Notio API Documentation

Finnegan Southey

July 6, 2001

Contents

Chapter 1

Package notio.translators

Package Contents *Page*

1.1 Interfaces

1.1.1 Interface CGIFParserImpl.ScannerConstants

A switch for use in debugging the parser.

Declaration

```
public static interface CGIFParserImpl.ScannerConstants
```

Fields

- public static final int EOF
  –
- public static final int IF
  –
- public static final int THEN
  –
- public static final int ELSE
  –
- public static final int EITHER
  –
- public static final int OR
  –
- public static final int SCOPE
  –
- public static final int LAMBDA
  –
- public static final int LEFT_PAREN
  –
- public static final int RIGHT_PAREN
  –
- public static final int LEFT_BRACKET
  –
- public static final int RIGHT_BRACKET

–

- public static final int LEFT_BRACE

  –

- public static final int RIGHT_BRACE

  –

- public static final int LEFT_ANGLE

  –

- public static final int RIGHT_ANGLE

  –

- public static final int COMMA

  –

- public static final int PERIOD

  –

- public static final int SEMICOLON

  –

- public static final int COLON

  –

- public static final int QUESTIONMARK

  –

- public static final int TILDE

  –

- public static final int AT

  –

- public static final int HASH

  –

- public static final int PERCENT

  –

- public static final int ASTERISK

  –

- public static final int VERT_BAR

  –

- public static final int AND

  –

- public static final int COMMENT

–

- public static final int SIGNED_INTEGER_LITERAL

  –

- public static final int UNSIGNED_INTEGER_LITERAL

  –

- public static final int SIGNED_FLOATING_POINT_LITERAL

  –

- public static final int UNSIGNED_FLOATING_POINT_LITERAL

  –

- public static final int EXPONENT

  –

- public static final int STRING_LITERAL

  –

- public static final int NAME_LITERAL

  –

- public static final int IDENTIFIER

  –

- public static final int LETTER

  –

- public static final int DIGIT

  –

- public static final int _jjVif

  –

- public static final int _jjVthen

  –

- public static final int _jjVelse

  –

- public static final int _jjVeither

  –

- public static final int _jjVor

  –

- public static final int _jjVsc

  –

- public static final int _jjVlambda

–

- public static final int _jjV_iB
  –
- public static final int _jjV_jB
  –
- public static final int _jjV_1C
  –
- public static final int _jjV_3C
  –
- public static final int _jjV_1D
  –
- public static final int _jjV_3D
  –
- public static final int _jjV_2B
  –
- public static final int _jjV_4B
  –
- public static final int _jjV_mB
  –
- public static final int _jjV_oB
  –
- public static final int _jjV_1B
  –
- public static final int _jjV_0B
  –
- public static final int _jjV_5B
  –
- public static final int _jjV_4D
  –
- public static final int _jjV_aC
  –
- public static final int _jjV_dB
  –
- public static final int _jjV_fB

–

- public static final int _jjV_kB

  –

- public static final int _jjV_2D

  –

- public static final int _jjV_gB

  –

- public static final int DEFAULT

  –

- public static final String tokenImage

  –

1.1.2 Interface **LFParserImpl.ScannerConstants**

Name of DefiningLabelTable unit in context.

Declaration

public static interface LFParserImpl.ScannerConstants

Fields

- public static final int EOF

  –

- public static final int NEWLINE

  –

- public static final int LAMBDA

  –

- public static final int SCOPE

  –

- public static final int TYPEMACRO

  –

- public static final int LEFT_PAREN

  –

- public static final int RIGHT_PAREN

–

- public static final int LEFT_BRACKET

  –

- public static final int RIGHT_BRACKET

  –

- public static final int LEFT_BRACE

  –

- public static final int RIGHT_BRACE

  –

- public static final int LEFT_ARROW

  –

- public static final int RIGHT_ARROW

  –

- public static final int LEFT_ANGLE

  –

- public static final int RIGHT_ANGLE

  –

- public static final int COMMA

  –

- public static final int PERIOD

  –

- public static final int COLON

  –

- public static final int SEMICOLON

  –

- public static final int QUESTIONMARK

  –

- public static final int HYPHEN

  –

- public static final int TILDE

  –

- public static final int AT

  –

- public static final int HASH

–

- public static final int ASTERISK

  –

- public static final int VERT_BAR

  –

- public static final int CONCEPT_COMMENT

  –

- public static final int RELATION_COMMENT

  –

- public static final int ACTOR_COMMENT

  –

- public static final int FORMAL_COMMENT

  –

- public static final int MULTI_LINE_COMMENT

  –

- public static final int INTEGER_LITERAL

  –

- public static final int DECIMAL_LITERAL

  –

- public static final int HEX_LITERAL

  –

- public static final int OCTAL_LITERAL

  –

- public static final int FLOATING_POINT_LITERAL

  –

- public static final int EXPONENT

  –

- public static final int CHARACTER_LITERAL

  –

- public static final int STRING_LITERAL

  –

- public static final int NAME_LITERAL

  –

- public static final int IDENTIFIER

–

- public static final int LETTER

  –

- public static final int DIGIT

  –

- public static final int _jjV_K

  –

- public static final int _jjV_iBlambda

  –

- public static final int _jjV_1CScope

  –

- public static final int _jjV_iBtypeMacro

  –

- public static final int _jjV_iB

  –

- public static final int _jjV_jB

  –

- public static final int _jjV_1C

  –

- public static final int _jjV_3C

  –

- public static final int _jjV_1D

  –

- public static final int _jjV_3D

  –

- public static final int _jjV_2B_nB

  –

- public static final int _jjV_nB_4B

  –

- public static final int _jjV_2B

  –

- public static final int _jjV_4B

  –

- public static final int _jjV_mB

–

- public static final int _jjV_oB

  –

- public static final int _jjV_0B

  –

- public static final int _jjV_1B

  –

- public static final int _jjV_5B

  –

- public static final int _jjV_nB

  –

- public static final int _jjV_4D

  –

- public static final int _jjV_aC

  –

- public static final int _jjV_dB

  –

- public static final int _jjV_kB

  –

- public static final int _jjV_2D

  –

- public static final int DEFAULT

  –

- public static final int IN_FORMAL_COMMENT

  –

- public static final int IN_MULTI_LINE_COMMENT

  –

- public static final String tokenImage

  –

1.2 Classes

1.2.1 Class **CGIFGenerator**

A CGIF Generator class.

Declaration

public class CGIFGenerator
extends notio.translators.SimpleGenerator
implements notio.Generator

Constructors

- *CGIFGenerator*
  `public` **CGIFGenerator( )**

Methods

- *generateActor*
  `public void` **generateActor(** `notio.Actor` **actor)**
  - **Usage**
    - Generates an actor to the output stream.
  - **Parameters**
    - `relation` - the relation to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.

- *generateCGStream*
  `public void` **generateCGStream(** `java.util.Vector` **graphs)**
  - **Usage**
    - Generates a vector of graphs to the output stream.
  - **Parameters**
    - `graphs` - the vector of graphs to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.

- *generateConcept*
  `public void` **generateConcept(** `notio.Concept` **concept)**
  - **Usage**
    - Generates a concept to the output stream.
  - **Parameters**
    - `concept` - the concept to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.

- *generateGraph*
  `public void` **generateGraph(** `notio.Graph` **graph)**
  - **Usage**

    * Generates a graph to the output stream.
  - **Parameters**
    * `graph` - the graph to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *generateOutermostContext*
  `public void` **generateOutermostContext**`( notio.Graph` **graph** `)`
  - **Usage**
    * Generates a graph to the output stream.
  - **Parameters**
    * `graph` - the graph to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *generateRelation*
  `public void` **generateRelation**`( notio.Relation` **relation** `)`
  - **Usage**
    * Generates a relation to the output stream.
  - **Parameters**
    * `relation` - the relation to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *getAllowIncompleteRelations*
  `public boolean` **getAllowIncompleteRelations**`( )`
  - **Usage**
    * Returns a flag indicating whether incomplete relations are allowed whilst generating. If this flag is false, a GeneratorException will be thrown when an incomplete relation is encountered. If it is true, any incomplete arguments will simply be skipped as if they did not exist. If incomplete relations are allowed there is no guarantee that any CGIF parser will accept the generate output if it enforces relation valence or type definitions. The default setting for this flag is false.
  - **Returns** - the value for the flag.
  - **See Also**
    * `notio.Relation.isComplete` (in 2.2.37, page 157)

- *getEmbedSingleRelatorConcepts*
  `public boolean` **getEmbedSingleRelatorConcepts**`( )`
  - **Usage**
    * Returns a flag indicating whether this generator will embed single relator concepts within the single relation. The default setting for this flag is false.
  - **Returns** - the value for the flag.

- *getIndentNestedGraphs*
  `public boolean` **getIndentNestedGraphs**`( )`

  - **Usage**
    * Returns a flag indicating whether this generator should generate indentation for nested graphs. The default setting for this flag is false.
  - **Returns** - the value for the flag.

- *getInitiateGraphsWithNewline*
`public boolean` **getInitiateGraphsWithNewline**`( )`

  - **Usage**
    * Returns a flag indicating whether this generator should generate a newline at the beginning of every graph. The default setting for this flag is false.
  - **Returns** - the value for the flag.

- *getSupressGraphComments*
`public boolean` **getSupressGraphComments**`( )`

  - **Usage**
    * Returns a flag indicating whether this generator will supress comments in graphs. The default setting for this flag is false.
  - **Returns** - the value for the flag.

- *getSupressNodeComments*
`public boolean` **getSupressNodeComments**`( )`

  - **Usage**
    * Returns a flag indicating whether this generator will supress comments in nodes (Concepts, Relations, and Actors). The default setting for this flag is false.
  - **Returns** - the value for the flag.

- *getTerminateGraphsWithNewline*
`public boolean` **getTerminateGraphsWithNewline**`( )`

  - **Usage**
    * Returns a flag indicating whether this generator should generate a newline at the end of every graph. The default setting for this flag is false.
  - **Returns** - the value for the flag.

- *initializeGenerator*
`public void` **initializeGenerator**`( java.io.Writer` **newWriter**`, notio.KnowledgeBase` **newKnowledgeBase**`, notio.TranslationContext` **newTranslationContext** `)`

  - **Usage**
    * Initializes the generator to write to the specified writer using the specified TranslationContext and KnowledgeBase.
  - **Parameters**
    * `newWriter` - the writer to be generated to.
    * `newKnowledgeBase` - the knowledge base to be used while generating.
    * `newTranslationContext` - the translationContext to be used while generating.

- *setAllowIncompleteRelations*
  `public void` **setAllowIncompleteRelations**`( boolean` **flag** `)`
  - **Usage**
    - Sets a flag indicating whether incomplete relations are allowed whilst generating. If this flag is false, a GeneratorException will be thrown when an incomplete relation is encountered. If it is true, any incomplete arguments will simply be skipped as if they did not exist. If incomplete relations are allowed there is no guarantee that any CGIF parser will accept the generate output if it enforces relation valence or type definitions. The default setting for this flag is false.
  - **Parameters**
    - `flag` - the new value for the flag.
  - **See Also**
    - `notio.Relation.isComplete` (in 2.2.37, page 157)

- *setEmbedSingleRelatorConcepts*
  `public void` **setEmbedSingleRelatorConcepts**`( boolean` **flag** `)`
  - **Usage**
    - Sets a flag indicating whether this generator should generate concepts that are not coreferent to other concepts and only related to a single relation into the relation itself. E.g. [Dog *x] (Sleeps ?x) would be rendered as (Sleeps [Dog]). The default setting for this flag is false.
  - **Parameters**
    - `flag` - the new value for the flag.

- *setIndentNestedGraphs*
  `public void` **setIndentNestedGraphs**`( boolean` **flag** `)`
  - **Usage**
    - Sets a flag indicating whether this generator should generate indentation before each graph, increasing the indentation for every level of nesting. Note that this will probably not look right unless the generator is set to "initiate graphs with newline". The indent will be applied after any newline, intiating or terminating. The default setting for this flag is false.
  - **Parameters**
    - `flag` - the new value for the flag.

- *setInitiateGraphsWithNewline*
  `public void` **setInitiateGraphsWithNewline**`( boolean` **flag** `)`
  - **Usage**
    - Sets a flag indicating whether this generator should generate a newline at the beginning of every graph. The default setting for this flag is false.
  - **Parameters**
    - `flag` - the new value for the flag.

- *setSupressGraphComments*
  `public void` **setSupressGraphComments**`( boolean` **flag** `)`
  - **Usage**

    * Sets a flag indicating whether this generator should supress comments in graphs. The default setting for this flag is false.
  - **Parameters**
    * `flag` - the new value for the flag.

- *setSupressNodeComments*
  `public void` **setSupressNodeComments**`( boolean` **flag** `)`
  - **Usage**
    * Sets a flag indicating whether this generator should supress comments in nodes (Concepts, Relations, and Actors). The default setting for this flag is false.
  - **Parameters**
    * `flag` - the new value for the flag.

- *setTerminateGraphsWithNewline*
  `public void` **setTerminateGraphsWithNewline**`( boolean` **flag** `)`
  - **Usage**
    * Sets a flag indicating whether this generator should generate a newline at the end of every graph. The default setting for this flag is false.
  - **Parameters**
    * `flag` - the new value for the flag.

METHODS INHERITED FROM CLASS `notio.translators.SimpleGenerator`

- *escapeCharactersInString*
  `public static String` **escapeCharactersInString**`( java.lang.String` **in**`, java.lang.String` **chars**`, char` **escapeSequence** `)`
  - **Usage**
    * Adds escape sequences for specified characters whenever they occur within the specified string.
  - **Parameters**
    * `in` - the string to be modified.
    * `chars` - a string containing all characters that need to be escaped.
    * `escapeSequence` - the character that should prefix characters that need escaping (e.g. backslash).
  - **Returns** - the modified string which now includes escape sequences.

- *generateUnit*
  `public void` **generateUnit**`( java.lang.Object` **unit** `)`
  - **Usage**
    * Generates a graph to the output stream.
  - **Parameters**
    * `unit` - the unit object to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *getUnitClass*
  `public Class` **getUnitClass**`( )`
  - **Usage**

    * Returns a Class object that indicates what class the Unit parse method will return.
  - **Returns** - a Class object that indicates what class the Unit parse method will return.
  - **See Also**
    * `notio.Generator.generateUnit`

- *initializeGenerator*
  `public void` **initializeGenerator**`( java.io.Writer` **newWriter**`, notio.KnowledgeBase` **newKnowledgeBase**`, notio.TranslationContext` **newTranslationContext** `)`
  - **Usage**
    * Initializes the generator to write to the specified writer using the specified TranslationContext and KnowledgeBase.
  - **Parameters**
    * `newWriter` - the writer to be generated to.
    * `newKnowledgeBase` - the knowledge base to be used while generating.
    * `newTranslationContext` - the translationContext to be used while generating.

1.2.2 Class CGIFParser

A CGIF Parser class.

Declaration

public class CGIFParser
extends java.lang.Object
implements notio.Parser

Constructors

- *CGIFParser*
  `public` **CGIFParser**`( )`

Methods

- *getCreateTypesOnDemand*
  `public boolean` **getCreateTypesOnDemand**`( )`
  - **Usage**
    * This method may be used to check whether the parser will create type objects for any parsed type labels not currently in the database, or throw an exception. This flag is set using setCreateTypesOnDemand().
  - **Returns** - true if types will created on demand or false if an exception will be thrown.

- *getUnitClass*
  `public Class` **getUnitClass**`( )`
  - **Usage**
    * Returns the unit object parsed from the input stream.

  - **Returns** - the unit object parsed from the input stream.

- *initializeParser*
  `public void` **initializeParser(** `java.io.Reader` **reader,** `notio.KnowledgeBase` **kBase,** `notio.TranslationContext` **tContext)**
  - **Usage**
    - Initializes the parser to parse the specified reader.
  - **Parameters**
    - `reader` - the reader whose contents are to be parsed.
    - `kBase` - the knowledge base to be used while parsing.
    - `tContext` - the translation context to be used while parsing.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while initializing the parser.

- *parseActor*
  `public Actor` **parseActor( )**
  - **Usage**
    - Attempts to parse an actor from the input stream.
  - **Returns** - the actor parsed from the input stream.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while parsing.
    - `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseCGStream*
  `public Vector` **parseCGStream( )**
  - **Usage**
    - Attempts to parse a CGStream from the input stream.
  - **Returns** - a vector of graphs parsed from the input stream.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while parsing.

- *parseConcept*
  `public Concept` **parseConcept( )**
  - **Usage**
    - Attempts to parse a concept from the input stream.
  - **Returns** - the concept parsed from the input stream.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while parsing.

- *parseGraph*
  `public Graph` **parseGraph( )**
  - **Usage**
    - Attempts to parse a graph from the input stream.
  - **Returns** - the graph parsed from the input stream.
  - **Exceptions**

    * `notio.ParserException` - if an error occurs while parsing.

- *parseOutermostContext*
  `public Graph` **parseOutermostContext( )**
  - **Usage**
    * Attempts to parse a graph which is treated as the outermost context for purposes for scoping. This method should be used instead of parseGraph() when no translation information is to be used from previous translation sessions and when the parser can safely assume that the graph is "self-contained".
  - **Returns** - the graph parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.
    * `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseRelation*
  `public Relation` **parseRelation( )**
  - **Usage**
    * Attempts to parse a relation from the input stream.
  - **Returns** - the relation parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *parseUnit*
  `public Object` **parseUnit( )**
  - **Usage**
    * Attempts to parse the default unit from the input stream. The default unit is whatever a particular parser is usually intended to parse.
  - **Returns** - the unit object parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *setCreateTypesOnDemand*
  `public void` **setCreateTypesOnDemand(** `boolean` **flag)**
  - **Usage**
    * This method may be used to tell the parser whether it should create type objects for any parsed type labels not currently in the database, or throw an exception. Created types have only the universal and absurd types as parents and children respectively. The flag is true by default.
  - **Parameters**
    * `flag` - true or false to turn automatic creation on or off.

1.2.3 Class **DefiningLabelTable**

A TranslationInfoUnit that serves as a symbol table relating defining labels to defining concepts and coreference sets. This table allows stack-like operations for maintaining label scope.

Declaration

public class DefiningLabelTable
extends notio.translators.SimpleInfoUnit

Serializable Fields

- private LabelFactory defLabelFactory
  - A factory for producing defining labels.
- private DefiningLabelTable.ContextTable baseTable
  - The base table of the ContextTable stack.
- private DefiningLabelTable.ContextTable topTable
  - The top table of the ContextTable stack.

Constructors

- *DefiningLabelTable*
  `public` **DefiningLabelTable**`( )`
  - **Usage**
    - Constructs a new DefiningLabelTable.

- *DefiningLabelTable*
  `public` **DefiningLabelTable**`( notio.translators.DefiningLabelTable` **orgTable** `)`
  - **Usage**
    - Constructs a new DefiningLabelTable by copying the specified table.
  - **Parameters**
    - `orgTable` - the table to be copied.

Methods

- *addBoundConcept*
  `public final void` **addBoundConcept**`( java.lang.String` **newLabel**`,`
  `notio.Concept` **newBoundConcept** `)`
  - **Usage**
    - Adds an entry to the table that relates the specified label to the specified bound concept.
  - **Parameters**
    - `newLabel` - the label to be be added.
    - `newBoundConcept` - the bound concept to be added.

- *addCoreferenceSet*
  `public void` **addCoreferenceSet(** `java.lang.String` **definingLabel,** `notio.CoreferenceSet` **corefSet)**
  - **Usage**
    - Adds an entry to the table that relates the specified label to the specified coreference set. If the coreference set is already associated with the label, nothing will done. If the coreference set has an associated defining concept, that concept will also be included in the entry.
  - **Parameters**
    - `definingLabel` - the defining label being mapped.
    - `corefSet` - the coreference set being mapped.
  - **Exceptions**
    - `java.lang.IllegalArgumentException` - if the label is already associated with some other coreference set, or if it is associated with a defining concept other than the one in the coreference set.

- *addDefiningConcept*
  `public final void` **addDefiningConcept(** `java.lang.String` **newLabel,** `notio.Concept` **newDefConcept)**
  - **Usage**
    - Adds an entry to the table that relates the specified label to the specified defining concept. If the defining concept has an associated coreference set, that set will also be included in the entry.
  - **Parameters**
    - `newLabel` - the label to be be added.
    - `newDefConcept` - the defining concept to be added.

- *clearDefiningLabelToCoreferenceSetMapping*
  `public void` **clearDefiningLabelToCoreferenceSetMapping( )**
  - **Usage**
    - Clears all existing mappings between coreference labels and coreference sets.

- *clearDefiningLabelToDefiningConceptMapping*
  `public void` **clearDefiningLabelToDefiningConceptMapping( )**
  - **Usage**
    - Clears all existing mappings between defining concepts and defining labels.

