shows how to use the LCLongestCommonSubsequence comparator in combination with ECMatched. The ordered children of the dedicated class are compared based on whether they are already matched or not.

<CompareFunction class="ChildrenCF" weight="1"
parameter="LCLongestCommonSubsequence[ECMatched;1f]"/>

org.sidiff.core.compare.comparefunctions.comparators.ECMatched

Author:

Pit Pietsch

Constructors

LCLongestCommonSubsequence

```
public  LCLongestCommonSubsequence(EClass dedicatedClass,
                                   java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        java.util.List sequenceA,
                        java.util.List sequenceB)
```

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class SCGreedyMatchedOrSimilar

```
java.lang.Object
    |
    +--AbstractSetComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.SCGreedyMatchedOrSi
```

< Constructors > < Methods >

---

public class **SCGreedyMatchedOrSimilar**
extends AbstractSetComparator

This comparator calculates a similarity value between two sets of elements based on matchings and similarities. The greedy algorithm used is parted in 3 steps:

(1) In a first step the algorithm looks for matches. Every match is deemed as a similarity of 1f and added to a temporary similarity value.

(2) In a second step a greedy algorithm tries to match the rest of the elements as best as possible based on their similarity. The similarity-value of every found element-pair is added to the temporary similarity.

(3) In a last step the temporary similarity is normalized by the size of the bigger element-set and the calculated similarity is returned.

This comparator does not need a parameter.

Author:

wenzel

Constructors

SCGreedyMatchedOrSimilar

```
public  SCGreedyMatchedOrSimilar(EClass dedicatedClass,
                                 java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.util.Collection collectionA,
                     java.util.Collection collectionB)
```

The calculation of the similarity value is separated in three parts.

(1) In a first step the algorithm looks for matches. Every match is deemed as a similarity of 1f and added to a temporary similarity value.

(2) In a second step a greedy algorithm tries to match the rest of the elements as best as possible based on their similarity. The similarity-value of every found element-pair is added to the temporary similarity.

(3) In a last step the temporary similarity is normalized by the size of the bigger element-set and the calculated similarity is returned.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class SCGreedySimilarity

```
java.lang.Object
    |
    +--AbstractSetComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.SCGreedySimilarity
```

< Constructors > < Methods >

---

public class **SCGreedySimilarity**

extends AbstractSetComparator

The calculation of the similarity value is separated in two parts:

(1) In a first step the algorithm looks tries to match the elements as best as possible based on their similarity. The similarity-value of every found element-pair is added to the temporary similarity.

(2) In a second step the temporary similarity is normalized by the size of the bigger element-set and the calculated similarity is returned.

This comparator does not need a parameter.

org.sidiff.core.compare.comparefunctions.abstractcomparators.AbstractSetComparator#compare(java.uti
java.util.Collection)

Author:

Pit Pietsch

Constructors

SCGreedySimilarity

```
public  SCGreedySimilarity(EClass dedicatedClass,
                           java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.util.Collection collectionA,
                     java.util.Collection collectionB)
```

The calculation of the similarity value is separated in two parts:

(1) In a first step the algorithm looks tries to match the elements as best as possible based on their similarity. The similarity-value of every found element-pair is added to the temporary similarity.

(2) In a second step the temporary similarity is normalized by the size of the bigger element-set and the calculated similarity is returned.

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class SCMatched

```
java.lang.Object
    |
    +--AbstractSetComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.SCMatched
```

< Constructors > < Methods >

---

public class **SCMatched**
extends AbstractSetComparator

This comparator counts the number of matched elements between to sets and normalize the sum by dividing through the number of elements in the bigger set.

It is assured by assertion that this comparator can only be used for 1:1 matchings.

This comparator does not need a parameter.

Author:

Pit Pietsch

Constructors

SCMatched

```
public  SCMatched(EClass dedicatedClass,
                  java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.util.Collection collectionA,
                     java.util.Collection collectionB)
```

The number of matched elements between to sets are counted and the sum is normalized by dividing through the number of elements in the bigger set.

It is assured by assertion that this comparator can only be used for 1:1 matchings.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class SCSize

```
java.lang.Object
    |
    +--AbstractSetComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.SCSize
```

< Constructors > < Methods >

---

public class **SCSize**
extends AbstractSetComparator

This comparator compares two collections based on their .size()-method. The returned similarity is

- 1f in case the given collections have the same size or
- 0f in case their size differ.

No additional parameters are necessary to use this comparator.

The following example-configuration shows how to use the SCSize comparator to compare two (Method-)elements based on their number of operations.

<CompareFunction class="NeighborsSingleReferenceCF" weight="1" parameter="SCSize;operations"/>

Author:
wenzel

Constructors

SCSize

```
public  SCSize(EClass dedicatedClass,
               java.lang.String parameter)
```

Methods

compare

```
protected float compare(EObject contextElementA,
                        EObject contextElementB,
                        java.util.Collection collectionA,
                        java.util.Collection collectionB)
```

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class VCCharacterEqualsCI

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.VCCharacterEqualsCI
```

< Constructors > < Methods >

---

public class **VCCharacterEqualsCI**
extends AbstractValueComparator

This comparator performs an case-insensitive .equals()-comparison between two Character. In case they are equals 1f is returned, 0f otherwise. The CEquals comparator should be used for case-sensitive equal-comparison instead. The comparator needs no additional parameter. It is assured by assertion that object to be compared are of the type strings. The following example-configuration shows how to use the VCStringEqualsCI comparator to compare the name-attributes from elements of type Operation

<CompareFunction class="SelfCF" weight="1"
parameter="ECAttributeStatic[VCCharakterEqualsCI;Operation;name]"/>

CEquals

Author:

Pit Pietsch

Constructors

VCCharacterEqualsCI