- *copyUnit*
  `public TranslationInfoUnit` **copyUnit( )**
  - **Usage**
    - Returns a duplicate of this information unit that is distinct from the original. This means that changes to the original will not affect the duplicate and vice versa.
  - **Returns** - a duplicate of this unit.

- *findDefiningEntryByLabel*
  `public final DefiningLabelTable.TableEntry` **findDefiningEntryByLabel(** `java.lang.String` **label)**

  - **Usage**
    * Finds the first (topmost) defining entry corresponding to the specified label.
  - **Parameters**
    * `label` - the label to be found.
  - **Returns** - the entry, or null.

- *findEntryByConcept*
  `public final DefiningLabelTable.TableEntry` **findEntryByConcept(**
  `notio.Concept` **concept)**

  - **Usage**
    * Finds the entry corresponding to the specified Concept.
  - **Parameters**
    * `concept` - the concept to be found.
  - **Returns** - the entry, or null.

- *findEntryByCoreferenceSet*
  `public final DefiningLabelTable.TableEntry` **findEntryByCoreferenceSet(**
  `notio.CoreferenceSet` **corefSet)**

  - **Usage**
    * Finds the entry corresponding to the specified CoreferenceSet.
  - **Parameters**
    * `corefSet` - the coreference set to be found.
  - **Returns** - the entry, or null.

- *findEntryByDefiningConcept*
  `public final DefiningLabelTable.TableEntry` **findEntryByDefiningConcept(**
  `notio.Concept` **defConcept)**

  - **Usage**
    * Finds the entry corresponding to the specified defining Concept. This returns null if the concept is not in the table or is not a defining concept.
  - **Parameters**
    * `defConcept` - the defining concept to be found.
  - **Returns** - the entry, or null.

- *findFirstEntryByLabel*
  `public final DefiningLabelTable.TableEntry` **findFirstEntryByLabel(**
  `java.lang.String` **label)**

  - **Usage**
    * Finds the first (topmost) entry corresponding to the specified label.
  - **Parameters**
    * `label` - the label to be found.
  - **Returns** - the entry, or null.

- *findLocalEntryByLabel*
  `public final DefiningLabelTable.TableEntry` **findLocalEntryByLabel(**
  `java.lang.String` **label)**

  - **Usage**
    * Finds an entry in the current context corresponding to the specified label.
  - **Parameters**
    * `label` - the label to be found.
  - **Returns** - the entry, or null.

- *getCurrentContextLevel*
`public int` **getCurrentContextLevel( )**

  - **Usage**
    * Returns the current context level of this table. This is the number of contexts sitting on the stack minus one. It is never less than 0.
  - **Returns** - the current context level of this table.

- *getLastDefiningLabel*
`public String` **getLastDefiningLabel( )**

  - **Usage**
    * Returns the last defining label generated by this context or null if no labels have generated. Note that this method does not generated any new labels.
  - **Returns** - the last defining label generated by this context or null.

- *getNextAvailableDefiningLabel*
`public String` **getNextAvailableDefiningLabel( )**

  - **Usage**
    * Returns a new defining label that is not already present in the defining label/coreference set mapping. There are no constraints on the form of the label other than that they conform to a valid CGIF identifier. Otherwise, labels' forms are determined by the underlying implementation.
  - **Returns** - a new, unique, and unused defining label.

- *popContext*
`public final void` **popContext( )**

  - **Usage**
    * Pops a context off the stack. Any entries added to this table since the corresponding the push will be removed.

- *pushContext*
`public final void` **pushContext( )**

  - **Usage**
    * Pushes a new context onto the stack. Any entries added to this table after the push will be removed when a corresponding pop is called.

- *resetAvailableDefiningLabel*
`public void` **resetAvailableDefiningLabel( )**

  - **Usage**

* Resets the generator for defining labels so that it starts again. This method would typically be called after clearing the defining label - coreference set mapping so that labels may be reused in a new translation, if possible. Implementations are not required to honour this although they should silently do nothing rather than throwing an exception. If possible, implementations should reset their label generators so that they produce a sequence of labels as if this were a newly constructed context.

- *resetUnit*
  `public void` **resetUnit( )**
  - **Usage**
    * Resets this information unit to its initial state. This method can be called by translators in order to ensure that a unit contains no information relating to earlier translation sessions. This causes all contexts to be emptied and popped and fresh initial context pushed.

METHODS INHERITED FROM CLASS `notio.translators.SimpleInfoUnit`

(in 1.2.8, page 36)

- *getUnitName*
  `public String` **getUnitName( )**
  - **Usage**
    * Returns the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Returns** - the name for this unit.

- *setUnitName*
  `public void` **setUnitName(** `java.lang.String` **newName)**
  - **Usage**
    * Sets the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Parameters**
    * `newName` - the new name for this unit.

1.2.4 CLASS DefiningLabelTable.TableEntry

An inner class used to hold table entries.

DECLARATION

public final class DefiningLabelTable.TableEntry
extends java.lang.Object

METHODS

- *getContextLevel*
  `public final int` **getContextLevel( )**
  - **Usage**
    - Returns the context level to which this entry belongs.
  - **Returns** - the context level to which this entry belongs.

1.2.5 CLASS LFGenerator

A LF Generator class.

DECLARATION

public class LFGenerator **extends** notio.translators.SimpleGenerator **implements** notio.Generator

CONSTRUCTORS

- *LFGenerator*
  `public` **LFGenerator( )**

METHODS

- *Actor*
  `public void` **Actor(** `notio.Actor` **actor)**
  - **Usage**
    - Generates an actor to the output stream.
  - **Parameters**
    - `actor` - the actor to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.
- *ActorComment*
  `public void` **ActorComment(** `java.lang.String` **comment)**
  - **Usage**
    - Generates a actor comment to output stream.
  - **Parameters**
    - `comment` - the actor comment to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.

- *ConceptComment*
  `public void` **ConceptComment(** `java.lang.String` **comment)**
  - **Usage**
    - Generates a concept comment to output stream.
  - **Parameters**
    - `comment` - the concept comment to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.

- *ConceptType*
  `public void` **ConceptType(** `notio.ConceptType` **conType)**
  - **Usage**
    - Generates a concept type to the output stream.
  - **Parameters**
    - `conType` - the concept type to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an IO error occurs.

- *decreaseIndent*
  `public void` **decreaseIndent(** **)**
  - **Usage**
    - Decreases the current indent level.

- *Designator*
  `public void` **Designator(** `notio.Designator` **designator)**
  - **Usage**
    - Generates a designator to the output stream.
  - **Parameters**
    - `designator` - the designator to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.

- *generateActor*
  `public void` **generateActor(** `notio.Actor` **actor)**
  - **Usage**
    - Generates the specified actor.
  - **Parameters**
    - `actor` - the actor to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.
    - `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateCGStream*
  `public void` **generateCGStream(** `java.util.Vector` **graphs)**

  - **Usage**
    * LFGenerator does not support the CGStream construct. Calling this method will cause an UnimplementedFeatureException to be thrown.
  - **Parameters**
    * `graphsa` - the vector of graphs to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.
    * `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateConcept*
  `public void` **generateConcept**`( notio.Concept` **concept** `)`

  - **Usage**
    * Generates a concept to the output stream.
  - **Parameters**
    * `concept` - the concept to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *generateGraph*
  `public void` **generateGraph**`( notio.Graph` **graph** `)`

  - **Usage**
    * Generates a graph to the output stream.
  - **Parameters**
    * `graph` - the graph to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *generateOutermostContext*
  `public void` **generateOutermostContext**`( notio.Graph` **graph** `)`

  - **Usage**
    * Generates the specified graph which is treated as the outermost context for purposes for scoping. This method should be used instead of generateGraph() when no translation information is to be used from previous translation sessions and when the generator can safely assume that the graph is "self-contained".
  - **Parameters**
    * `graph` - the graph to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.
    * `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateRelation*
  `public void` **generateRelation**`( notio.Relation` **relation** `)`

  - **Usage**
    * Generates a relation to the output stream.

  - **Parameters**
    * `relation` - the relation to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *GraphComment*
  `public void` **GraphComment**`( java.lang.String` **comment** `)`
  - **Usage**
    * Generates a graph comment to output stream.
  - **Parameters**
    * `comment` - the graph comment to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *increaseIndent*
  `public void` **increaseIndent**`( )`
  - **Usage**
    * Increases the current indent level.

- *initializeGenerator*
  `public void` **initializeGenerator**`( java.io.Writer` **newWriter**, `notio.KnowledgeBase` **newKnowledgeBase**, `notio.TranslationContext` **newTranslationContext** `)`
  - **Usage**
    * Initializes the generator to write to the specified writer using the specified TranslationContext and KnowledgeBase.
  - **Parameters**
    * `newWriter` - the writer to be generated to.
    * `newKnowledgeBase` - the knowledge base to be used while generating.
    * `newTranslationContext` - the translationContext to be used while generating.

- *LiteralDesignator*
  `public void` **LiteralDesignator**`( notio.LiteralDesignator` **designator** `)`
  - **Usage**
    * Generates a literal designator to output stream.
  - **Parameters**
    * `designator` - the designator to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *MarkerDesignator*
  `public void` **MarkerDesignator**`( notio.MarkerDesignator` **designator** `)`
  - **Usage**
    * Generates a marker designator to output stream.
  - **Parameters**
    * `designator` - the designator to be generated.

  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *NameDesignator*
  `public void` **NameDesignator**`( notio.NameDesignator` **designator** `)`
  - **Usage**
    * Generates a name designator to output stream.
  - **Parameters**
    * `designator` - the designator to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *Quantifier*
  `public void` **Quantifier**`( notio.Macro` **quantifier** `)`
  - **Usage**
    * Generates a quantifier to the output stream.
  - **Parameters**
    * `quantifier` - the quantifier to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an IO error occurs.

- *Referent*
  `public void` **Referent**`( notio.Referent` **referent** `)`
  - **Usage**
    * Generates a referent to the output stream.
  - **Parameters**
    * `referent` - the referent to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an IO error occurs.

- *RelationComment*
  `public void` **RelationComment**`( java.lang.String` **comment** `)`
  - **Usage**
    * Generates a relation comment to output stream.
  - **Parameters**
    * `comment` - the relation comment to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *RelationType*
  `public void` **RelationType**`( notio.RelationType` **relType** `)`
  - **Usage**
    * Generates a relation type to the output stream.
  - **Parameters**
    * `relType` - the relation type to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an IO error occurs.

METHODS INHERITED FROM CLASS `notio.translators.SimpleGenerator`

- *escapeCharactersInString*
  `public static String` **escapeCharactersInString**`( java.lang.String` **in**`, java.lang.String` **chars**`, char` **escapeSequence** `)`
  - **Usage**
    - Adds escape sequences for specified characters whenever they occur within the specified string.
  - **Parameters**
    - `in` - the string to be modified.
    - `chars` - a string containing all characters that need to be escaped.
    - `escapeSequence` - the character that should prefix characters that need escaping (e.g. backslash).
  - **Returns** - the modified string which now includes escape sequences.

- *generateUnit*
  `public void` **generateUnit**`( java.lang.Object` **unit** `)`
  - **Usage**
    - Generates a graph to the output stream.
  - **Parameters**
    - `unit` - the unit object to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.

- *getUnitClass*
  `public Class` **getUnitClass**`( )`
  - **Usage**
    - Returns a Class object that indicates what class the Unit parse method will return.
  - **Returns** - a Class object that indicates what class the Unit parse method will return.
  - **See Also**
    - `notio.Generator.generateUnit`

- *initializeGenerator*
  `public void` **initializeGenerator**`( java.io.Writer` **newWriter**`, notio.KnowledgeBase` **newKnowledgeBase**`, notio.TranslationContext` **newTranslationContext** `)`
  - **Usage**
    - Initializes the generator to write to the specified writer using the specified TranslationContext and KnowledgeBase.
  - **Parameters**
    - `newWriter` - the writer to be generated to.
    - `newKnowledgeBase` - the knowledge base to be used while generating.
    - `newTranslationContext` - the translationContext to be used while generating.

1.2.6 CLASS LFParser

A LF Parser class.

Declaration

public class LFParser
extends java.lang.Object
implements notio.Parser

Constructors

- *LFParser*
  `public` **LFParser( )**

Methods

- *getCreateTypesOnDemand*
  `public boolean` **getCreateTypesOnDemand( )**
  - **Usage**
    - This method may be used to check whether the parser will create type objects for any parsed type labels not currently in the database, or throw an exception. This flag is set using setCreateTypesOnDemand().
  - **Returns** - true if types will created on demand or false if an exception will be thrown.

- *getUnitClass*
  `public Class` **getUnitClass( )**
  - **Usage**
    - Returns the unit object parsed from the input stream.
  - **Returns** - the unit object parsed from the input stream.

- *initializeParser*
  `public void` **initializeParser(** `java.io.Reader` **reader,** `notio.KnowledgeBase` **kBase,** `notio.TranslationContext` **tContext)**
  - **Usage**
    - Initializes the parser to parse the specified reader.
  - **Parameters**
    - `reader` - the reader whose contents are to be parsed.
    - `kBase` - the knowledge base to be used while parsing.
    - `tContext` - the translation context to be used while parsing.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while initializing the parser.

- *parseActor*
  `public Actor` **parseActor( )**
  - **Usage**
    - Attempts to parse an actor from the input stream.
  - **Returns** - the actor parsed from the input stream.

  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *parseCGStream*
`public Vector` **parseCGStream( )**
  - **Usage**
    * Attempts to parse a CGStream from the input stream.
  - **Returns** - a Vector of graphs parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *parseConcept*
`public Concept` **parseConcept( )**
  - **Usage**
    * Attempts to parse a concept from the input stream.
  - **Returns** - the concept parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *parseGraph*
`public Graph` **parseGraph( )**
  - **Usage**
    * Attempts to parse a graph from the input stream.
  - **Returns** - the graph parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *parseOutermostContext*
`public Graph` **parseOutermostContext( )**
  - **Usage**
    * Attempts to parse a graph from the input stream.
  - **Returns** - the graph parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *parseRelation*
`public Relation` **parseRelation( )**
  - **Usage**
    * Attempts to parse a relation from the input stream.
  - **Returns** - the relation parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *parseUnit*
`public Object` **parseUnit( )**

  - **Usage**
    * Attempts to parse the default unit from the input stream. The default unit is whatever a particular parser is usually intended to parse.
  - **Returns** - the unit object parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.

- *setCreateTypesOnDemand*
  `public void` **setCreateTypesOnDemand**`( boolean` **flag** `)`

  - **Usage**
    * This method may be used to tell the parser whether it should create type objects for any parsed type labels not currently in the database, or throw an exception. Created types have only the universal and absurd types as parents and children respectively. The flag is true by default.
  - **Parameters**
    * `flag` - true or false to turn automatic creation on or off.

1.2.7 Class MarkerTable

A TranslationInfoUnit that serves as a symbol table relating foreign markers to native markers.

Declaration

public class MarkerTable
extends notio.translators.SimpleInfoUnit

Serializable Fields

- private Hashtable foreignIDToNativeMarkerTable
  - A lookup table that maps foreign markers to native markers.
- private Hashtable nativeMarkerToForeignIDTable
  - A lookup table that maps native markers to foreign markers.

Constructors

- *MarkerTable*
  `public` **MarkerTable**`( )`

  - **Usage**
    * Constructs a new MarkerTable.

- *MarkerTable*
  `public` **MarkerTable**`( notio.translators.MarkerTable` **originalTable** `)`

- **Usage**
  - Constructs a new MarkerTable that is a copy of the specified original. Changes to the new table will not affect the old one or vice versa.
- **Parameters**
  - `originalContext` - the translation context to be copied.

METHODS

- *clearForeignMarkerIDToNativeMarkerMapping*
  `public void` **clearForeignMarkerIDToNativeMarkerMapping**`( )`
  - **Usage**
    - Clears all existing mappings between foreign marker ID's and native markers.

- *copyUnit*
  `public TranslationInfoUnit` **copyUnit**`( )`
  - **Usage**
    - Returns a duplicate of this information unit that is distinct from the original. This means that changes to the original will not affect the duplicate and vice versa.
  - **Returns** - a duplicate of this unit.

- *getForeignMarkerIDByNativeMarker*
  `public String` **getForeignMarkerIDByNativeMarker**`( notio.Marker` **marker** `)`
  - **Usage**
    - Returns the foreign marker ID associated with the specified native marker in this translation context.
  - **Parameters**
    - `marker` - the native marker used for lookup.
  - **Returns** - the foreign marker ID associated with the marker, or null if no mapping exists.

- *getNativeMarkerByForeignMarkerID*
  `public Marker` **getNativeMarkerByForeignMarkerID**`( java.lang.String` **foreignID** `)`
  - **Usage**
    - Returns the native marker associated with the specified foreign marker ID in this translation context.
  - **Parameters**
    - `foreignID` - the foreign marker ID used for lookup.
  - **Returns** - the native marker associated with foreignID, or null if no mapping exists.

- *mapForeignMarkerIDToNativeMarker*
  `public void` **mapForeignMarkerIDToNativeMarker**`( java.lang.String` **foreignID**`, notio.Marker` **marker** `)`
  - **Usage**

  * Creates a two-way mapping between a foreign marker ID and a native marker instance. This is used to facilitate marker assignment and lookup while translating. During parsing, newly encountered foreign marker ID's can have a corresponding native marker instantiated and a mapping created. Subsequent references to the foreign ID during during parsing can then use the same native marker. The table may be used for generation purposes in order to preserve consistency between parsed and generated markers. This mapping may need to be cleared in some situations. For example, when parsing two graphs in succession that contain identical foreign marker ID's but whose resulting markers should not be identical internally. Exactly when the table should be cleared should be decided by the application.

- **Parameters**
  * `foreignID` - the foreign marker ID being mapped.
  * `marker` - the native marker being mapped.

- *resetUnit*
  `public void` **resetUnit**`( )`

  - **Usage**
    * Resets this information unit to its initial state. This method can be called by translators in order to ensure that a unit contains no information relating to earlier translation sessions.

METHODS INHERITED FROM CLASS `notio.translators.SimpleInfoUnit`

(in 1.2.8, page 36)

- *getUnitName*
  `public String` **getUnitName**`( )`

  - **Usage**
    * Returns the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Returns** - the name for this unit.

- *setUnitName*
  `public void` **setUnitName**`( java.lang.String` **newName** `)`

  - **Usage**
    * Sets the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Parameters**
    * `newName` - the new name for this unit.

1.2.8 CLASS SimpleInfoUnit

An abstract base-class for implementors the TranslationInfoUnit interface. This class simply provides support for the name-related methods.

Declaration

public abstract class SimpleInfoUnit
extends java.lang.Object
implements notio.TranslationInfoUnit

Serializable Fields

- private String unitName
  - The name of this unit.

Constructors

- *SimpleInfoUnit*
  `public` **SimpleInfoUnit**`( )`

Methods

- *getUnitName*
  `public String` **getUnitName**`( )`
  - **Usage**
    - Returns the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Returns** - the name for this unit.

- *setUnitName*
  `public void` **setUnitName**`( java.lang.String` **newName** `)`
  - **Usage**
    - Sets the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Parameters**
    - `newName` - the new name for this unit.

Chapter 2

Package notio

Package Contents *Page*

2.1 Interfaces

2.1.1 Interface **Generator**

The Generator interface. All generators should implement this interface to facilitate pluggability. While a generator must implement the entire interface, it can opt to throw an UnimplementedFeature exception if it does not support generating of a particular nonterminal. Minimally, a generator should support the Unit nonterminal.

Declaration

public interface Generator

Methods

- *generateActor*
  `public void` **generateActor**`( notio.Actor` **actor** `)`
  - **Usage**
    - Generates the specified actor.
  - **Parameters**
    - `actor` - the actor to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.
    - `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateCGStream*
  `public void` **generateCGStream**`( java.util.Vector` **graphs** `)`
  - **Usage**
    - Generates the specified vector of graphs as a CGStream.
  - **Parameters**
    - `graphs` - the vector of graphs to be generated.
  - **Exceptions**
    - `notio.GeneratorException` - if an error occurs while generating.
    - `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateConcept*
  `public void` **generateConcept**`( notio.Concept` **concept** `)`
  - **Usage**
    - Generates the specified concept.
  - **Parameters**
    - `concept` - the concept to be generated.
  - **Exceptions**

    * `notio.GeneratorException` - if an error occurs while generating.
    * `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateGraph*
  `public void` **generateGraph**`( notio.Graph` **graph** `)`
  - **Usage**
    * Generates the specified graph.
  - **Parameters**
    * `graph` - the graph to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.
    * `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateOutermostContext*
  `public void` **generateOutermostContext**`( notio.Graph` **graph** `)`
  - **Usage**
    * Generates the specified graph which is treated as the outermost context for purposes for scoping. This method should be used instead of generateGraph() when no translation information is to be used from previous translation sessions and when the generator can safely assume that the graph is "self-contained".
  - **Parameters**
    * `graph` - the graph to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.
    * `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateRelation*
  `public void` **generateRelation**`( notio.Relation` **relation** `)`
  - **Usage**
    * Generates the specified relation.
  - **Parameters**
    * `relation` - the relation to be generated.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.
    * `notio.UnimplementedFeatureException` - if this generator does not support this generation method.

- *generateUnit*
  `public void` **generateUnit**`( java.lang.Object` **unit** `)`
  - **Usage**
    * Generates the specified unit object.
  - **Parameters**
    * `unit` - the unit to be generated.

  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while generating.

- *getUnitClass*
  `public Class` **getUnitClass( )**
  - **Usage**
    * Returns a Class object that indicates what class the Unit generate method requires.
  - **Returns** - a Class object that indicates what class the Unit generate method requires.
  - **See Also**
    * `notio.Generator.generateUnit`

- *initializeGenerator*
  `public void` **initializeGenerator(** `java.io.Writer` **writer,** `notio.KnowledgeBase` **kBase,** `notio.TranslationContext` **tContext)**
  - **Usage**
    * Initializes the generator to generate into the specified writer.
  - **Parameters**
    * `writer` - the writer to be generated into.
    * `kBase` - the knowledge base to be used while parsing.
    * `tContext` - the translation context to be used while parsing.
  - **Exceptions**
    * `notio.GeneratorException` - if an error occurs while initializing the generator.

2.1.2 Interface Macro

Interface for Macros. This interface specifies the methods that all macros must implement. Essentially, a macro is some unspecified operator that can be executed. It is up to specific applications to decide when macros should be executed, what the appropriate arguments are, and what should be done with the results.

Declaration

public interface Macro

Methods

- *executeMacro*
  `public Object` **executeMacro(** `java.lang.Object []` **args)**
  - **Usage**
    * Executes the macro with the specified array of Objects for arguments and returns whatever results as an array of Objects.
  - **Parameters**
    * `args` - an array of Objects that are the argument to the macro.
  - **Returns** - the results of executing the macro as an array of Objects.

- *getName*
  `public String` **getName**`( )`
  - **Usage**
    - Returns the name of the macro (e.g. &forall)
  - **Returns** - the name of the macro.

2.1.3 Interface MarkerComparator

Interface for marker comparators. During graph matching, and in other situations, the Notio package may need to establish whether two markers are equivalent (refer to the same entity). This decision is likely to be application-specific since markers may refer to entities outside of the scope of the Notio layer, or the application may wish to form relationships when comparisons are made. In order to allow applications to specify the exact rules, they may provide an implementation of this interface as part of a MatchingScheme. The comparator will be called whenever two markers are compared. Possible uses include:

- comparing complex data objects such as images referred to by markers
- testing and creating equivalencies between markers due to assertions about them
- references to external database for which the markers are keys

Declaration

public interface MarkerComparator

Methods

- *compareMarkers*
  `public boolean` **compareMarkers**`( notio.Marker` **firstMarker**`, notio.Marker` **secondMarker** `)`
  - **Usage**
    - Called by matching routines when they need to determine whether two individual markers should be considered equivalent. Should return true if the two markers are considered to be equivalent, and false otherwise.
  - **Parameters**
    - `firstMarker` - the first of the two markers being compared.
    - `secondMarker` - the second of the two markers being compared.
  - **Returns** - true if the two markers should be treated as equivalent for matching purpose, false otherwise.

2.1.4 Interface Parser

The required interface for all parsers. All parsers should implement this interface to facilitate pluggability. While a parser must implement the entire interface, it can opt to throw an UnimplementedFeature exception if it does not support parsing of a particular nonterminal. Minimally, a parser should support the Unit nonterminal.

DECLARATION

public interface Parser

METHODS

- *getUnitClass*
  `public Class` **getUnitClass( )**
  - **Usage**
    - Returns a Class object that indicates what class the parseUnit() method will return.
  - **Returns** - a Class object that indicates what class the parseUnit() method will return.
  - **See Also**
    - `notio.Parser.parseUnit` (in 2.1.4, page 47)

- *initializeParser*
  `public void` **initializeParser(** `java.io.Reader` **reader,** `notio.KnowledgeBase` **kBase,** `notio.TranslationContext` **tContext)**
  - **Usage**
    - Initializes the parser to parse the specified character reader.
  - **Parameters**
    - `reader` - the reader whose contents are to be parsed.
    - `kBase` - the knowledge base to be used while parsing.
    - `tContext` - the translation context to be used while parsing.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while initializing the parser.

- *parseActor*
  `public Actor` **parseActor( )**
  - **Usage**
    - Attempts to parse an actor from the input stream.
  - **Returns** - the actor parsed from the input stream.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while parsing.
    - `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseCGStream*
  `public Vector` **parseCGStream( )**
  - **Usage**
    - Attempts to parse a CGStream from the input stream.
  - **Returns** - a vector of graphs parsed from the input stream.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while parsing.

    * `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseConcept*
  `public Concept` **parseConcept( )**
  - **Usage**
    * Attempts to parse a concept from the input stream.
  - **Returns** - the concept parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.
    * `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseGraph*
  `public Graph` **parseGraph( )**
  - **Usage**
    * Attempts to parse a graph from the input stream.
  - **Returns** - the graph parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.
    * `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseOutermostContext*
  `public Graph` **parseOutermostContext( )**
  - **Usage**
    * Attempts to parse a graph which is treated as the outermost context for purposes for scoping. This method should be used instead of parseGraph() when no translation information is to be used from previous translation sessions and when the parser can safely assume that the graph is "self-contained".
  - **Returns** - the graph parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.
    * `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseRelation*
  `public Relation` **parseRelation( )**
  - **Usage**
    * Attempts to parse a relation from the input stream.
  - **Returns** - the relation parsed from the input stream.
  - **Exceptions**
    * `notio.ParserException` - if an error occurs while parsing.
    * `notio.UnimplementedFeatureException` - if this parser does not support this parsing method.

- *parseUnit*
  `public Object` **parseUnit( )**
  - **Usage**
    - Attempts to parse the default unit from the input stream. The default unit is whatever a particular parser is usually intended to parse.
  - **Returns** - the unit object parsed from the input stream.
  - **Exceptions**
    - `notio.ParserException` - if an error occurs while parsing.

2.1.5 Interface **QuantifierMacro**

Interface for quantifier macros. This interface is an extension of the Macro interface with no additions. It is provided chiefly as a means for differentiating quantifier macros from other macros and also in case any quantifier specific interface needs to be added in the future.

Declaration

public interface QuantifierMacro
implements Macro

2.1.6 Interface **TranslationInfoUnit**

An interface for information units relating to translation so they can be given to different parsers and generators to ensure consistent representations. Common examples of these units include symbol tables. TranslationInfoUnits are contained within and passed around using TranslationContexts. Translators that are to exchange information via these units must agree on both a name for particular unit and a sub-interface of this interface which defines the methods usable on that unit. This allows for multiple instances of the same type of unit (e.g. a generic symbol table) but with different names for instances being used for different purposes.

Declaration

public interface TranslationInfoUnit
implements java.io.Serializable

Methods

- *copyUnit*
  `public TranslationInfoUnit` **copyUnit( )**
  - **Usage**
    - Returns a duplicate of this information unit that is distinct from the original. This means that changes to the original will not affect the duplicate and vice versa.
  - **Returns** - a duplicate of this unit.

- *getUnitName*
  `public String` **getUnitName( )**
  - **Usage**
    * Returns the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Returns** - the name for this unit.

- *resetUnit*
  `public void` **resetUnit( )**
  - **Usage**
    * Resets this information unit to its initial state. This method can be called by translators in order to ensure that a unit contains no information relating to earlier translation sessions.

- *setUnitName*
  `public void` **setUnitName(** `java.lang.String` **newName)**
  - **Usage**
    * Sets the name of this translation unit. Parsers and generators can use a name to retrieve the units they need from a TranslationContext.
  - **Parameters**
    * `newName` - the new name for this unit.

2.2 Classes

2.2.1 Class **Actor**

The actor node class.

Declaration

public class Actor
extends notio.Relation
implements java.io.Serializable

Constructors

- *Actor*
  `public` **Actor( )**
  - **Usage**
    * Constructs an actor that has no type and no arguments. The number of arguments is automatically set to zero.

- *Actor*
  `public` **Actor(** `notio.Concept []` **newArguments)**
  - **Usage**
    - Constructs an actor with no type and the specified arguments. For a non-zero valence, the last argument is assumed to be the sole output argument.
  - **Parameters**
    - `newArguments` - the concepts related by this actor.

- *Actor*
  `public` **Actor(** `notio.Concept []` **newArguments,** `int` **newOutputStartIndex)**
  - **Usage**
    - Constructs an actor with no type and the specified arguments with output arguments starting at the specified index.
  - **Parameters**
    - `newArguments` - the concepts related by this actor.
    - `newOutputStartIndex` - the index into the newArguments array at which the output arcs start.

- *Actor*
  `public` **Actor(** `notio.RelationType` **newType)**
  - **Usage**
    - Constructs an actor with the given type. The valence is taken from the specified type and the arguments are initialized to null. If the type's valence is unspecified, a zero-length argument array is used for construction. For a specified non-zero valence, the last argument is assumed to be the sole output argument.
  - **Parameters**
    - `newType` - the type for this actor.

- *Actor*
  `public` **Actor(** `notio.RelationType` **newType,** `notio.Concept []` **newArguments)**
  - **Usage**
    - Constructs an actor with the given type and arguments. The last argument is assumed to be the sole output argument. The number of arguments must conform to the valence of the relation type unless it is unspecified in which case the valence is set to the number of arguments in the array.
  - **Parameters**
    - `newType` - the type for this actor.
    - `newArguments` - the concepts related by this actor.

- *Actor*
  `public` **Actor(** `notio.RelationType` **newType,** `notio.Concept []` **newInputArguments,** `notio.Concept []` **newOutputArguments)**
  - **Usage**

    * Constructs an actor with the given type and the specified input and output arguments. The number of arguments must conform to the valence of the relation type unless it is unspecified in which case the valence is set to the number of arguments in the array.
  - **Parameters**
    * `newType` - the type for this actor.
    * `newInputArguments` - the input concepts related by this actor.
    * `newOutputArguments` - the output concepts related by this actor.

- *Actor*
  `public` **Actor(** `notio.RelationType` **newType,** `notio.Concept []` **newArguments,** `int` **newOutputStartIndex)**
  - **Usage**
    * Constructs an actor with the given type and arguments and output argument start index. The number of arguments must conform to the valence of the relation type unless it is unspecified in which case the valence is set to the number of arguments in the array.
  - **Parameters**
    * `newType` - the type for this actor.
    * `newArguments` - the concepts related by this actor.
    * `newOutputStartIndex` - the index into the newArguments array at which the output arcs start.

METHODS

- *copy*
  `public Relation` **copy(** `notio.CopyingScheme` **copyScheme)**
  - **Usage**
    * Performs a copy operation on this actor according to the the specified CopyingScheme. The result may be a new node or simply a reference to this actor depending on the scheme. Note that the returned object is truly an Actor object even though the return type is Relation. This is because Java does not allow overriding of return types.
  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
  - **Returns** - the result of the copy operation.

- *copy*
  `public Relation` **copy(** `notio.CopyingScheme` **copyScheme,** `java.util.Hashtable` **substitutionTable)**
  - **Usage**
    * Performs a copy operation on this actor according to the the specified CopyingScheme. The result may be a new node or simply a reference to this actor depending on the scheme. Note that the returned object is truly an Actor object even though the return type is Relation. This is because Java does not allow overriding of return types.

- **Parameters**
  - `copyScheme` - the copying scheme used to control the copy operation.
  - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
- **Returns** - the result of the copy operation.

METHODS INHERITED FROM CLASS `notio.Relation`

(in 2.2.37, page 153)

- *copy*
  `public Relation` **copy**`( notio.CopyingScheme` **copyScheme** `)`
  - **Usage**
    - Performs a copy operation on this relation according to the the specified CopyingScheme. The result may be a new node or simply a reference to this relation depending on the scheme.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
  - **Returns** - the result of the copy operation.

- *copy*
  `public Relation` **copy**`( notio.CopyingScheme` **copyScheme**`, java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    - Performs a copy operation on this relation according to the the specified CopyingScheme. The result may be a new node or simply a reference to this relation depending on the scheme.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
    - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getArguments*
  `public Concept` **getArguments**`( )`
  - **Usage**
    - Returns all of the concepts related by this relation. Input arguments are followed by output arguments. In a 'classic' relation, the last argument is always the sole output argument. If the relation has a type whose valence has changed since its definition, the arguments will not be adjusted. A call to isComplete() should be made to effect this adjustment.
  - **Returns** - the arguments to this relation.

- *getInputArguments*
  `public Concept` **getInputArguments**`( )`
  - **Usage**
    - Returns the input arguments for this relation. If the relation has a type whose valence has changed since its definition, the arguments will not be adjusted. A call to isComplete() should be made to effect this adjustment.
  - **Returns** - an array, possibly empty, containing input arguments for this relation.

- *getOutputArguments*
  `public Concept` **getOutputArguments**`( )`

- **Usage**
  - Returns the output arguments for this relation. If the relation has a type whose valence has changed since its definition, the arguments will not be adjusted. A call to isComplete() should be made to effect this adjustment.
- **Returns** - an array, possibly empty, containing output arguments for this relation.

- *getOutputStartIndex*
`public int` **getOutputStartIndex**`( )`
  - **Usage**
    - Returns the index of the first output argument within the array returned by getArguments(). This method will return -1 if the relation has a valence of zero. If there are inputs, but no outputs, the output start index will be 1 greater than the number of inputs. This means that any code using the output start index to index into the array returned by getArguments() should check first to see if the array is long enough.
  - **Returns** - the index of the first output argument within the array returned by getArguments().
  - **See Also**
    - `notio.Relation.getArguments()`

- *getType*
`public RelationType` **getType**`( )`
  - **Usage**
    - Returns this relation's type.
  - **Returns** - this relation's type.

- *getValence*
`public int` **getValence**`( )`
  - **Usage**
    - This method returns the valence (number of arguments) that this relation has defined. This number includes null arguments. If this relation's type has a defined valence, the number returned by this method will equal the valence of the type.
  - **Returns** - the number of arguments this relation has defined (including nulls).

- *isComplete*
`public boolean` **isComplete**`( )`
  - **Usage**
    - Returns true if this relation's arguments are completely specified (are all non-null). If the relation has a type, its valence will be checked to ensure that the relation is complete according to the type. This check is made in case the valence of the type has changed. If it has changed, the arguments will be adjusted accordingly.
  - **Returns** - true if this relation's arguments are all non-null.

- *matchRelations*
`public static boolean` **matchRelations**`( notio.Relation` **first**`, notio.Relation` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    - Compares two relations to decide if they match. The exact semantics of matching are determined by the matching scheme.
  - **Parameters**
    - `first` - the first relation being matched.
    - `second` - the second relation being matched.
    - `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two concepts match according to the scheme's criteria.

- *relatesConcept*
  `public boolean` **relatesConcept**`( notio.Concept` **concept** `)`
  - **Usage**
    - Returns true if the specified concept is an argument of this relation.
  - **Parameters**
    - `concept` - the concept being checked for.
  - **Returns** - true if the specified concept is an argument of this relation.

- *replaceArgument*
  `public void` **replaceArgument**`( notio.Concept` **oldConcept**`, notio.Concept` **newConcept** `)`
  - **Usage**
    - Replaces all occurrences of the specified argument with a new one regardless of whether it is an input or output argument. Replacing nulls with non-nulls or vice-versa is allowed.
  - **Parameters**
    - `oldConcept` - the concept being replaced.
    - `newConcept` - the replacement concept.

- *replaceInputArgument*
  `public void` **replaceInputArgument**`( notio.Concept` **oldConcept**`, notio.Concept` **newConcept** `)`
  - **Usage**
    - Replaces all occurrences of the specified input argument with a new one. Replacing nulls with non-nulls or vice-versa is allowed. Replacing an argument with null effectively removes that argument.
  - **Parameters**
    - `oldConcept` - the input argument being replaced.
    - `newConcept` - the replacement argument.

- *replaceOutputArgument*
  `public void` **replaceOutputArgument**`( notio.Concept` **oldConcept**`, notio.Concept` **newConcept** `)`
  - **Usage**
    - Replaces all occurrences of the specified output argument with a new one. Replacing nulls with non-nulls or vice-versa is allowed. Replacing an argument with null effectively removes that argument.
  - **Parameters**
    - `oldConcept` - the output argument being replaced.
    - `newConcept` - the replacement argument.

- *restrictTo*
  `public Relation` **restrictTo**`( notio.RelationType` **subType** `)`
  - **Usage**
    - Returns a new node identical to this but restricted to the new type.
  - **Parameters**
    - `subType` - the type to be restricted to.
  - **Returns** - the new restricted relation.
  - **Exceptions**
    - `notio.RestrictionException` - if subType is not a real subtype of the current type

---

- *setArgument*
  `public void` **setArgument**`( int` **index**`, notio.Concept` **newConcept** `)`
  - **Usage**

    * Sets the specified argument to the specified concept. The index of the first argument (often labelled "1" in diagrams) is zero. If the argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the specified index exceeds that valence, a IllegalArgumentException is thrown. If the valence is undefined and a previously unused argument index is specified, the valence of this particular relation will be increased to accomodate the index. No other relations nor the relation's type are affected by this increase. All currently specified arguments are preserved and any newly created but unspecified arguments are set to null. If the valence of the relation's type has changed since the arguments were specified, the arguments will be adjusted as if isComplete() had been called. Setting an argument to null effectively removes that argument.
  - **Parameters**
    * `index` - the index of the argument being set.
    * `newConcept` - the concept to be used as an argument.

- *setArguments*
  `public void` **setArguments**`( notio.Concept [] ` **newConcepts** `)`
  - **Usage**
    * Sets the arguments according to the specified array of concepts. If an argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the number of arguments in the array exceeds that valence, a IllegalArgumentException is thrown. Otherwise, this method behaves like a series of calls to setArgument(), with the array index used as the argument's index.
  - **Parameters**
    * `newConcepts` - the array of concepts to be used as arguments.
  - **See Also**
    * `notio.Relation.setArgument`

- *setInputArgument*
  `public void` **setInputArgument**`( int ` **index**`, notio.Concept ` **newConcept** `)`
  - **Usage**
    * Sets the specified argument to the specified concept. If the argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the specified index exceeds that valence, a IllegalArgumentException is thrown. If the valence is undefined and a previously unused argument index is specified, the valence of this particular relation will be increased to accomodate the index. No other relations nor the relation's type are affected by this increase. All currently specified arguments are preserved and any newly created but unspecified arguments are set to null. If the valence of the relation's type has changed since the arguments were specified, the arguments will be adjusted as if isComplete() had been called. Setting an argument to null effectively removes that argument.
  - **Parameters**
    * `index` - the index of the input argument being set (the nth input argument).
    * `newConcept` - the concept to be used as an argument.

- *setOutputArgument*
  `public void` **setOutputArgument**`( int ` **index**`, notio.Concept ` **newConcept** `)`
  - **Usage**
    * Sets the specified argument to the specified concept. If the argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the specified index exceeds that valence, a IllegalArgumentException is thrown. If the valence is undefined and a previously unused argument index is specified, the valence of this particular relation will be increased to accomodate the index. No other relations nor the relation's type are affected by this increase. All currently specified arguments are

preserved and any newly created but unspecified arguments are set to null. If the valence of the relation's type has changed since the arguments were specified, the arguments will be adjusted as if isComplete() had been called. Setting an argument to null effectively removes that argument.

- **Parameters**
  - `index` - the index of the output argument being set (the nth output argument).
  - `newConcept` - the concept to be used as an argument.

- *setOutputStartIndex*
  `public void` **setOutputStartIndex**`( int` **newOutputStartIndex** `)`
  - **Usage**
    - Sets the index of the first output argument within the array returned by getArguments(). For relations with a valence of zero, the only valid start index is -1. For all other valences, the start index must be within the array's range unless there are no outputs, in which case the index may be 1 greater than the length of the array. Note that this method should rarely be needed since the start index can be specified in a constructor and it is unlikely to change. This method is provided chiefly for completeness.
  - **Parameters**
    - `newOutputStartIndex` - the index of the first output argument within the array returned by getArguments().
  - **See Also**
    - `notio.Relation.getArguments()`

- *setType*
  `public void` **setType**`( notio.RelationType` **newType** `)`
  - **Usage**
    - Sets the type for this relation. If a type has already been set, it will be replaced. If the new type has a valence that is different from the previous type and arguments have already been specified for this relation, the arguments will be updated as if isComplete() had been called.
  - **Parameters**
    - `newType` - the new type for this relation.
  - **See Also**
    - `notio.Relation.isComplete` (in 2.2.37, page 157)

Methods inherited from class `notio.Node`

(in 2.2.31, page 137)

- *getComment*
  `public String` **getComment**`( )`
  - **Usage**
    - Returns the comment string for this node.
  - **Returns** - the comment string associated with this node or null.