```
public  VCCharacterEqualsCI(EClass dedicatedClass,
                            java.lang.String parameter)
```

Parameters:

dedicatedClass -
parameter -

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class VCGauss

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.VCGauss
```

< Constructors > < Methods >

---

public class **VCGauss**
extends AbstractValueComparator

This comparator compares two numerical values based on a gauss-calculation. It is assured by assertion that the values are of a numerical type. The parameter of this comparefunction specifies at position

- 0: The scale for the gauss-calculation

The following example-configuration shows how to use the VCGauss comparator to compare two (numerical) attributes from elements of type ElementB.

<CompareFunction class="SelfCF" weight="1"
parameter="ECAttributeStatic[VCGauss[0.7f];ElementB;numberX]"/>

java.lang.Number

Author:

Pit Pietsch

Constructors

VCGauss

```
public  VCGauss(EClass dedicatedClass,
                java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

Calculates the similarity of two numerical values based on a gauss-calculation.

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class VCStringEMFCompare

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.VCStringEMFCompare
```

< Constructors > < Methods >

public class **VCStringEMFCompare**
extends AbstractValueComparator

The suggested string similarity measure by Xing/Stroulia This code is blatantly stolen from org.eclipse.emf.compare.match.internal.statistic.NameSimilarity

Constructors

VCStringEMFCompare

```
public  VCStringEMFCompare(EClass dedicatedClass,
                           java.lang.String parameter)
```

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

init

```
public void init(ServiceContext serviceContext)
```

org.sidiff.core.compare.comparefunctions.comparators

Class VCStringEqualsCI

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.VCStringEqualsCI
```

< Constructors > < Methods >

public class **VCStringEqualsCI**
extends AbstractValueComparator

This comparator performs an case-insensitive .equals()-comparison between two Strings. In case they are

equals 1f is returned, 0f otherwise. The CEquals comparator should be used for case-sensitive equal-comparison instead. The comparator needs no additional parameter. It is assured by assertion that object to be compared are of the type strings. The following example-configuration shows how to use the VCStringEqualsCI comparator to compare the name-attributes from elements of type Operation

<CompareFunction class="SelfCF" weight="1"
parameter="ECAttributeStatic[VCStringEqualsCI;Operation;name]"/>

CEquals

Author:

Pit Pietsch

Constructors

VCStringEqualsCI

```
public  VCStringEqualsCI(EClass dedicatedClass,
                         java.lang.String parameter)
```

Parameters:

dedicatedClass -
parameter -

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

---

init

```
public void init(ServiceContext serviceContext)
```

org.sidiff.core.compare.comparefunctions.comparators

Class VCStringIndexOf

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.VCStringIndexOf
```

< Constructors > < Methods >

---

public class **VCStringIndexOf**
extends AbstractValueComparator

This comparator compares two string values based on a their index-of similarity. It is assured by assertion that the two values are of the type string. The parameter of this comparefunction specifies at position

- 0: Indication whether the strings are to be compared case-insensitive (ci) or not (cs)

The following example-configuration shows how to use the VCStringLCS comparator to compare the name-attributes from elements of type Operation with an case-insensitive index-of similarity calculation.

<CompareFunction class="SelfCF" weight="1"
parameter="ECAttributeStatic[VCStringIndexOf[ci];Operation;name]"/>

Author:

Pit Pietsch

Constructors

VCStringIndexOf

```
public  VCStringIndexOf(EClass dedicatedClass,
                        java.lang.String parameter)
```

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object elementA,
                     java.lang.Object elementB)
```

---

init

```
public void init(ServiceContext serviceContext)
```

---

org.sidiff.core.compare.comparefunctions.comparators

Class VCStringLCS

```
java.lang.Object
    |
    +--AbstractValueComparator
        |
        +--org.sidiff.core.compare.comparefunctions.comparators.VCStringLCS
```

< Constructors > < Methods >

---

public class **VCStringLCS**
extends AbstractValueComparator

This comparator compares two string values based on a longest common subsequence calculation. It is assured by assertion that the two values are of the type string. The parameter of this comparefunction specifies at position

- 0: Indication whether the strings are to be compared case-insensitive (ci) or not (cs)

The following example-configuration shows how to use the VCStringLCS comparator to compare the name-attributes from elements of type Operation with an case-sensitive longest common subsequence calculation.

<CompareFunction class="SelfCF" weight="1"
parameter="ECAttributeStatic[VCStringLCS[cs];Operation;name]"/>

Author:

Pit Pietsch

Constructors

VCStringLCS

```
public  VCStringLCS(EClass dedicatedClass,
                    java.lang.String parameter)
```

Constructor.

Parameters:

dedicatedClass - the Class this comparator is used on
parameter - the parameter for this comparator

Methods

compare

```
public float compare(EObject contextElementA,
                     EObject contextElementB,
                     java.lang.Object valueA,
                     java.lang.Object valueB)
```

Calculates the similarity of two string values based on the longest common subsequence. It is assured by assertion that the two values are of the strings.

---

init

```
public void init(ServiceContext serviceContext)
```