- *getEnclosingGraph*
  `public Graph` **getEnclosingGraph**`( )`
  - **Usage**
    - Returns the graph that encloses this node or null if the node does not currently belong to a graph.
  - **Returns** - the node's enclosing graph.

- *setComment*
  `public void` **setComment**`( java.lang.String` **newComment** `)`
  - **Usage**
    - Sets the comment string for this node.
  - **Parameters**
    - `newComment` - the new comment string for this node.

2.2.2 Class ActorAddError

Error thrown when the addition of a actor gives rise to an error.

Declaration

public class ActorAddError
extends notio.OperationError

Constructors

- *ActorAddError*
  `public` **ActorAddError**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an error with the specified message.
  - **Parameters**
    - `message` - The details of the error.
- *ActorAddError*
  `public` **ActorAddError**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an error with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the error.
    - `newSubThrowable` - the sub-throwable to be embedded in this error.

Methods inherited from class `notio.OperationError`

(in 2.2.33, page 141)
- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

METHODS INHERITED FROM CLASS `java.lang.Error`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.3 CLASS AdjacencyMatrix

This class provides an adjacency matrix representation suitable for conceptual graphs. The matrix is three-dimensional. Since CG's are bipartite, the first and second dimensions correspond to relations and concepts respectively. The first element in the third dimension is the number of arcs between the specified concept and relation. Note that CG's can indeed have multiple arcs between a concept and relation since arcs are ordered and therefore distinct. A zero indicates that the nodes are not adjacent. The remaining elements along the third dimension of an entry are the numbers of the arcs connecting the two nodes. For example, if a relation and concept had two arcs number 1 and 4 connecting them, the entry would be an array like this: { 2, 1, 4 }. Of course, if one only wishes to test the adjacency of two nodes, one can simply look to see if the first element of the third dimension is non-zero. Once generated, instances of this class are independent of the graph from which they are derived. That is, if the original graph changes, the adjacency matrix is not updated.

This alternate representation is provided to allow for the easy implementation of certain graph algorithms. Since efficiency is a primary concern and instances of this class are not updated along with the structures upon which they are based, the usual Notio convention of using accessor (getX/putX methods) which provide safe access to the information is avoided and public members are used instead.

Declaration

public class AdjacencyMatrix
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- public Concept concepts
  - The array of concepts indicating the concept ordering.
- public Relation relations
  - The array of relations indicating the relation ordering.
- public int adjacencies
  - A two dimensional array with the adjacencies. The first index is for relations, the second for concepts, and the third is for arc count followed by arc numbers.

Fields

- public Concept concepts
  - The array of concepts indicating the concept ordering.
- public Relation relations
  - The array of relations indicating the relation ordering.
- public int adjacencies
  - A two dimensional array with the adjacencies. The first index is for relations, the second for concepts, and the third is for arc count followed by arc numbers.

Constructors

- *AdjacencyMatrix*
  `public` **`AdjacencyMatrix( )`**
  - **Usage**
    - Constructs an AdjacencyMatrix with no initial values for the fields.

- *AdjacencyMatrix*
  `public` **`AdjacencyMatrix(`** `notio.Concept []` **`newConcepts,`** `notio.Relation []` **`newRelations,`** `int [][][]` **`newAdjacencies )`**
  - **Usage**
    - Constructs an AdjacencyMatrix by copying the supplied arrays.
  - **Parameters**
    - `newConcepts` - the concept ordering array for the matrix.
    - `newRelations` - the relation ordering array for the matrix.
    - `newAdjacencies` - the matrix of adjancies.

2.2.4 Class **Concept**

The concept node class. A concept consists primarily of a type and a referent, either or both of which can be null. Concepts can also be members of coreference sets, linking them to other concepts.

Declaration

public class Concept
extends notio.Node
implements java.io.Serializable

Serializable Fields

- private Referent referent
  - The referent for this concept.
- private Vector corefSets
  - The coreference sets associated with this concept.

Constructors

- *Concept*
  `public` **Concept**`( )`
  - **Usage**
    - Constructs a concept neither type nor referent. Both type and referent are initialized to null. A null referent indicates a generic concept.

- *Concept*
  `public` **Concept**`( notio.ConceptType` **newType** `)`
  - **Usage**
    - Constructs a concept with the specified type and no referent. The referent in this case is set to null, indicating a generic concept.
  - **Parameters**
    - `newType` - the concept type for this concept.

- *Concept*
  `public` **Concept**`( notio.ConceptType` **newType**`, notio.Referent` **newReferent** `)`
  - **Usage**
    - Constructs a concept with the specified type and referent. A null value may be used for either or both. A null referent indicates a generic concept.
  - **Parameters**
    - `newType` - the concept type for this concept.

    * `newReferent` - the referent for this concept.

- *Concept*
  `public` **Concept**`( notio.Referent` **newReferent** `)`
  - **Usage**
    * Constructs a concept with the specified referent and no type. A null referent indicates a generic concept.
  - **Parameters**
    * `newReferent` - the referent for this concept.

METHODS

- *copy*
  `public Concept` **copy**`( notio.CopyingScheme` **copyScheme** `)`
  - **Usage**
    * Performs a copy operation on this concept according to the the specified CopyingScheme. The result may be a new object of exactly the same class as the original or simply a reference to this concept depending on the copying scheme. Coreference sets will be copied as needed depending on the copying scheme.
  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
  - **Returns** - the result of the copy operation.

- *copy*
  `public Concept` **copy**`( notio.CopyingScheme` **copyScheme**`, java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    * Performs a copy operation on this concept according to the the specified CopyingScheme. The result may be a new object of exactly the same class as the original or simply a reference to this concept depending on the copying scheme. Coreference sets will be copied as needed depending on the copying scheme.
  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
    * `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getCoreferenceSets*
  `public CoreferenceSet` **getCoreferenceSets**`( )`
  - **Usage**
    * Returns the coreferences sets of which this concept is a member. If this method returns an empty array, then this concept is considered to be the defining concept and sole member of an "implicit" coreference set. This means that that the isDefiningConcept() method will return true if this method returns an empty array. Coreference sets only need to be created when they have multiple concepts.

- **Returns** - an array containing all of the coreference sets of which this concept is a member, possibly empty.

- *getCoreferentConcepts*
`public Concept` **getCoreferentConcepts( )**
  - **Usage**
    - Returns all of the concepts coreferent to this concept. This is essentially the union of all the coreferent sets to which this concept belongs. Note that every concept is coreferent to itself so this method will always return at least one element. Note that if this concept has no explicitly added coreference sets, this method will still return the concept itself, the sole member of an "implicit" coreference set.
  - **Returns** - an array containing of the concepts coreferent to this concept.

- *getEnclosedGraph*
`public Graph` **getEnclosedGraph( )**
  - **Usage**
    - Returns the enclosed (nested) descriptor graph if this concept is a context or null otherwise. This is really a convenience method for quickly traversing compound graphs. It is a counter-part to getEnclosingGraph().
  - **Returns** - the enclosed graph if this concept is a context, or null otherwise.
  - **See Also**
    - `notio.Node.getEnclosingGraph()`

- *getReferent*
`public Referent` **getReferent( )**
  - **Usage**
    - Returns this concept's referent. Null indicates a generic concept.
  - **Returns** - this concept's referent.

- *getRelators*
`public Relation` **getRelators( )**
  - **Usage**
    - Returns an array (possibly empty) of the relations in the enclosing graph that relate this concept. This method will return null if the concept does not belong to any graph (getEnclosingGraph() returns null).
  - **Returns** - an array containing of the relations that relate this concept.

- *getType*
`public ConceptType` **getType( )**
  - **Usage**
    - Returns this concept's type. Null indicates an untyped concept.
  - **Returns** - this concept's type

- *isContext*
`public boolean` **isContext( )**

  - **Usage**
    * Returns true if this concept is a context. A context is a concept whose descriptor is a non-blank conceptual graph. This method will return true if this concept has a referent which returns true when its isContext() method is called.
  - **Returns** - true if this concept is a context.
  - **See Also**
    * `notio.Referent.isContext()`

- *isDefiningConcept*
`public boolean` **isDefiningConcept**`( )`

  - **Usage**
    * Returns true if this concept is the defining concept of a coreference set. Note that if this concept is not part of any explicit coreference sets, it is considered to be the defining node and sole member of an "implicit" coreference set, so this method will return true.
  - **Returns** - true if this concept is the defining concept of a coreference set.

- *isDominantConcept*
`public boolean` **isDominantConcept**`( )`

  - **Usage**
    * Returns true if this concept is a dominant node of a coreference set. Note that if this concept is not part of any explicit coreference sets, it is considered to be a dominant node and sole member of an "implicit" coreference set, so this method will return true.
  - **Returns** - true if this concept is a dominant node of a coreference set.

- *isEnclosedBy*
`public boolean` **isEnclosedBy**`( notio.Concept` **concept** `)`

  - **Usage**
    * Returns true if this concept is enclosed by the specified concept.
  - **Parameters**
    * `concept` - the concept being checked for as enclosing this concept.
  - **Returns** - true if this concept is enclosed by the specified concept.

- *isEnclosedBy*
`public boolean` **isEnclosedBy**`( notio.Graph` **graph** `)`

  - **Usage**
    * Returns true if this concept is enclosed by the specified graph.
  - **Parameters**
    * `graph` - the graph being checked for as enclosing this concept.
  - **Returns** - true if this concept is enclosed by the specified graph.

- *isGeneric*
`public boolean` **isGeneric**`( )`

  - **Usage**

    * Returns true if this concept is generic; false if the concept is specific. A generic concept either has a referent of null, or a referent in which the quantifier, designator, and descriptor are null.
  - **Returns** - true if this concept is generic; false if specific.

- *isolate*
  `public void isolate( )`
  - **Usage**
    * Isolates this concept by removing it from all coreference sets to which it belongs and by isolating any and all concepts that may be nested within it.

- *matchConcepts*
  `public static MatchResult` **matchConcepts**`( notio.Concept` **first**`, notio.Concept` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    * Compares two concepts to decide if they match. The exact semantics of matching are determined by the matching scheme. This method may compare more than two concepts depending on the setting of the coreference matching flags.
  - **Parameters**
    * `first` - the first concept being matched.
    * `second` - the second concept being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two concepts match according to the scheme's criteria.

- *restrictTo*
  `public Concept` **restrictTo**`( notio.ConceptType` **subType** `)`
  - **Usage**
    * Returns a new concept identical to this but restricted to the new type.
  - **Parameters**
    * `subType` - the type to be restricted to.
  - **Returns** - the new restricted concept.
  - **Exceptions**
    * `notio.RestrictionException` - if subType is not a real subtype of the current type

---

- *restrictTo*
  `public Concept` **restrictTo**`( notio.ConceptType` **subType**`, notio.Referent` **newReferent** `)`
  - **Usage**
    * Returns a new concept identical to this but restricted to the given referent and subtype.
  - **Parameters**
    * `subType` - the type to be restricted to.
    * `newReferent` - the referent to be restricted to.
  - **Returns** - the new restricted concept.

  - **Exceptions**
    * `notio.RestrictionException` - if subType is not a real sub-type of the current type and/or if there is already an equally specific referent.

- *restrictTo*
`public Concept` **restrictTo**`( notio.Referent` **newReferent** `)`
  - **Usage**
    * Returns a new concept identical to this but restricted to the given referent.
  - **Parameters**
    * `newReferent` - the referent to be restricted to.
  - **Returns** - the new restricted concept.
  - **Exceptions**
    * `notio.RestrictionException` - if there is already an equally specific referent

---

- *setReferent*
`public void` **setReferent**`( notio.Referent` **newReferent** `)`
  - **Usage**
    * Sets this concept's referent. If a referent has already been set, it is replaced. The referent may be set to null, indicating a generic concept.
  - **Parameters**
    * `newReferent` - the new referent for this concept.

- *setType*
`public void` **setType**`( notio.ConceptType` **newType** `)`
  - **Usage**
    * Sets this concept's type. If a type has already been set, it is replaced.
  - **Parameters**
    * `newType` - the new type for this concept.

- *testCoreference*
`public static boolean` **testCoreference**`( notio.Concept` **first**`, notio.Concept` **second** `)`
  - **Usage**
    * Returns true if the two concepts specified are coreferent. Note that every concept is coreferent to itself so this method will return true if the first and second concept are the same concept.
  - **Parameters**
    * `first` - the first concept.
    * `second` - the second concept.
  - **Returns** - true if the two concepts specified are coreferent.

METHODS INHERITED FROM CLASS `notio.Node`

(in 2.2.31, page 137)

- *getComment*
  `public String` **getComment**`( )`
  - **Usage**
    - Returns the comment string for this node.
  - **Returns** - the comment string associated with this node or null.

- *getEnclosingGraph*
  `public Graph` **getEnclosingGraph**`( )`
  - **Usage**
    - Returns the graph that encloses this node or null if the node does not currently belong to a graph.
  - **Returns** - the node's enclosing graph.

- *setComment*
  `public void` **setComment**`( java.lang.String` **newComment** `)`
  - **Usage**
    - Sets the comment string for this node.
  - **Parameters**
    - `newComment` - the new comment string for this node.

2.2.5 CLASS ConceptAddError

Error thrown when the addition of a concept gives rise to an error.

DECLARATION

public class ConceptAddError
extends notio.OperationError

CONSTRUCTORS

- *ConceptAddError*
  `public` **ConceptAddError**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an error with the specified message.
  - **Parameters**
    - `message` - The details of the error.

- *ConceptAddError*
  `public` **ConceptAddError**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**

    * Constructs an error with the specified message and sub-throwable.
  - **Parameters**
    * `message` - the details of the error.
    * `newSubThrowable` - the sub-throwable to be embedded in this error.

METHODS INHERITED FROM CLASS `notio.OperationError`

(in 2.2.33, page 141)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    * This method retrieves arbitrary throwables embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    * This method allows arbitrary throwables to be embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    * `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

METHODS INHERITED FROM CLASS `java.lang.Error`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.6 Class ConceptRemoveException

Exception thrown when the removal of a concept gives rise to an error.

Declaration

public class ConceptRemoveException
extends notio.OperationException

Constructors

- *ConceptRemoveException*
  `public` **ConceptRemoveException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.

- *ConceptRemoveException*
  `public` **ConceptRemoveException**`( java.lang.String` **message**`,`
  `java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.
    - `newSubThrowable` - the sub-throwable to be embedded in this exception.

Methods inherited from class `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.

– **Parameters**
  * `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.7 CLASS ConceptReplaceException

Exception thrown when the replacing of a concept gives rise to an error.

DECLARATION

public class ConceptReplaceException
extends notio.OperationException

CONSTRUCTORS

- *ConceptReplaceException*
  `public` **ConceptReplaceException**`( java.lang.String` **message** `)`
  – **Usage**
    * Constructs an exception with the specified message.
  – **Parameters**
    * `message` - The details of the exception.
- *ConceptReplaceException*
  `public` **ConceptReplaceException**`( java.lang.String` **message**`,`
  `java.lang.Throwable` **newSubThrowable** `)`

- **Usage**
  - Constructs an exception with the specified message and sub-throwable.
- **Parameters**
  - `message` - the details of the exception.
  - `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.8 Class **ConceptType**

The concept type class. This class encapsulates all available information about a concept type. The type can be defined by a label and/or a type definition.

Declaration

public class ConceptType
extends notio.Type
implements java.io.Serializable

Serializable Fields

- private String label
  - The type label for this type.
- private ConceptTypeDefinition typeDefinition
  - The type definition for this type.
- private String comment
  - The comment associated with this type.

Constructors

- *ConceptType*
  `public` **ConceptType**`( )`
  - **Usage**
    - Constructs a ConceptType with no type label or type definition. Note that this method does not add the type to any hierarchy.

- *ConceptType*
  `public` **ConceptType**`( notio.ConceptTypeDefinition` **newDefinition** `)`
  - **Usage**
    - Constructs an unlabelled ConceptType with the specified type definition. Note that this method does not add the type to any hierarchy.
  - **Parameters**
    - `newDefinition` - the type definition for this type.

- *ConceptType*
  `public` **ConceptType**`( java.lang.String` **newLabel** `)`
  - **Usage**
    - Constructs a labelled ConceptType with the specified type label and no type definition. Note that this method does not add the type to any hierarchy.

  - **Parameters**
    * `newLabel` - the type label for this type.

- *ConceptType*
  `public` **ConceptType**`( java.lang.String` **newLabel**`,`
  `notio.ConceptTypeDefinition` **newDefinition** `)`

  - **Usage**
    * Constructs a labelled ConceptType with the specified type label and type definition. Note that this method does not add the type to any hierarchy.
  - **Parameters**
    * `newLabel` - the type label for this type.
    * `newDefinition` - the type definition for this type.

METHODS

- *getComment*
  `public String` **getComment**`( )`

  - **Usage**
    * Returns the comment string for this type.
  - **Returns** - the comment string associated with this type or null.

- *getImmediateSubTypes*
  `public ConceptType` **getImmediateSubTypes**`( )`

  - **Usage**
    * Returns the immediate subtypes of this type.
  - **Returns** - an array of immediate subtypes of this type.

- *getImmediateSuperTypes*
  `public ConceptType` **getImmediateSuperTypes**`( )`

  - **Usage**
    * Returns the immediate supertypes of this type.
  - **Returns** - an array of immediate supertypes of this type.

- *getLabel*
  `public String` **getLabel**`( )`

  - **Usage**
    * Returns the type label for this type.
  - **Returns** - the string that is the label for this type.

- *getProperSubTypes*
  `public ConceptType` **getProperSubTypes**`( )`

  - **Usage**
    * Returns all subtypes of this type.
  - **Returns** - an array of subtypes of this type.

- *getProperSuperTypes*
  `public ConceptType` **getProperSuperTypes( )**
  - **Usage**
    - Returns all supertypes of this type.
  - **Returns** - an array of supertypes of this type.

- *getTypeDefinition*
  `public ConceptTypeDefinition` **getTypeDefinition( )**
  - **Usage**
    - Returns the type definition for this type (if any).
  - **Returns** - the type definition for this type or null if there isn't one.

- *hasProperSubType*
  `public boolean` **hasProperSubType(** `notio.ConceptType` **queryType)**
  - **Usage**
    - Tests whether the specified type is a proper subtype of this type.
  - **Parameters**
    - `queryType` - the type being tested.
  - **Returns** - true if queryType is a proper subtype of this type, false otherwise.

- *hasProperSuperType*
  `public boolean` **hasProperSuperType(** `notio.ConceptType` **queryType)**
  - **Usage**
    - Tests whether the specified type is a proper supertype of this type.
  - **Parameters**
    - `queryType` - the type being tested.
  - **Returns** - true if queryType is a proper supertype of this type, false otherwise.

- *hasSubType*
  `public boolean` **hasSubType(** `notio.ConceptType` **queryType)**
  - **Usage**
    - Tests whether the specified type is a subtype of this type.
  - **Parameters**
    - `queryType` - the type being tested.
  - **Returns** - true if queryType is a subtype of this type, false otherwise.

- *hasSuperType*
  `public boolean` **hasSuperType(** `notio.ConceptType` **queryType)**
  - **Usage**
    - Tests whether the specified type is a supertype of this type.
  - **Parameters**
    - `queryType` - the type being tested.
  - **Returns** - true if queryType is a supertype of this type, false otherwise.

- *matchConceptTypes*
  `public static boolean` **matchConceptTypes**`( notio.ConceptType` **first**`, notio.ConceptType` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    - Compares two concept types to decide if they match. The exact semantics of matching are determined by the match control flag.
  - **Parameters**
    - `first` - the first concept type being matched.
    - `second` - the second concept type being matched.
    - `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two concept types match according to the scheme's criteria.

- *setComment*
  `public void` **setComment**`( java.lang.String` **newComment** `)`
  - **Usage**
    - Sets the comment string for this type.
  - **Parameters**
    - `newComment` - the new comment string for this type.

- *setLabel*
  `public void` **setLabel**`( java.lang.String` **newLabel** `)`
  - **Usage**
    - Sets the type label for this type. If the type already had a label, it is replaced. If a null is used, the label is removed from the type.
  - **Parameters**
    - `newLabel` - the string that is the label for this type.
  - **Exceptions**
    - `notio.TypeChangeError` - if the type belongs to a hierarchy and the new label is already in use within it.

- *setTypeDefinition*
  `public void` **setTypeDefinition**`( notio.ConceptTypeDefinition` **newDefinition** `)`
  - **Usage**
    - Sets the type definition for this type. If the type already has a definition, it is replaced. If null is used, the type definition will be removed.
  - **Parameters**
    - `newDefinition` - the new type definition for this type.

METHODS INHERITED FROM CLASS `notio.Type`

- *getLabel*
  `public abstract String` **getLabel**`( )`
  - **Usage**
    - Returns the type label for this type.
  - **Returns** - the type label for this type.

2.2.9 Class **ConceptTypeDefinition**

The concept type definition class. This class provides the functionality of a monadic lambda expression used for describing a concept type.

Declaration

public class ConceptTypeDefinition
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private Abstraction abstraction
  - The abstraction used in this definition.
- private ConceptType signature
  - The signature used in this definition.

Constructors

- *ConceptTypeDefinition*
  `public` **ConceptTypeDefinition**`( notio.Concept` **newParameter**`, notio.Graph` **newDifferentia** `)`
  - **Usage**
    - Constructs a new concept type definition using the differentia graph and the concept in that graph that acts as a formal parameter for this definition. The parameter concept must have a null referent. The signature of this definition is derived from the formal parameter.
  - **Parameters**
    - `newParameter` - the single concept that is the formal parameter used in this definition.
    - `newDifferentia` - the differentia graph for this definition.

- *ConceptTypeDefinition*
  `public` **ConceptTypeDefinition**`( notio.ConceptType` **newSignature** `)`
  - **Usage**
    - Constructs a new concept type definition using the specified signature.
  - **Parameters**
    - `newSignature` - the single concept type that forms the signature used in this definition.

METHODS

- *getDifferentia*
  `public Graph` **getDifferentia( )**
  - **Usage**
    - Returns the differentia graph for this definition or null if no differentia graph has been specified.
  - **Returns** - the differentia graph for this definition.

- *getFormalParameter*
  `public Concept` **getFormalParameter( )**
  - **Usage**
    - Returns the formal parameter concept or null if no differentia graph has been specified.
  - **Returns** - the formal parameter concept.

- *getSignature*
  `public ConceptType` **getSignature( )**
  - **Usage**
    - Returns the signature of this definition.
  - **Returns** - the signature of this definition.

2.2.10 CLASS ConceptTypeHierarchy

The concept type hierarchy class.

DECLARATION

public class ConceptTypeHierarchy
extends notio.TypeHierarchy
implements java.io.Serializable

FIELDS

- public static final String UNIVERSAL_TYPE_LABEL
  - The predefined type label for the universal type.
- public static final String ABSURD_TYPE_LABEL
  - The predefined type label for the absurd type.

CONSTRUCTORS

- *ConceptTypeHierarchy*
  `public` **ConceptTypeHierarchy**`( )`
  - **Usage**
    - Constructs a new concept type hierarchy.

METHODS

- *addSubTypesToType*
  `public void` **addSubTypesToType**`( notio.ConceptType` **subjectType**`, notio.ConceptType [] newSubTypes )`
  - **Usage**
    - Adds the specified list of subtypes as sub types to the subject type.
  - **Parameters**
    - `subjectType` - the type having subtypes added.
    - `newSubTypes` - the array of types to be added as subtypes.
  - **Exceptions**
    - `notio.TypeChangeError` - if specified subtypes are not in hierarchy, or if addition of subtypes creates a type order conflict.

- *addSubTypeToType*
  `public void` **addSubTypeToType**`( notio.ConceptType` **subjectType**`, notio.ConceptType` **newSubType** `)`
  - **Usage**
    - Adds the specified subtype to the subject type.
  - **Parameters**
    - `subjectType` - the type having a subtype added.
    - `newSubType` - the subtype to be added.
  - **Exceptions**
    - `notio.TypeChangeError` - if specified subtype is not in hierarchy or, if addition of the subtype creates a type order conflict.

- *addSuperTypesToType*
  `public void` **addSuperTypesToType**`( notio.ConceptType` **subjectType**`, notio.ConceptType [] newSuperTypes )`
  - **Usage**
    - Adds the specified list of supertypes as super types to the subject type.
  - **Parameters**
    - `subjectType` - the type having supertypes added.
    - `newSuperTypes` - the array of types to be added as supertypes.
  - **Exceptions**
    - `notio.TypeChangeError` - if specified supertypes are not in hierarchy, or if addition of supertypes creates a type order conflict.

- *addSuperTypeToType*
  `public void` **addSuperTypeToType(** `notio.ConceptType` **subjectType,** `notio.ConceptType` **newSuperType)**
  - **Usage**
    - Adds the specified supertype to the subject type.
  - **Parameters**
    - `subjectType` - the type having a supertype added.
    - `newSuperType` - the supertype to be added.
  - **Exceptions**
    - `notio.TypeChangeError` - if specified supertype is not in hierarchy or, if addition of the supertype creates a type order conflict.

- *addTypeToHierarchy*
  `public void` **addTypeToHierarchy(** `notio.ConceptType` **newType)**
  - **Usage**
    - Adds a new type to this hierarchy with the Universal type as its only supertype, and the Absurd type as its only subtype. Note that unlike the other versions of this method, this does not throw notio.TypeChangeError since the Universal and Absurd types are always present in the hierarchy.
  - **Parameters**
    - `newType` - the type being added.
  - **Exceptions**
    - `notio.TypeAddError` - if type label is already in use by another type.

- *addTypeToHierarchy*
  `public void` **addTypeToHierarchy(** `notio.ConceptType` **newType,** `notio.ConceptType []` **supertypes,** `notio.ConceptType []` **subtypes)**
  - **Usage**
    - Adds a new type to the hierarchy with the specified supertypes and subtypes.
  - **Parameters**
    - `newType` - the type being added.
    - `supertypes` - the array of supertypes to be used, or null (assumes universal type as only supertype).
    - `subtypes` - the array of subtypes to be used, or null (assumes absurd type as only subtype).
  - **Exceptions**
    - `notio.TypeAddError` - if type label is already in use by another type.
    - `notio.TypeChangeError` - if specified supertypes or subtypes are not in hierarchy, or if addition of type creates a type order conflict.

- *addTypeToHierarchy*
  `public void` **addTypeToHierarchy(** `notio.ConceptType` **newType,** `notio.ConceptType` **supertype,** `notio.ConceptType` **subtype)**
  - **Usage**
    - Adds a new type to this hierarchy with the specified supertype and subtype.
  - **Parameters**

    * `newType` - the type being added.
    * `supertype` - the single supertype to be used, or null (assumes universal type as subtype).
    * `subtype` - the single subtype to be used, or null (assumes absurd type as subtype).
  - **Exceptions**
    * `notio.TypeAddError` - if type label is already in use by another type.
    * `notio.TypeChangeError` - if specified supertypes or subtypes are not in hierarchy, or if addition of a type creates a type order conflict.

- *getCaseSensitiveLabels*
  `public boolean` **getCaseSensitiveLabels**`( )`
  - **Usage**
    * Returns true if the processing of type labels in this hierarchy is case-sensitive.
  - **Returns** - true if the processing of type labels in this hierarchy is case-sensitive.

- *getImmediateSubTypesOf*
  `public ConceptType` **getImmediateSubTypesOf**`( notio.ConceptType` **subjectType** `)`
  - **Usage**
    * Returns the immediate subtypes of the subject type in no particular order.
  - **Parameters**
    * `subjectType` - the type whose immediate subtypes will be returned.
  - **Returns** - an array of the immediate subtypes of the subject type, possibly empty.

- *getImmediateSuperTypesOf*
  `public ConceptType` **getImmediateSuperTypesOf**`( notio.ConceptType` **subjectType** `)`
  - **Usage**
    * Returns the immediate subtypes of the subject type in no particular order.
  - **Parameters**
    * `subjectType` - the type whose immediate subtypes will be returned.
  - **Returns** - an array of the immediate subtypes of the subject type, possibly empty.

- *getProperSubTypesOf*
  `public ConceptType` **getProperSubTypesOf**`( notio.ConceptType` **subjectType** `)`
  - **Usage**
    * Returns all the subtypes of the subject type, not just the immediate subtypes, in no particular order.
  - **Parameters**
    * `subjectType` - the type whose subtypes will be returned.
  - **Returns** - an array of the subtypes of the subject type, possibly empty.

- *getProperSuperTypesOf*
  `public ConceptType` **getProperSuperTypesOf**`( notio.ConceptType` **subjectType** `)`
  - **Usage**

    * Returns all the supertypes of the subject type, not just the immediate supertypes, in no particular order.
  - **Parameters**
    * `subjectType` - the type whose supertypes will be returned.
  - **Returns** - an array of the supertypes of the parent type, possibly empty.

- *getTypeByLabel*
  `public ConceptType` **getTypeByLabel**`( java.lang.String` **label** `)`
  - **Usage**
    * Looks up and returns the type object associated with the specified label.
  - **Parameters**
    * `label` - the label used to lookup.
  - **Returns** - the type associated with the label or null if non-existant.

- *getUnlabelledTypes*
  `public ConceptType` **getUnlabelledTypes**`( )`
  - **Usage**
    * Returns all unlabelled types in this hierarchy in no particular order.
  - **Returns** - an array of the unlabelled types in this hierarchy, possibly empty.

- *isProperSubTypeOf*
  `public boolean` **isProperSubTypeOf**`( notio.ConceptType` **subject**`, notio.ConceptType` **object** `)`
  - **Usage**
    * Determines whether subject is a subtype of object.
  - **Parameters**
    * `subject` - the potential subtype being tested.
    * `object` - the potential supertype being tested.
  - **Returns** - true if subject is a subtype of object.

- *isProperSuperTypeOf*
  `public boolean` **isProperSuperTypeOf**`( notio.ConceptType` **subject**`, notio.ConceptType` **object** `)`
  - **Usage**
    * Determines whether subject is a proper supertype of object.
  - **Parameters**
    * `subject` - the potential supertype being tested.
    * `object` - the potential subtype being tested.
  - **Returns** - true if subject is a supertype of object.

- *isSubTypeOf*
  `public boolean` **isSubTypeOf**`( notio.ConceptType` **subject**`, notio.ConceptType` **object** `)`
  - **Usage**
    * Determines whether subject is a subtype of object.

- **Parameters**
  - `subject` - the potential subtype being tested.
  - `object` - the potential supertype being tested.
- **Returns** - true if subject is a subtype of object.

- *isSuperTypeOf*
  `public boolean` **isSuperTypeOf**`( notio.ConceptType` **subject**`, notio.ConceptType` **object** `)`
  - **Usage**
    - Determines whether subject is a supertype of object.
  - **Parameters**
    - `subject` - the potential supertype being tested.
    - `object` - the potential subtype being tested.
  - **Returns** - true if subject is a supertype of object.

- *removeTypeFromHierarchy*
  `public void` **removeTypeFromHierarchy**`( notio.ConceptType` **deadType** `)`
  - **Usage**
    - Removes a type from this hierarchy. Removing a type from its hierarchy does not remove it from any of the nodes in which it may be used. This must be done manually if it is necessary. In most cases it is expected that types are being replaced rather than actually removed entirely. In this case, it is much easier to simply the change the characteristics of the existing type. If a type is being replaced by more than one type, it will naturally be necessary to perform at least part of the replacement manually. If replacements must be performed, the utility method replaceTypeInGraph() may be used.
  - **Parameters**
    - `deadType` - the type being removed.
  - **Exceptions**
    - `notio.TypeRemoveError` - if specified type is not in hierarchy
  - **See Also**
    - `notio.ConceptTypeHierarchy.replaceTypeInGraph`

- *replaceTypeInGraph*
  `public static void` **replaceTypeInGraph**`( notio.Graph` **graph**`, notio.ConceptType` **oldType**`, notio.ConceptType` **newType** `)`
  - **Usage**
    - Traverses the specified graph, replacing all occurrences of the specified type with a replacement type. Either of these types may be null. If the old type is null, untyped concepts will be set to the new type. If the new type is null, concepts using the old type will become untyped. In most cases where a non-null type is replacing a non-null type, it is probably more efficient to modify the old type directly and avoid the use of this method.

- *setCaseSensitiveLabels*
  `public void` **setCaseSensitiveLabels**`( boolean` **flag** `)`
  - **Usage**

  - Sets a flag indicating whether or not the processing of type labels within this hierarchy is case-sensitive.
- **Parameters**
  - `flag` - the flag setting for case-sensitivity.

METHODS INHERITED FROM CLASS `notio.TypeHierarchy`

2.2.11 CLASS CopyingScheme

A class used to specify how copying of graph elements should be performed. When copying graphs or graph components, a CopyingScheme instance is used to describe exactly how copying should be performed. Not all copying schemes need be implemented by a given implementation and many schemes would not make sense. The exact behaviour of an implementation under these circumstances is undefined but it is recommended that the implementation throw notio.UnimplementedFeatureException either when an invalid scheme is constructed or when it is used in a copying method. CopyingScheme instances may be reused as they are not altered by the copying process.

DECLARATION

public class CopyingScheme
extends java.lang.Object

FIELDS

- public static final int GR_COPY_DUPLICATE
  - Graph copy control flag: a duplicate graph will be created.
- public static final int GR_COPY_REFERENCE
  - Graph copy control flag: a reference to the existing graph will be used.
- public static final int CN_COPY_DUPLICATE
  - Concept copy control flag: a duplicate node will be created.
- public static final int CN_COPY_REFERENCE
  - Concept copy control flag: a reference to the existing node will be used.
- public static final int RN_COPY_DUPLICATE
  - Relation copy control flag: a duplicate node will be created.
- public static final int RN_COPY_REFERENCE
  - Relation copy control flag: a reference to the existing node will be used.
- public static final int DG_COPY_DUPLICATE
  - Designator copy control flag: a duplicate designator will be created.

- public static final int DG_COPY_REFERENCE
  - Designator copy control flag: a reference to the existing designator will be used.
- public static final int COMM_COPY_OFF
  - Comment copy control flag: node and graph comments will not be copied.
- public static final int COMM_COPY_ON
  - Comment copy control flag: node and graph comments will be copied.

Constructors

- *CopyingScheme*
  `public` **CopyingScheme**`( int` **newGraphFlag**`, int` **newConceptFlag**`, int` **newRelationFlag**`, int` **newDesignatorFlag**`, int` **newCommentFlag**`, notio.CopyingScheme` **newNestedScheme** `)`
  - **Usage**
    - Constructs a copying scheme with the specified control flags.
  - **Parameters**
    - `newGraphFlag` - Copying flag for graphs.
    - `newConceptFlag` - Copying flag for concepts.
    - `newRelationFlag` - Copying flag for relations.
    - `newDesignatorFlag` - Copying flag for designators.
    - `newCommentFlag` - Copying flag for comments.
    - `newNestedScheme` - A nested copying scheme to be used for copying nested graphs (null means use present scheme).

Methods

- *getCommentFlag*
  `public int` **getCommentFlag**`( )`
  - **Usage**
    - Returns the comment copying control flag for this scheme.
  - **Returns** - the comment copying control flag for this scheme.

- *getConceptFlag*
  `public int` **getConceptFlag**`( )`
  - **Usage**
    - Returns the concept copying control flag for this scheme.
  - **Returns** - the concept copying control flag for this scheme.

- *getDesignatorFlag*
  `public int` **getDesignatorFlag**`( )`
  - **Usage**
    - Returns the designator copying control flag for this scheme.
  - **Returns** - the designator copying control flag for this scheme.

- *getGraphFlag*
  `public int` **getGraphFlag( )**
  - **Usage**
    - Returns the graph copying control flag for this scheme.
  - **Returns** - the graph copying control flag for this scheme.

- *getNestedCopyingScheme*
  `public CopyingScheme` **getNestedCopyingScheme( )**
  - **Usage**
    - Returns the nested copying scheme or null. The nested copying scheme is used for copying nested graphs. If it is set to null, the current scheme is used.
  - **Returns** - the nested copying scheme or null.

- *getRelationFlag*
  `public int` **getRelationFlag( )**
  - **Usage**
    - Returns the relation copying control flag for this scheme.
  - **Returns** - the relation copying control flag for this scheme.

2.2.12 Class CorefAddException

Exception thrown when the addition of a coreference gives rise to an error.

Declaration

public class CorefAddException
extends notio.OperationException

Constructors

- *CorefAddException*
  `public` **CorefAddException(** `java.lang.String` **message)**
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.

- *CorefAddException*
  `public` **CorefAddException(** `java.lang.String` **message,** `java.lang.Throwable` **newSubThrowable)**
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.

- **Parameters**
  - `message` - the details of the exception.
  - `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.13 Class CoreferenceSet

Class to implement coreference sets (also known as lines of identity). Note that it is necessary to add at least one valid dominant concept to this set before adding subordinate concepts that may belong to other coreference sets.

Declaration

public class CoreferenceSet
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private Set corefSet
  - Set of coreferent concepts.
- private Concept definingConcept
  - The defining concept of the set.
- private String definingLabel
  - The defining label of the set included as an aid to translators.
- private Graph dominantGraph
  - The graph containing the dominant nodes.
- private boolean checkScope
  - A flag indicating whether or not coreference scope is checked.

Constructors

- *CoreferenceSet*
  `public` **CoreferenceSet**`( )`
  - **Usage**
    - Constructs a new, empty coreference set.

- *CoreferenceSet*
  `public` **CoreferenceSet**`( java.lang.String` **newDefLabel** `)`
  - **Usage**
    - Constructs a new, empty coreference set with the specified defining label. The defining label is not an essential part of the coreference set and is not required (it can be null). It is included to make parsing and generating easier.
  - **Parameters**
    - `newDefLabel` - the defining label for this coreference set.

METHODS

- *addCoreferentConcept*
  `public void` **addCoreferentConcept(** `notio.Concept` **newConcept)**
  - **Usage**
    - Adds a concept to this coreference set.
  - **Parameters**
    - `newConcept` - the concept to be added to this coref set.
  - **Exceptions**
    - `notio.CorefAddException` - if the specified concept is dominant in another coreference set, or if the specified concept would be dominant in this set but is a member of other sets, or if the specified concept is not in the correct scope for this set, or if the specified concept is not enclosed by any graph.

- *getCoreferentConcepts*
  `public Concept` **getCoreferentConcepts( )**
  - **Usage**
    - Returns all of the concepts in this coreference set.
  - **Returns** - an array containing all the concepts in this coreference set.

- *getDefiningConcept*
  `public Concept` **getDefiningConcept( )**
  - **Usage**
    - Returns the defining concept associated with this coreference set or null if none was specified.
  - **Returns** - the defining concept associated with this coreference set.

- *getDefiningLabel*
  `public String` **getDefiningLabel( )**
  - **Usage**
    - Returns the defining label associated with this coreference set or null if none was specified.
  - **Returns** - the defining label associated with this coreference set.

- *getDominantConcepts*
  `public Concept` **getDominantConcepts( )**
  - **Usage**
    - Returns the dominant concepts in this coreference set. Note that the return value of this method is undefined if scope checking is disabled.
  - **Returns** - an array containing the dominant concepts in this coreference set.

- *getEnableScopeChecking*
  `public boolean` **getEnableScopeChecking( )**
  - **Usage**

    - Returns the current value of the flag which enables or disables scope checking (disabled by default) when modifying this coreference set.
  - **Returns** - the current setting for the scope checking flag.

- *getSubordinateConcepts*
  `public Concept` **getSubordinateConcepts( )**
  - **Usage**
    - Returns the subordinate concepts in this coreference set. Note that this array may be empty. Note that the return value of this method is undefined if scope checking is disabled.
  - **Returns** - an array containing the subordinate concepts in this coreference set, possibly empty.

- *hasConcept*
  `public boolean` **hasConcept(** `notio.Concept` **concept)**
  - **Usage**
    - Tests whether the specified concept is a member of this coref set whether dominant or subordinate.
  - **Parameters**
    - `concept` - the concept being tested.
  - **Returns** - true if the specified concept is a member of this coref set.

- *hasDominantConcept*
  `public boolean` **hasDominantConcept(** `notio.Concept` **concept)**
  - **Usage**
    - Tests whether the specified concept is a dominant concept in this coreference set. Note that the return value of this method is undefined if scope checking is disabled.
  - **Parameters**
    - `concept` - the concept being tested to see if it a dominant concept in this set.
  - **Returns** - true if the specified concept is a dominant concept of this coreference set.

- *hasSubordinateConcept*
  `public boolean` **hasSubordinateConcept(** `notio.Concept` **concept)**
  - **Usage**
    - Tests whether the specified concept is a subordinate concept of this coref set. Note that the return value of this method is undefined if scope checking is disabled.
  - **Parameters**
    - `concept` - the concept being tested.
  - **Returns** - true if the specified concept is a subordinate concept of this coref set.

- *removeConcepts*
  `public void` **removeConcepts(** `notio.Concept []` **deadConcepts)**
  - **Usage**
    - Removes the specified concepts from this coreference set. Note that this method will automatically clear the current defining concept if it is removed from the set.

- **Parameters**
  - `deadConcepts` - the array of concepts to be removed from this coref set.
- **Exceptions**
  - `notio.CorefRemoveException` - if removal of the specified concepts would result in an invalid coreference set.

- *removeCoreferentConcept*
  `public void` **removeCoreferentConcept**`( notio.Concept` **deadConcept** `)`
  - **Usage**
    - Removes the specified concept from this coreference set. Note that this method will automatically clear the current defining concept if it is removed from the set.
  - **Parameters**
    - `deadConcept` - the concept to be removed from this coref set.
  - **Exceptions**
    - `notio.CorefRemoveException` - if removal of the specified concept would result in an invalid coreference set.

- *setDefiningConcept*
  `public void` **setDefiningConcept**`( notio.Concept` **newDefConcept** `)`
  - **Usage**
    - Sets the defining concept associated with this coreference set. This is the concept directly associated with the defining label. The primary use for this feature is related to parsing and generating, particulary as regards constructing arcs for relations. The specified concept must already be a dominant member of the coreference set.
  - **Parameters**
    - `newDefConcept` - the defining concept to be associated with this coreference set.
  - **Exceptions**
    - `notio.InvalidDefiningConceptException` - if the specified concept is not a dominant concept in this set.

- *setDefiningLabel*
  `public void` **setDefiningLabel**`( java.lang.String` **newDefLabel** `)`
  - **Usage**
    - Sets the defining label associated with this coreference set. No check is made to see if this label is in use by any other set so incautious use of this method may confuse translators. There is really little reason to change the coreference label of a set once established but the method is included for the sake of completeness.
  - **Parameters**
    - `newDefLabel` - the defining label to be associated with this coreference set.

- *setEnableScopeChecking*
  `public void` **setEnableScopeChecking**`( boolean` **flag** `)`
  - **Usage**
    - Sets a flag which enables or disables scope checking (enabled by default) when modifying this coreference set.

  - **Parameters**
    * `flag` - the new setting for the scope checking flag.
  - **Exceptions**
    * `notio.CorefAddException` - if any of the concepts in the set have no enclosing graph, or if any of the concepts is out of scope, or if any of the domintant concepts belongs to one or more other coreference sets.
    * `notio.InvalidDefiningConceptException` - if a defining concept has been specified and is not dominant.

- *size*
  `public int size( )`
  - **Usage**
    * Returns the number of concepts in this coreference set.
  - **Returns** - the number of concepts in this coreference set.

2.2.14 Class CorefRemoveException

Exception thrown when the removal of a concept from a coreference set gives rise to an error.

Declaration

public class CorefRemoveException **extends** notio.OperationException

Constructors

- *CorefRemoveException*
  `public` **CorefRemoveException(** `java.lang.String` **message)**
  - **Usage**
    * Constructs an exception with the specified message.
  - **Parameters**
    * `message` - The details of the exception.

- *CorefRemoveException*
  `public` **CorefRemoveException(** `java.lang.String` **message,**
  `java.lang.Throwable` **newSubThrowable)**
  - **Usage**
    * Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    * `message` - the details of the exception.
    * `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.15 CLASS DefinedDesignator

Class for defined designators. This class is essentially a dummy since it's not clear what it should do.

DECLARATION

public class DefinedDesignator
extends notio.Designator
implements java.io.Serializable

CONSTRUCTORS

- *DefinedDesignator*
  `public` **DefinedDesignator( )**
  - **Usage**
    - Constructs a new DefinedDesignator.

METHODS

- *copy*
  `public Designator` **copy(** `notio.CopyingScheme` **copyScheme,**
  `java.util.Hashtable` **substitutionTable)**
  - **Usage**
    - Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
    - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getDesignatorKind*
  `public int` **getDesignatorKind( )**
  - **Usage**
    - Returns a constant indicating which kind of designator is. In this case the constant will be: Designator.DESIGNATOR_DEFINED
  - **Returns** - a constant indicating the kind of the designator.

METHODS INHERITED FROM CLASS `notio.Designator`

(in 2.2.16, page 92)

- *copy*
  `public abstract Designator` **copy(** `notio.CopyingScheme` **copyScheme,**
  `java.util.Hashtable` **substitutionTable)**
  - **Usage**

    * Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.
  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
    * `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getCaseSensitiveLabels*
`public boolean` **getCaseSensitiveLabels**`( )`
  - **Usage**
    * Returns true if the processing of labels in this designator is case-sensitive.
  - **Returns** - true if the processing of labels in this designator is case-sensitive.

- *getDesignatorKind*
`public abstract int` **getDesignatorKind**`( )`
  - **Usage**
    * Returns a constant indicating which kind of designator this is. The constant is one of: Designator.DESIGNATOR_LITERAL Designator.DESIGNATOR_MARKER Designator.DESIGNATOR_DEFINED Designator.DESIGNATOR_NAME
  - **Returns** - a constant indicating the kind of the designator.

- *getEnclosingReferent*
`public Referent` **getEnclosingReferent**`( )`
  - **Usage**
    * Returns the enclosing referent for this designator or null if there isn't one.
  - **Returns** - the enclosing referent for this designator or null.

- *matchDesignators*
`public static MatchResult` **matchDesignators**`( notio.Designator` **first**`, notio.Designator` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    * Compares two designators to decide if they match. The exact semantics of matching are determined by the match control flag.
  - **Parameters**
    * `first` - the first designators being matched.
    * `second` - the second designators being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two designators match according to the scheme's criteria.

- *setCaseSensitiveLabels*
`public void` **setCaseSensitiveLabels**`( boolean` **flag** `)`
  - **Usage**
    * Sets a flag indicating whether or not the processing of labels within this designator is case-sensitive.
  - **Parameters**
    * `flag` - the flag setting for case-sensitivity.

2.2.16 Class Designator

Base class for designators.

Declaration

public abstract class Designator
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private boolean caseSensitiveLabels
  - Flag indicating whether labels (e.g. names) are case sensitive or not.
- private Referent enclosingReferent
  - The referent to which this designator belongs.

Fields

- public static final int DESIGNATOR_LITERAL
  - Indicates that a designator is a LiteralDesignator.
- public static final int DESIGNATOR_MARKER
  - Indicates that a designator is a LocatorDesignator.
- public static final int DESIGNATOR_DEFINED
  - Indicates that a designator is a DefinedDesignator.
- public static final int DESIGNATOR_NAME
  - Indicates that a designator is a NameDesignator.

Constructors

- *Designator*
  `public` **Designator( )**

Methods

- *copy*
  `public abstract Designator` **copy(** `notio.CopyingScheme` **copyScheme,**
  `java.util.Hashtable` **substitutionTable)**
  - **Usage**
    - Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.
  - **Parameters**

    * `copyScheme` - the copying scheme used to control the copy operation.
    * `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getCaseSensitiveLabels*
  `public boolean` **getCaseSensitiveLabels**`( )`
  - **Usage**
    * Returns true if the processing of labels in this designator is case-sensitive.
  - **Returns** - true if the processing of labels in this designator is case-sensitive.

- *getDesignatorKind*
  `public abstract int` **getDesignatorKind**`( )`
  - **Usage**
    * Returns a constant indicating which kind of designator this is. The constant is one of: Designator.DESIGNATOR_LITERAL Designator.DESIGNATOR_MARKER Designator.DESIGNATOR_DEFINED Designator.DESIGNATOR_NAME
  - **Returns** - a constant indicating the kind of the designator.

- *getEnclosingReferent*
  `public Referent` **getEnclosingReferent**`( )`
  - **Usage**
    * Returns the enclosing referent for this designator or null if there isn't one.
  - **Returns** - the enclosing referent for this designator or null.

- *matchDesignators*
  `public static MatchResult` **matchDesignators**`( notio.Designator` **first**`, notio.Designator` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    * Compares two designators to decide if they match. The exact semantics of matching are determined by the match control flag.
  - **Parameters**
    * `first` - the first designators being matched.
    * `second` - the second designators being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two designators match according to the scheme's criteria.

- *setCaseSensitiveLabels*
  `public void` **setCaseSensitiveLabels**`( boolean` **flag** `)`
  - **Usage**
    * Sets a flag indicating whether or not the processing of labels within this designator is case-sensitive.
  - **Parameters**
    * `flag` - the flag setting for case-sensitivity.

2.2.17 Class Extensions

Class which may be queried to verify that specific, independant extensions to the API are available and also to activate or deactivate any extended features. Note that all methods in this class are static and specific instances of it are not needed.

Declaration

public class Extensions
extends java.lang.Object

Constructors

- *Extensions*
  `public` **Extensions( )**

Methods

- *activateExtension*
  `public static void` **activateExtension(** `java.lang.String` **extensionName,** `java.lang.Object [] args` **)**
  - **Usage**
    - Used to activate or configure API extensions. Obviously, not all extensions will be optional or require configuration. If an extension does not require these services then the implementation can simply ignore calls to this method with regards to that extension.
  - **Parameters**
    - `extensionName` - the extension being activated/configured
    - `args` - an array of objects used to configure the extension if necessary.

- *deactivateExtension*
  `public static void` **deactivateExtension(** `java.lang.String` **extensionName,** `java.lang.Object [] args` **)**
  - **Usage**
    - Used to deactivate API extensions. Obviously, not all extensions will be optional. If an extension does not require these services then the implementation can simply ignore calls to this method with regards to that extension. The arguments are provided for deactivation in case any details need to be specified about the deactivation process.
  - **Parameters**
    - `extensionName` - the extension being activated/configured
    - `args` - an array of objects used to configure the extension if necessary.

- *getExtensionParameters*
  `public static Object` **getExtensionParameters(** `java.lang.String` **extensionName)**
  - **Usage**
    - Used to query the current configuration of an extension. Not all extensions will have configuration information. The result of calling this method for such extensions is undefined.
  - **Parameters**
    - `extensionName` - the extension being looked for.
  - **Returns** - an array of Objects containing the desired information, possibly empty, or null.

- *isExtensionActive*
  `public static boolean` **isExtensionActive(** `java.lang.String` **extensionName)**
  - **Usage**
    - Used to check if a particular extension is 'active'. Not all extensions will have an active or inactive status. The results for calling this method for such extensions is undefined.
  - **Parameters**
    - `extensionName` - the extension being looked for.
  - **Returns** - true if the extension is active.

- *isExtensionAvailable*
  `public static boolean` **isExtensionAvailable(** `java.lang.String` **extensionName)**
  - **Usage**
    - Used to check if a particular extension is available.
  - **Parameters**
    - `extensionName` - the extension being looked for.
  - **Returns** - true if the extension is available.

- *listAvailableExtensions*
  `public static String` **listAvailableExtensions( )**
  - **Usage**
    - Returns an array of strings that contains the names of all available extensions. Extension names should look something like: EXTENSION_THREAD_SAFETY
  - **Returns** - an array of strings containing the names of available extensions, possibly empty, or null.

2.2.18 Class GeneratorException

Exception thrown when some parser's operation gives rise to an error.

DECLARATION

public class GeneratorException
extends notio.TranslationException

CONSTRUCTORS

- *GeneratorException*
  `public` **GeneratorException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.

- *GeneratorException*
  `public` **GeneratorException**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.
    - `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.TranslationException`

(in 2.2.45, page 179)

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.

– **Parameters**
  * `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.19 CLASS Graph

The basic conceptual graph class.

DECLARATION

public class Graph
extends java.lang.Object
implements java.io.Serializable

SERIALIZABLE FIELDS

- private Vector concepts
  – The concepts in this graph.
- private Vector relations
  – The relations in this graph.
- private Vector comments
  – The comments on this graph.
- private boolean allowIncompleteRelations

  - Flag indicating whether incomplete Relation nodes may be added.

- private int contextDepth
  - An integer that specifies the current context depth of this graph.

- private Referent enclosingReferent
  - The Referent that encloses this graph, if any.

CONSTRUCTORS

- *Graph*
  `public` **Graph**`( )`
  - **Usage**
    * Constructs an empty graph.

METHODS

- *addComment*
  `public void` **addComment**`( java.lang.String` **newComment** `)`
  - **Usage**
    * Adds a comment to this graph. The comment has no strict interpretation but may be used be applications or humans to discover extra information about this graph.
  - **Parameters**
    * `newComment` - the comment to be added to this graph.

- *addConcept*
  `public void` **addConcept**`( notio.Concept` **newConcept** `)`
  - **Usage**
    * Adds a concept to this graph. If the concept is already part of the graph, nothing is changed and no exception is thrown.
  - **Parameters**
    * `newConcept` - the concept to be added.
  - **Exceptions**
    * `notio.ConceptAddError` - if the specified concept is already part of another graph.

- *addConcepts*
  `public void` **addConcepts**`( notio.Concept [] newConcepts )`
  - **Usage**
    * Adds one or more concepts to this graph. Any nulls in the array of concepts will be ignored.
  - **Parameters**
    * `newConcepts` - the array of concepts to be added.
  - **Exceptions**
    * `notio.ConceptAddError` - if one of the concepts is already part of another graph. All concepts before the offending concept are still added.

- *addRelation*
  `public void` **addRelation**`( notio.Relation` **newRelation** `)`
  - **Usage**
    - Adds a relation to this graph and adds any related concepts that are not already in this graph at the same time.
  - **Parameters**
    - `newRelation` - the relation to be added.
  - **Exceptions**
    - `notio.RelationAddError` - if newRelation already belongs to another graph, or if newRelation is incomplete and this graph does not allow incomplete relations.
    - `notio.ConceptAddError` - if an error is thrown whilst adding one of arguments.
  - **See Also**
    - `notio.Graph.getAllowIncompleteRelations()`

- *addRelations*
  `public void` **addRelations**`( notio.Relation [] newRelations )`
  - **Usage**
    - Adds one or more relations to this graph and adds any related concepts that are not already in this graph at the same time. Any nulls in the array of relations will be ignored.
  - **Parameters**
    - `newRelations` - the array of relations to be added.
  - **Exceptions**
    - `notio.RelationAddError` - if one of the relations already belongs to another graph. All additions up to the one that causes the error are still performed.
    - `notio.ConceptAddError` - if an error is thrown whilst adding one of arguments.

- *copy*
  `public Graph` **copy**`( notio.CopyingScheme` **copyScheme** `)`
  - **Usage**
    - Performs a copy operation on this graph according to the the specified CopyingScheme. This method will always produce a new Graph object and only nested graphs will be affect by the value of MatchingScheme.getGraphFlag().
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
  - **Returns** - the result of the copy operation.

- *copy*
  `public Graph` **copy**`( notio.CopyingScheme` **copyScheme**`, java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    - Performs a copy operation on this graph according to the the specified CopyingScheme. This method will always produce a new object of the exact same type as the original and only nested graphs will be affected by the value of MatchingScheme.getGraphFlag(). Note that the flag returned by Graph.getAllowIncompleteRelation() will be copied to the new graph.

- **Parameters**
  - `copyScheme` - the copying scheme used to control the copy operation.
  - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
- **Returns** - the result of the copy operation.
- **See Also**
  - `notio.Graph.getAllowIncompleteRelations` (in 2.2.19, page 102)

- *expandConceptType*
  `public void` **expandConceptType(** `notio.Concept` **concept)**
  - **Usage**
    - Performs minimal type expansion on the specified concept if a type definition is available for that type.
  - **Parameters**
    - `concept` - the concept which is to undergo type expansion.

- *expandConceptType*
  `public void` **expandConceptType(** `notio.Concept []` **concepts)**
  - **Usage**
    - Performs minimal type expansion on the specified concepts if type definitions is available for their types.
  - **Parameters**
    - `concepts` - the array of concepts to undergo type expansion.

- *expandConceptType*
  `public void` **expandConceptType(** `notio.ConceptType` **conceptType)**
  - **Usage**
    - Performs minimal type expansion on all concepts in this graph of the specified type, if a type definition is available for that type.
  - **Parameters**
    - `conceptType` - the concept type to expand.

- *expandRelationType*
  `public void` **expandRelationType(** `notio.Relation` **relation)**
  - **Usage**
    - Performs minimal type expansion on the specified relation if a type definition is available for that type.
  - **Parameters**
    - `relation` - the relation which is to undergo type expansion.

- *expandRelationType*
  `public void` **expandRelationType(** `notio.Relation []` **relations)**
  - **Usage**
    - Performs minimal type expansion on the specified relations if type definitions is available for their types.

- **Parameters**
  - `relations` - the array of relations to undergo type expansion.

- *expandRelationType*
  `public void` **expandRelationType(** `notio.RelationType` **relationType)**
  - **Usage**
    - Performs minimal type expansion on all relations in this graph of the specified type, if a type definition is available for that type.
  - **Parameters**
    - `relationType` - the relation type to expand.

- *getActorRelations*
  `public Actor` **getActorRelations( )**
  - **Usage**
    - Returns a list of the those relations in this graph that are actors. No guarantee is made as to the ordering of the actors in the array. Subsequent calls to this method may produce different orderings.
  - **Returns** - an array containing all actors in this graph.

- *getAllowIncompleteRelations*
  `public boolean` **getAllowIncompleteRelations( )**
  - **Usage**
    - Returns the flag setting for this graph indicating whether or not incomplete relations and actors may be added to this graph.
  - **Returns** - the flag setting for this graph.

- *getComments*
  `public String` **getComments( )**
  - **Usage**
    - Returns an array of all comments associated with this graph.
  - **Returns** - an array of Strings (possibly empty) that are all the comments associated with this graph.

- *getConcepts*
  `public Concept` **getConcepts( )**
  - **Usage**
    - Returns a list of the concepts in this graph. No guarantee is made as to the ordering of the concepts in the array. Subsequent calls to this method may produce different orderings.
  - **Returns** - an array containing all concepts in this graph.

- *getConceptsWithExactType*
  `public Concept` **getConceptsWithExactType(** `notio.ConceptType` **conType)**
  - **Usage**

    * Returns all concepts in the graph with exactly the type specified. Subtypes will not be included.
  - **Parameters**
    * `conType` - the concept type to be matched exactly.
  - **Returns** - an array of all concepts of the exactly the specified type or null if no concepts of the specified type are present in the graph.

- *getConceptsWithSubType*
  `public Concept` **getConceptsWithSubType(** `notio.ConceptType` **conType)**
  - **Usage**
    * Returns all concepts in the graph that are supertypes of the specified type (which includes the type itself).
  - **Parameters**
    * `conType` - the concept type whose supertypes will be matched.
  - **Returns** - an array of all concepts that are supertypes of the specified type.

- *getConceptsWithSuperType*
  `public Concept` **getConceptsWithSuperType(** `notio.ConceptType` **conType)**
  - **Usage**
    * Returns all concepts in the graph that are subtypes of the specified type (which includes the type itself).
  - **Parameters**
    * `conType` - the concept type whose subtypes will be matched.
  - **Returns** - an array of all concepts that are subtypes of the specified type.

- *getContext*
  `public Concept` **getContext(** **)**
  - **Usage**
    * Returns the context of this graph or null if this graph is in the outermost context. This may also be thought of as the enclosing concept of this graph, accessible via the enclosing referent. This method is provided for convenience since the enclosing concept of a graph is often of interest.
  - **Returns** - the concept that is the context of the graph or null if it is not enclosed by a concept.

- *getContextDepth*
  `public int` **getContextDepth(** **)**
  - **Usage**
    * Returns the context depth (level of nesting) of this graph. A depth of zero indicates that this graph is at the outermost level.
  - **Returns** - the context depth of this graph.

- *getContextGraph*
  `public Graph` **getContextGraph(** **)**
  - **Usage**

    * Returns the graph containing the context of this graph or null if this graph is not enclosed by a graph. This is equivalent to calling getEnclosingGraph() in this graph's context (if it has one). This method is provided for convenience since the enclosing graph of a graph is often of interest.
  - **Returns** - the concept that is the context of the graph or null if it is not enclosed by a graph.

- *getEnclosingReferent*
  `public Referent` **getEnclosingReferent**`( )`
  - **Usage**
    * Return the enclosing referent for this graph if it is nested or null otherwise.
  - **Returns** - returns the referent enclosing this graph or null if it is not enclosed by by a referent.

- *getMatchingConcepts*
  `public static Concept` **getMatchingConcepts**`( notio.Graph` **first**`, notio.Graph` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    * For each concept the first graph, this method finds what concepts match it from the second graph. Returns an array of concepts from the second graph that match the first graph's concepts.
  - **Parameters**
    * `first` - the first graph from which concepts are to be matched.
    * `second` - the second graph from which concepts are to be matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - an array of matching concepts from the second graph.

- *getMatchingRelations*
  `public static Relation` **getMatchingRelations**`( notio.Graph` **first**`, notio.Graph` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    * For each relation the first graph, this method finds what relations match it from the second graph. Returns an array of relations from the second graph that match the first graph's relations.
  - **Parameters**
    * `first` - the first graph from which relations are to be matched.
    * `second` - the second graph from which relations are to be matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - an array of matching relations from the second graph.

- *getNormalRelations*
  `public Relation` **getNormalRelations**`( )`
  - **Usage**
    * Returns a list of the those relations in this graph that are NOT actors. No guarantee is made as to the ordering of the relations in the array. Subsequent calls to this method may produce different orderings.

  - **Returns** - an array containing all non-actor relations in this graph.

- *getNumberOfConcepts*
  `public int` **getNumberOfConcepts( )**
  - **Usage**
    - Returns the number of concepts in this graph.
  - **Returns** - the number of concepts in this graph.

- *getNumberOfRelations*
  `public int` **getNumberOfRelations( )**
  - **Usage**
    - Returns the number of relations in this graph. This includes all relations including Actors rather than just Relations.
  - **Returns** - the number of relations in this graph.

- *getRelations*
  `public Relation` **getRelations( )**
  - **Usage**
    - Returns a list of the relations in this graph. This includes all relations, including Actors. No guarantee is made as to the ordering of the relations in the array. Subsequent calls to this method may produce different orderings.
  - **Returns** - an array containing all relations in this graph.

- *getRelationsWithExactType*
  `public Relation` **getRelationsWithExactType(** `notio.RelationType` **relType)**
  - **Usage**
    - Returns all relations in the graph with exactly the type specified. Subtypes will not be included.
  - **Parameters**
    - `relType` - the relation type to be matched exactly.
  - **Returns** - an array of all relations of the exactly the specified type or null if no relations of the specified type are present in the graph.

- *getRelationsWithSubType*
  `public Relation` **getRelationsWithSubType(** `notio.RelationType` **relType)**
  - **Usage**
    - Returns all relations in the graph that are supertypes of the specified type (which includes the type itself).
  - **Parameters**
    - `relType` - the relation type whose supertypes will be matched.
  - **Returns** - an array of all relations that are supertypes of the specified type.

- *getRelationsWithSuperType*
  `public Relation` **getRelationsWithSuperType(** `notio.RelationType` **relType)**
  - **Usage**

    * Returns all relations in the graph that are subtypes of the specified type (which includes the type itself).
  - **Parameters**
    * `relType` - the relation type whose subtypes will be matched.
  - **Returns** - an array of all relations that are subtypes of the specified type.

- *hasConcept*
  `public boolean` **hasConcept**`( notio.Concept` **concept** `)`
  - **Usage**
    * Returns true if the graph contains the specified concept.
  - **Parameters**
    * `concept` - the concept being checked for.
  - **Returns** - true if the specified concept is part of this graph.

- *hasConcepts*
  `public boolean` **hasConcepts**`( notio.Concept [] conceptArr )`
  - **Usage**
    * Returns true if the graph contains the specified concepts.
  - **Parameters**
    * `conceptArr` - the array of concepts being checked for.
  - **Returns** - true if the specified concepts are part of this graph.

- *hasRelation*
  `public boolean` **hasRelation**`( notio.Relation` **relation** `)`
  - **Usage**
    * Returns true if the graph contains the specified relation.
  - **Parameters**
    * `relation` - the relation being checked for.
  - **Returns** - true if the specified relation is part of this graph.

- *hasRelations*
  `public boolean` **hasRelations**`( notio.Relation [] relationArr )`
  - **Usage**
    * Returns true if the graph reltains the specified relations.
  - **Parameters**
    * `relationArr` - the array of relations being checked for.
  - **Returns** - true if the specified relations are part of this graph.

- *isBlank*
  `public boolean` **isBlank**`( )`
  - **Usage**
    * Returns true if this graph is blank. Blank means that there are no concepts, relations, or actors in this graph.
  - **Returns** - true if this graph is blank.

- *isComplete*
  `public boolean` **isComplete**`( )`
  - **Usage**
    - Returns true if this graph is complete. A complete graph is one in which all relations are complete.
  - **Returns** - true if this graph is complete.
  - **See Also**
    - `notio.Relation.isComplete()`

- *isEnclosedBy*
  `public boolean` **isEnclosedBy**`( notio.Concept` **concept** `)`
  - **Usage**
    - Returns true if this graph is enclosed by the specified concept.
  - **Parameters**
    - `concept` - the concept being checked for as enclosing this graph.
  - **Returns** - true if this graph is enclosed by the specified concept.

- *isEnclosedBy*
  `public boolean` **isEnclosedBy**`( notio.Graph` **graph** `)`
  - **Usage**
    - Returns true if this graph is enclosed by the specified graph.
  - **Parameters**
    - `graph` - the graph being checked for as enclosing this graph.
  - **Returns** - true if this graph is enclosed by the specified graph.

- *join*
  `public static Graph` **join**`( notio.Graph` **firstGraph**`, notio.Concept []` **firstConcepts**`, notio.Graph` **secondGraph**`, notio.Concept []` **secondConcepts**`, notio.MatchingScheme` **matchingScheme**`, notio.CopyingScheme` **copyScheme** `)`
  - **Usage**
    - Joins two graphs on the specified concept nodes using the specified matching scheme to determine if the nodes form a valid join point.
  - **Parameters**
    - `firstGraph` - the first graph to be joined.
    - `firstConcepts` - the joining concepts in the first graph.
    - `secondGraph` - the second graph to be joined.
    - `secondConcepts` - the joining concepts in the second graph.
    - `matchingScheme` - the matching scheme used to determine if the specified concepts form valid join points.
    - `copyScheme` - the copying scheme used to create the new graph from the two being joined.
  - **Returns** - the graph that results from the join.
  - **Exceptions**
    - `notio.JoinException` - if specified concepts are not part of the specified graphs and/or if the concepts do not form valid join points.

- *join*
  `public static Graph` **join(** `notio.Graph` **firstGraph,** `notio.Concept` **firstConcept,** `notio.Graph` **secondGraph,** `notio.Concept` **secondConcept,** `notio.MatchingScheme` **matchingScheme,** `notio.CopyingScheme` **copyScheme)**
  - **Usage**
    - Joins two graphs on the specified concept nodes using the specified matching scheme to determine if the nodes form a valid join point.
  - **Parameters**
    - `firstGraph` - the first graph to be joined.
    - `firstConcept` - the joining concept in the first graph.
    - `secondGraph` - the second graph to be joined.
    - `secondConcept` - the joining concept in the second graph.
    - `matchingScheme` - the matching scheme used to determine if the specified concepts are a valid join point.
    - `copyScheme` - the copying scheme used to create the new graph from the two being joined.
  - **Returns** - the graph that results from the join.
  - **Exceptions**
    - `notio.JoinException` - if specified concepts are not part of the specified graphs and/or if the concepts do not form a valid join point.

- *matchGraphs*
  `public static MatchResult` **matchGraphs(** `notio.Graph` **first,** `notio.Graph` **second,** `notio.MatchingScheme` **matchingScheme)**
  - **Usage**
    - Attempts a match between the two specified graphs and returns the results in a MatchResult object. Matching is governed by the specified MatchingScheme.
  - **Parameters**
    - `first` - the first graph being matched.
    - `second` - the second graph being matched.
    - `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - the MatchResult object describing the details of the match.

- *maximalJoin*
  `public static Graph` **maximalJoin(** `notio.Graph` **firstGraph,** `notio.Graph` **secondGraph,** `notio.MatchingScheme` **matchingScheme,** `notio.CopyingScheme` **copyScheme)**
  - **Usage**
    - Maximally joins two graphs using all possible mappings.
  - **Parameters**
    - `firstGraph` - the first graph to be joined.
    - `secondGraph` - the second graph to be joined.
    - `matchingScheme` - the matching scheme used to determine if the specified concepts form valid join points.
    - `copyScheme` - the copying scheme used to create the new graph from the two being joined.

- **Returns** - the graphs that form the set of all possible maximal joins or null if no join exists.
- **Exceptions**
  - `notio.JoinException` - if specified concepts are not part of the specified graphs and/or if the concepts do not form valid join points.

- *maximalJoin*
  `public static Graph` **maximalJoin(** `notio.Graph` **firstGraph,** `notio.Graph` **secondGraph,** `notio.MatchingScheme` **matchingScheme,** `notio.CopyingScheme` **copyScheme,** `int` **N)**
  - **Usage**
    - Maximally joins two graphs, using N or fewer node mappings, depending on what is possible. If the specified N less than 1, all possible mappings will be used.
  - **Parameters**
    - `firstGraph` - the first graph to be joined.
    - `secondGraph` - the second graph to be joined.
    - `matchingScheme` - the matching scheme used to determine if the specified concepts form valid join points.
    - `copyScheme` - the copying scheme used to create the new graph from the two being joined.
    - `N` - the maximum number of graphs that will be produced.
  - **Returns** - up to N graphs that are maximal joins or null if no join exists.
  - **Exceptions**
    - `notio.JoinException` - if specified concepts are not part of the specified graphs and/or if the concepts do not form valid join points.

- *removeComment*
  `public void` **removeComment(** `java.lang.String` **deadComment)**
  - **Usage**
    - Removes a comment from this graph. Removes the first occurrence of the specified comment from this graph. If the comment is not part of this graph, nothing happens.
  - **Parameters**
    - `deadComment` - the comment to be removed from this graph.

- *removeConcept*
  `public void` **removeConcept(** `notio.Concept` **deadConcept)**
  - **Usage**
    - Removes the specified concept from this graph. Any relations involving the concept must be removed before the concept may be removed. Any corefence links to this concept are also removed.
  - **Parameters**
    - `deadConcept` - the concept to be removed.
  - **Exceptions**
    - `notio.ConceptRemoveException` - if deadConcept is involved in a relation in this graph, or if deadConcept is not present in this graph, or if removal of deadConcept will result in an invalid coreference set.

- *removeConcepts*
  `public void` **removeConcepts**`( notio.Concept [] deadConcepts )`
  - **Usage**
    - Removes the specified concepts from this graph. Any nulls in the array will be ignored.
  - **Parameters**
    - `deadConcepts` - the array of concepts to be removed.
  - **Exceptions**
    - `notio.ConceptRemoveException` - if one of the concepts is part of a relation in this graph, or if one of the concepts is not present in this graph, or if one of the removals will result in an invalid coreference set. All removals up to the one that causes the exception are still performed.

- *removeRelation*
  `public void` **removeRelation**`( notio.Relation` **deadRelation** `)`
  - **Usage**
    - Removes the specified relation from this graph. If the relation is not part of this graph, nothing happens. Note: This does not remove any of the related concepts.
  - **Parameters**
    - `deadRelation` - the relation to be removed.

- *removeRelations*
  `public void` **removeRelations**`( notio.Relation [] deadRelations )`
  - **Usage**
    - Removes the specified relations from this graph. If a relation is not part of the graph, nothing happens and all other relations are removed as usual. Any nulls in the array will be ignored. Note: This does not remove any of the related concepts.
  - **Parameters**
    - `deadRelations` - the array of relations to be removed.

- *replaceConcept*
  `public void` **replaceConcept**`( notio.Concept` **oldConcept**`, notio.Concept` **newConcept** `)`
  - **Usage**
    - Replaces one concept with another in a graph. The new concept fills exactly the same role as the old concept with respect to relations and coreference sets.
  - **Parameters**
    - `oldConcept` - the old concept being replaced.
    - `newConcept` - the new concept that replaces the old.
  - **Exceptions**
    - `notio.ConceptReplaceException` - if oldConcept is not present in the graph.

- *setAllowIncompleteRelations*
  `public void` **setAllowIncompleteRelations**`( boolean` **flag** `)`
  - **Usage**

* Sets a flag for this graph indicating whether or not incomplete relations and actors may be added to this graph. An incomplete relation (or actor) is one which returns false when its isComplete() method is called. Note that changing this flag from true to false does not guarantee that all relations/actors in the graph are complete. The check is only performed when the node is added. Use the Graph.isComplete() method to check whether a graph is complete.

– **Parameters**

* `flag` - the new setting for this flag.

– **See Also**

* `notio.Relation.isComplete` (in 2.2.37, page 157)
* `notio.Actor.isComplete`
* `notio.Graph.isComplete` (in 2.2.19, page 107)

- *simplify*
`public void` **simplify**`( )`

– **Usage**

* Simplifies all relations in this graph.

- *simplify*
`public void` **simplify**`( notio.Relation` **relation** `)`

– **Usage**

* Simplifies the specified relation node by removing any duplicates of that relation in this graph.

– **Parameters**

* `relation` - the relation node to simplify.

– **Exceptions**

* `java.lang.IllegalArgumentException` - if the specified relation is not part of this graph.

- *simplify*
`public void` **simplify**`( notio.RelationType` **relType** `)`

– **Usage**

* Simplifies all instances of the specified relation type in this graph.

– **Parameters**

* `relType` - the relation type whose instances are to be simplified.

2.2.20 Class **InvalidDefiningConceptException**

Exception thrown when the setting of a defining concept gives rise to an error.

Declaration

public class InvalidDefiningConceptException
extends notio.OperationException

CONSTRUCTORS

- *InvalidDefiningConceptException*
  `public` **InvalidDefiningConceptException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.

- *InvalidDefiningConceptException*
  `public` **InvalidDefiningConceptException**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.
    - `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.21 CLASS JoinException

Exception thrown when an invalid join operation is attempted.

DECLARATION

public class JoinException
extends notio.OperationException

CONSTRUCTORS

- *JoinException*
  `public` **JoinException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.
- *JoinException*
  `public` **JoinException**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.
    - `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.22 CLASS KnowledgeBase

A class that defines a knowledge base.

Declaration

public class KnowledgeBase
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private ConceptTypeHierarchy conceptHierarchy
  - The concept type hierarchy for the knowledge base.
- private RelationTypeHierarchy relationHierarchy
  - The relation type hierarchy for the knowledge base.
- private MarkerSet markerSet
  - The marker set for the knowledge base.
- private Concept outermostContext
  - The outermost context of this knowledge base.

Constructors

- *KnowledgeBase*
  `public` **KnowledgeBase**`( )`
  - **Usage**
    - Constructs a new knowledge base and creates required components automatically. Currently, this automatically creates a ConceptType with the label "Proposition" and adds it to the concept type hierarchy with the universal and absurd types as super- and subtype respectively. It then creates a Concept with the proposition type, a null (existential) quantifier, and a DescriptorDesignator that references an empty Graph.

- *KnowledgeBase*
  `public` **KnowledgeBase**`( notio.ConceptTypeHierarchy` **newConceptHierarchy**`, notio.RelationTypeHierarchy` **newRelationHierarchy**`, notio.MarkerSet` **newMarkerSet**`, notio.Concept` **newOutermostContext** `)`
  - **Usage**
    - Constructs a new knowledge base with the specified components.
  - **Parameters**
    - `newConceptHierarchy` - the concept type hierarchy.
    - `newRelationHierarchy` - the relation type hierarchy.
    - `newMarkerSet` - the marker set.
    - `newOutermostContext` - the outermost context.

METHODS

- *getConceptTypeHierarchy*
  `public ConceptTypeHierarchy` **getConceptTypeHierarchy**`( )`
  - **Usage**
    - Returns the concept type hierarchy.
  - **Returns** - the concept type hierarchy.

- *getMarkerSet*
  `public MarkerSet` **getMarkerSet**`( )`
  - **Usage**
    - Returns the marker set.
  - **Returns** - the marker set.

- *getOutermostContext*
  `public Concept` **getOutermostContext**`( )`
  - **Usage**
    - Returns the outermost context.
  - **Returns** - the outermost context.

- *getRelationTypeHierarchy*
  `public RelationTypeHierarchy` **getRelationTypeHierarchy**`( )`
  - **Usage**
    - Returns the relation type hierarchy.
  - **Returns** - the relation type hierarchy.

2.2.23 CLASS LiteralDesignator

Class for literal designators. This is used for references to any Java Object or Object sub-class. This allows for Strings, Integers, Images, or any type of data.

DECLARATION

public class LiteralDesignator
extends notio.Designator
implements java.io.Serializable

SERIALIZABLE FIELDS

- private Object literal
  - Literal object associated with this designator.
- private Marker marker
  - Marker associated with the literal object.

CONSTRUCTORS

- *LiteralDesignator*
  `public` **LiteralDesignator**`( )`
  - **Usage**
    - Constructs a new LiteralDesignator unassociated with any literal object. A literal may be associated using setLiteral().
  - **See Also**
    - `notio.LiteralDesignator.setLiteral`

- *LiteralDesignator*
  `public` **LiteralDesignator**`( java.lang.Object` **newLiteral**`, notio.MarkerSet` **markerSet** `)`
  - **Usage**
    - Constructs a new LiteralDesignator with the specified Object.
  - **Parameters**
    - `newLiteral` - the literal object.
    - `markerSet` - the marker set associated with this designator.

METHODS

- *copy*
  `public Designator` **copy**`( notio.CopyingScheme` **copyScheme**`, java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    - Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme. If the a new designator instance is created, the literal is not copied.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
    - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getDesignatorKind*
  `public int` **getDesignatorKind**`( )`
  - **Usage**
    - Returns a constant indicating which kind of designator this is. In this case the constant will be: Designator.DESIGNATOR_LITERAL
  - **Returns** - a constant indicating the kind of the designator.

- *getLiteral*
  `public Object` **getLiteral**`( )`

- **Usage**
  - Returns the literal object referenced by this designator. This should be the same as the result returned by a getIndividual() call to the marker associated with this designator.
- **Returns** - the literal object referenced by this designator.

- *getMarker*
  `public Marker` **getMarker( )**
  - **Usage**
    - Returns the marker associated with this literal designator.
  - **Returns** - the marker associated with this literal designator.

- *setLiteral*
  `public void` **setLiteral(** `java.lang.Object` **newLiteral,** `notio.MarkerSet` **markerSet)**
  - **Usage**
    - Sets the literal object referenced by this designator. If a literal has already been specified, both the literal and the marker will be replaced by the new values. Calling this method will associate this designator with the marker for the literal object or create one if necessary.
  - **Parameters**
    - `newLiteral` - the literal object to be referenced by this designator.
    - `markerSet` - the marker set associated with this designator.
  - **Returns** - the literal object referenced by this designator.

METHODS INHERITED FROM CLASS `notio.Designator`

(in 2.2.16, page 92)

- *copy*
  `public abstract Designator` **copy(** `notio.CopyingScheme` **copyScheme,** `java.util.Hashtable` **substitutionTable)**
  - **Usage**
    - Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
    - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getCaseSensitiveLabels*
  `public boolean` **getCaseSensitiveLabels( )**
  - **Usage**
    - Returns true if the processing of labels in this designator is case-sensitive.
  - **Returns** - true if the processing of labels in this designator is case-sensitive.

- *getDesignatorKind*
  `public abstract int` **getDesignatorKind( )**

  - **Usage**
    * Returns a constant indicating which kind of designator this is. The constant is one of: Designator.DESIGNATOR_LITERAL Designator.DESIGNATOR_MARKER Designator.DESIGNATOR_DEFINED Designator.DESIGNATOR_NAME
  - **Returns** - a constant indicating the kind of the designator.

- *getEnclosingReferent*
  `public Referent` **getEnclosingReferent**`( )`
  - **Usage**
    * Returns the enclosing referent for this designator or null if there isn't one.
  - **Returns** - the enclosing referent for this designator or null.

- *matchDesignators*
  `public static MatchResult` **matchDesignators**`( notio.Designator` **first**`, notio.Designator` **second**`, notio.MatchingScheme` **matchingScheme** `)`
  - **Usage**
    * Compares two designators to decide if they match. The exact semantics of matching are determined by the match control flag.
  - **Parameters**
    * `first` - the first designators being matched.
    * `second` - the second designators being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two designators match according to the scheme's criteria.

- *setCaseSensitiveLabels*
  `public void` **setCaseSensitiveLabels**`( boolean` **flag** `)`
  - **Usage**
    * Sets a flag indicating whether or not the processing of labels within this designator is case-sensitive.
  - **Parameters**
    * `flag` - the flag setting for case-sensitivity.

2.2.24 Class Marker

The marker class.

Declaration

public class Marker
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private Object individual
  - Individual associated with this marker.
- private Set types
  - Set of types to which this marker conforms.

CONSTRUCTORS

- *Marker*
  `public` **Marker(** `notio.MarkerSet` **newMarkerSet)**
  - **Usage**
    - Construct a new marker belonging to the specified marker set.
  - **Parameters**
    - `newMarkerSet` - the marker set to which this marker will belong.

- *Marker*
  `public` **Marker(** `notio.MarkerSet` **newMarkerSet,** `java.lang.Object` **newIndividual)**
  - **Usage**
    - Construct a new marker belonging to the specified marker set and with the specified Object as its individual.
  - **Parameters**
    - `newMarkerSet` - the marker set to which this marker will belong.
    - `newIndividual` - the Object that is the individual corresponding to this marker.

METHODS

- *addTypeConformance*
  `public void` **addTypeConformance(** `notio.ConceptType` **newType)**
  - **Usage**
    - Adds a new concept type to the set of those to which this marker conforms.
  - **Parameters**
    - `newType` - the concept type being added.

- *conformsToType*
  `public boolean` **conformsToType(** `notio.ConceptType` **queryType)**
  - **Usage**
    - Test whether the marker conforms to the specified concept type.
  - **Parameters**
    - `queryType` - the concept type we are testing for conformance.
  - **Returns** - true if this marker conforms to the specified concept type.

- *freeIndividual*
  `public void` **freeIndividual( )**
  - **Usage**
    - Frees the individual associated with this marker. This means that the reference to the individual will be set to null, allowing it to be garbage-collected if no other references exist. The marker remains as an entry in its MarkerSet but is no longer associated with an individual. If no individual is associated with this Marker when freeIndividual() is called, nothing happens.

- *getIndividual*
  `public Object` **getIndividual( )**
  - **Usage**
    - Returns the individual corresponding to this marker.
  - **Returns** - the Object that is the individual corresponding to this marker.

- *getMarkerID*
  `public String` **getMarkerID( )**
  - **Usage**
    - Returns the marker ID for this marker. This is a string that unique represents the marker within its current marker set. Note, the ID is automatically assigned when the marker is added to its marker set.
  - **Returns** - the ID for this marker.

- *getMarkerSet*
  `public MarkerSet` **getMarkerSet( )**
  - **Usage**
    - Returns the marker set to which this marker belongs.
  - **Returns** - the MarkerSet for this marker.

- *removeTypeConformance*
  `public void` **removeTypeConformance(** `notio.ConceptType` **deadType)**
  - **Usage**
    - Removes a concept type from the set of those to which this marker conforms.
  - **Parameters**
    - `deadType` - the concept type being removed.

2.2.25 Class MarkerDesignator

Class for locator designators. Used for references by individual marker.

Declaration

public class MarkerDesignator
extends notio.Designator
implements java.io.Serializable

Serializable Fields

- private Marker marker
  - Marker associated with this designator.

CONSTRUCTORS

- *MarkerDesignator*
  `public` **MarkerDesignator**`( )`
  - **Usage**
    - Constructs a new MarkerDesignator with no associated marker. A marker may be assigned later using setMarker().
  - **See Also**
    - `notio.MarkerDesignator.setMarker`

- *MarkerDesignator*
  `public` **MarkerDesignator**`( notio.Marker` **newMarker** `)`
  - **Usage**
    - Constructs a new MarkerDesignator with the specified marker. The marker is assumed to be valid and already a member of some marker set.
  - **Parameters**
    - `newMarker` - the marker associated with this designator.

- *MarkerDesignator*
  `public` **MarkerDesignator**`( notio.MarkerSet` **markerSet** `)`
  - **Usage**
    - Constructs a new MarkerDesignator with the specified marker set. A marker will be constructed and added to the set automatically.
  - **Parameters**
    - `markerSet` - the marker set associated with this designator.

METHODS

- *copy*
  `public Designator` **copy**`( notio.CopyingScheme` **copyScheme**`,`
  `java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    - Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
    - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getDesignatorKind*
  `public int` **getDesignatorKind**`( )`
  - **Usage**

    * Returns a constant indicating which kind of designator is. In this case the constant will be: Designator.DESIGNATOR_MARKER
  - **Returns** - a constant indicating the kind of the designator.

- *getMarker*
  `public Marker` **getMarker( )**
  - **Usage**
    * Returns the marker associated with this designator.
  - **Returns** - the marker associated with this designator.

- *setMarker*
  `public void` **setMarker(** `notio.Marker` **newMarker)**
  - **Usage**
    * Sets the marker associated with this designator.
  - **Parameters**
    * `newMarker` - the marker to be associated with this designator.

METHODS INHERITED FROM CLASS `notio.Designator`

(in 2.2.16, page 92)

- *copy*
  `public abstract Designator` **copy(** `notio.CopyingScheme` **copyScheme,**
  `java.util.Hashtable` **substitutionTable)**
  - **Usage**
    * Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.
  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
    * `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getCaseSensitiveLabels*
  `public boolean` **getCaseSensitiveLabels(** **)**
  - **Usage**
    * Returns true if the processing of labels in this designator is case-sensitive.
  - **Returns** - true if the processing of labels in this designator is case-sensitive.

- *getDesignatorKind*
  `public abstract int` **getDesignatorKind(** **)**
  - **Usage**
    * Returns a constant indicating which kind of designator this is. The constant is one of: Designator.DESIGNATOR_LITERAL Designator.DESIGNATOR_MARKER Designator.DESIGNATOR_DEFINED Designator.DESIGNATOR_NAME
  - **Returns** - a constant indicating the kind of the designator.

- *getEnclosingReferent*
  `public Referent` **getEnclosingReferent(** **)**

  - **Usage**
    * Returns the enclosing referent for this designator or null if there isn't one.
  - **Returns** - the enclosing referent for this designator or null.

- *matchDesignators*
`public static MatchResult` **matchDesignators**`( notio.Designator` **first**`, notio.Designator` **second**`, notio.MatchingScheme` **matchingScheme** `)`

  - **Usage**
    * Compares two designators to decide if they match. The exact semantics of matching are determined by the match control flag.
  - **Parameters**
    * `first` - the first designators being matched.
    * `second` - the second designators being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two designators match according to the scheme's criteria.

- *setCaseSensitiveLabels*
`public void` **setCaseSensitiveLabels**`( boolean` **flag** `)`

  - **Usage**
    * Sets a flag indicating whether or not the processing of labels within this designator is case-sensitive.
  - **Parameters**
    * `flag` - the flag setting for case-sensitivity.

2.2.26 Class MarkerSet

Class that stores a set of marker objects. Markers are added to the set by specifying the set in the constructor for the marker.

Declaration

public class MarkerSet
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private int nextMarkerID
  - Next marker id.
- private Hashtable idTable
  - Hashtable for marker lookups by ID.
- private Hashtable individualTable
  - Hashtable for marker lookups by individual.

CONSTRUCTORS

- *MarkerSet*
  `public` **MarkerSet**`( )`

METHODS

- *getMarkerByIndividual*
  `public Marker` **getMarkerByIndividual**`( java.lang.Object` **individual** `)`
  - **Usage**
    - Returns the marker associated with the specified individual or null.
  - **Parameters**
    - `individual` - the individual being used to lookup.
  - **Returns** - the marker associated with this individual or null.
- *getMarkerByMarkerID*
  `public Marker` **getMarkerByMarkerID**`( java.lang.String` **markerID** `)`
  - **Usage**
    - Returns the marker associated with the specified name or null.
  - **Parameters**
    - `markerID` - the marker ID being used to lookup.
  - **Returns** - the marker associated with this name or null.

2.2.27 CLASS MatchingScheme

A class used to specify how matching of graph elements should be performed. When matching graphs or graph components, a MatchingScheme instance is used to describe exactly how matching should be performed. Not all matching schemes need be implemented by a given implementation and many schemes would not make sense. The exact behaviour of an implementation under these circumstances is undefined but it is recommended that the implementation throw notio.UnimplementedFeatureException either when an invalid scheme is constructed or when it is used in a matching method. MatchingScheme instances may be reused as they are not altered by the matching process.

DECLARATION

public class MatchingScheme
extends java.lang.Object

FIELDS

- public static final int GR_MATCH_INSTANCE
  - Graph matching control flag: arguments must be the same Graph instance.

- public static final int GR_MATCH_COMPLETE
  - Graph matching control flag: arguments completely match.
- public static final int GR_MATCH_SUBGRAPH
  - Graph matching control flag: first argument must be a subgraph of the second.
- public static final int GR_MATCH_PROPER_SUBGRAPH
  - Graph matching control flag: first argument must be a subgraph of the second.
- public static final int GR_MATCH_EITHER_SUBGRAPH
  - Graph matching control flag: one argument must be a subgraph of the other.
- public static final int GR_MATCH_EITHER_PROPER_SUBGRAPH
  - Graph matching control flag: one argument must be a proper subgraph of the other.
- public static final int GR_MATCH_COMMON_SUBGRAPH
  - Graph matching control flag: graphs must share one or more common subgraphs.
- public static final int GR_MATCH_ANYTHING
  - Graph matching control flag: arguments will always match.
- public static final int CN_MATCH_INSTANCE
  - Concept matching control flag: arguments must be the same Concept instance.
- public static final int CN_MATCH_TYPES
  - Concept matching control flag: arguments will be matched according to the concept type match flag.
- public static final int CN_MATCH_REFERENTS
  - Concept matching control flag: arguments will be matched according to the coreference, designator and quantifier match flags.
- public static final int CN_MATCH_COREFERENTS
  - Concept matching control flag: arguments will be matched using coreference flag only.
- public static final int CN_MATCH_ALL
  - Concept matching control flag: arguments will be matched using coreference, quantifier, designator, and type flags.
- public static final int CN_MATCH_ANYTHING
  - Concept matching control flag: arguments will always match.
- public static final int RN_MATCH_INSTANCE
  - Relation matching control flag: arguments must be the same Relation instance.
- public static final int RN_MATCH_TYPES
  - Relation matching control flag: arguments will be matched according to the relation type match flag.
- public static final int RN_MATCH_ARCS

  - Relation matching control flag: arguments will be matched according to the arc match flag.

- public static final int RN_MATCH_ALL
  - Relation matching control flag: arguments will be matched according to all other match flags.
- public static final int RN_MATCH_ANYTHING
  - Relation matching control flag: arguments will always match.
- public static final int CT_MATCH_INSTANCE
  - ConceptType matching control flag: arguments must be the same ConceptType instance.
- public static final int CT_MATCH_LABEL
  - ConceptType matching control flag: arguments must have the exact same type label.
- public static final int CT_MATCH_SUBTYPE
  - ConceptType matching control flag: first argument must be a subtype of the second.
- public static final int CT_MATCH_SUPERTYPE
  - ConceptType matching control flag: first argument must be a supertype of the second.
- public static final int CT_MATCH_EQUIVALENT
  - ConceptType matching control flag: arguments must have a sub/supertype relationship.
- public static final int CT_MATCH_ANYTHING
  - ConceptType matching control flag: arguments will always match.
- public static final int RT_MATCH_INSTANCE
  - RelationType matching control flag: arguments must be the same RelationType instance.
- public static final int RT_MATCH_LABEL
  - RelationType matching control flag: arguments must have the exact same type label.
- public static final int RT_MATCH_SUBTYPE
  - RelationType matching control flag: first argument must be a subtype of the second.
- public static final int RT_MATCH_SUPERTYPE
  - RelationType matching control flag: first argument must be a supertype of the second.
- public static final int RT_MATCH_EQUIVALENT
  - RelationType matching control flag: arguments must have a sub/supertype relationship.
- public static final int RT_MATCH_ANYTHING
  - RelationType matching control flag: arguments will always match.
- public static final int QF_MATCH_ANYTHING

  - Quantifier matching control flag: arguments will always match.

- public static final int DG_MATCH_INSTANCE
  - Designator matching control flag: arguments must be the same Designator instance.
- public static final int DG_MATCH_INDIVIDUAL
  - Designator matching control flag: arguments refer to the same individual.
- public static final int DG_MATCH_EQUIVALENT
  - Designator matching control flag: arguments may have a generic/generic or generic/specific relationship.
- public static final int DG_MATCH_RESTRICTION
  - Designator matching control flag: the first argument may be a restriction of the second.
- public static final int DG_MATCH_GENERALIZATION
  - Designator matching control flag: the first argument may be a generalization of the second.
- public static final int DG_MATCH_PROPER_RESTRICTION
  - Designator matching control flag: the first argument must be a restriction of the second.
- public static final int DG_MATCH_PROPER_GENERALIZATION
  - Designator matching control flag: the first argument must be a generalization of the second.
- public static final int DG_MATCH_ANYTHING
  - Designator matching control flag: arguments will always match.
- public static final int ARC_MATCH_INSTANCE
  - Arc matching control flag: arguments must be same Concept instances.
- public static final int ARC_MATCH_CONCEPT
  - Arc matching control flag: arguments will be matched according to scheme's concept flag.
- public static final int ARC_MATCH_VALENCE
  - Arc matching control flag: arguments must have the same number of arcs.
- public static final int ARC_MATCH_ANYTHING
  - Arc matching control flag: arguments will always match.
- public static final int COREF_AUTOMATCH_OFF
  - Coreference matching control flag: coreferent concepts will be treated as normal concepts.
- public static final int COREF_AUTOMATCH_ON
  - Coreference matching control flag: coreferent concepts will match regardless of types and referents.

- public static final int COREF_AGREE_OFF
  - Coreference matching control flag: when matching a concept, all coreferent concepts must match as well.
- public static final int COREF_AGREE_ON
  - Coreference matching control flag: when matching a concept, coreferent concepts will not be considered.
- public static final int FOLD_MATCH_OFF
  - Folding matching control flag: folds will not be matched.
- public static final int FOLD_MATCH_ON
  - Folding matching control flag: folds will be matched.
- public static final int CONN_MATCH_OFF
  - Connected graph matching control flag: disconnected matches are allowed.
- public static final int CONN_MATCH_ON
  - Connected graph matching control flag: disconnected matches are not allowed.
- public static final int MARKER_MATCH_ID
  - Marker matching control flag: markers match only if their ID's are the same.
- public static final int MARKER_MATCH_COMPARATOR
  - Marker matching control flag: markers are matched using the marker comparator specified in this scheme.
- public static final int MARKER_MATCH_ANYTHING
  - Marker matching control flag: markers always match.

CONSTRUCTORS

- *MatchingScheme*
  `public` **MatchingScheme(** `int` **newGraphFlag,** `int` **newConceptFlag,** `int` **newRelationFlag,** `int` **newConceptTypeFlag,** `int` **newRelationTypeFlag,** `int` **newQuantifierFlag,** `int` **newDesignatorFlag,** `int` **newMarkerFlag,** `int` **newArcFlag,** `int` **newCorefAutoMatchFlag,** `int` **newCorefAgreementFlag,** `int` **newFoldingFlag,** `int` **newConnectedFlag,** `int` **newMaxMatches,** `notio.MarkerComparator` **newMarkerComparator,** `notio.MatchingScheme` **newNestedScheme)**
  - **Usage**
    - Constructs a matching scheme with the specified control flags.
  - **Parameters**
    - `newGraphFlag` - Matching flag for graphs.
    - `newConceptFlag` - Matching flag for concepts.
    - `newRelationFlag` - Matching flag for relations.
    - `newConceptTypeFlag` - Matching flag for concept types.
    - `newRelationTypeFlag` - Matching flag for relation types.
    - `newQuantifierFlag` - Matching flag for quantifiers.

  * `newDesignatorFlag` - Matching flag for designators.
  * `newMarkerFlag` - Matching flag for markers.
  * `newArcFlag` - Matching flag for arcs.
  * `newCorefAutoMatchFlag` - Flag for automatching coreference concepts.
  * `newCorefAgreementFlag` - Flag for forcing agreement with corefernt concepts.
  * `newFoldingFlag` - Matching flag for folding.
  * `newConnectedFlag` - Matching flag for connected graphs.
  * `newMaxMatches` - The maximum number of graph matches to generate in this context. A zero indicates that all possible matches should be generated.
  * `newMarkerComparator` - a MarkerComparator to be used for matching Markers (must be null if the marker matching flag does not call for a comparator).
  * `newNestedScheme` - A nested matching scheme to be used for matching nested graphs (null means use present scheme).

METHODS

- *getArcFlag*
`public int` **`getArcFlag( )`**
  - **Usage**
    * Returns the arc matching control flag for this scheme.
  - **Returns** - the arc matching control flag for this scheme.

- *getConceptFlag*
`public int` **`getConceptFlag( )`**
  - **Usage**
    * Returns the concept matching control flag for this scheme.
  - **Returns** - the concept matching control flag for this scheme.

- *getConceptTypeFlag*
`public int` **`getConceptTypeFlag( )`**
  - **Usage**
    * Returns the concept type matching control flag for this scheme.
  - **Returns** - the concept type matching control flag for this scheme.

- *getConnectedFlag*
`public int` **`getConnectedFlag( )`**
  - **Usage**
    * Returns the connected graph matching control flag for this scheme.
  - **Returns** - the connected graph matching control flag for this scheme.

- *getCoreferenceAgreementFlag*
`public int` **`getCoreferenceAgreementFlag( )`**
  - **Usage**
    * Returns the coreference agreement control flag for this scheme.
  - **Returns** - the coreference agreement control flag for this scheme.

- *getCoreferenceAutoMatchFlag*
  `public int` **getCoreferenceAutoMatchFlag( )**
  - **Usage**
    - Returns the coreference auto-matching control flag for this scheme.
  - **Returns** - the coreference auto-matching control flag for this scheme.

- *getDesignatorFlag*
  `public int` **getDesignatorFlag( )**
  - **Usage**
    - Returns the designator matching control flag for this scheme.
  - **Returns** - the designator matching control flag for this scheme.

- *getFoldingFlag*
  `public int` **getFoldingFlag( )**
  - **Usage**
    - Returns the folding matching control flag for this scheme.
  - **Returns** - the folding matching control flag for this scheme.

- *getGraphFlag*
  `public int` **getGraphFlag( )**
  - **Usage**
    - Returns the graph matching control flag for this scheme.
  - **Returns** - the graph matching control flag for this scheme.

- *getMarkerComparator*
  `public MarkerComparator` **getMarkerComparator( )**
  - **Usage**
    - Returns the MarkerComparator to be used in this scheme, if any.
  - **Returns** - the MarkerComparator to be used in this scheme, if any.

- *getMarkerFlag*
  `public int` **getMarkerFlag( )**
  - **Usage**
    - Returns the marker matching control flag for this scheme.
  - **Returns** - the marker matching control flag for this scheme.

- *getMaxMatches*
  `public int` **getMaxMatches( )**
  - **Usage**
    - Returns the maximum number of graph matches to be generated this scheme.
  - **Returns** - the maximum number of graph matches to be generated this scheme.

- *getNestedMatchingScheme*
  `public MatchingScheme` **getNestedMatchingScheme( )**

  - **Usage**
    * Returns the nested matching scheme or null. The nested matching scheme is used for matching nested graphs. If it is set to null, the current scheme is used.
  - **Returns** - the nested matching scheme or null.

- *getQuantifierFlag*
  `public int` **getQuantifierFlag( )**

  - **Usage**
    * Returns the quantifier matching control flag for this scheme.
  - **Returns** - the quantifier matching control flag for this scheme.

- *getRelationFlag*
  `public int` **getRelationFlag( )**

  - **Usage**
    * Returns the relation matching control flag for this scheme.
  - **Returns** - the relation matching control flag for this scheme.

- *getRelationTypeFlag*
  `public int` **getRelationTypeFlag( )**

  - **Usage**
    * Returns the relation type matching control flag for this scheme.
  - **Returns** - the relation type matching control flag for this scheme.

2.2.28 Class MatchResult

A class for reporting the results of a match operation. For most types of match, this is simply a boolean result, but for some match types, it also contains node mappings.

Declaration

public class MatchResult
extends java.lang.Object

Constructors

- *MatchResult*
  `public` **MatchResult(** `boolean` **resultFlag)**

  - **Usage**
    * Constructs a new MatchResult with the specified result flag.
  - **Parameters**
    * `resultFlag` - a boolean value indicating whether the match was successful or not.

- *MatchResult*
  `public` **MatchResult(** `notio.NodeMapping []` **newMappings)**

  - **Usage**
    * Constructs a new MatchResult with the specified mappings. The array of mappings must not be null although it could be empty.
  - **Parameters**
    * `newMappings` - an array of node mappings that describe the matches found.

METHODS

- *getMappings*
  `public NodeMapping` **getMappings**`( )`
  - **Usage**
    * Returns the node mappings generated by the match (if any). If the match failed, this method will return null.
  - **Returns** - an array containing node mappings generated by the match or null if the match failed.

- *getNumberOfMatches*
  `public int` **getNumberOfMatches**`( )`
  - **Usage**
    * Returns the number of matches (mappings) found. If the search succeeded but no mappings were generated, a 1 is returned. If the search fails, a 0 is returned.
  - **Returns** - the number of matches found.

- *matchSucceeded*
  `public boolean` **matchSucceeded**`( )`
  - **Usage**
    * Returns true if one or matches were found.
  - **Returns** - true if one or matches were found.

2.2.29 CLASS NameAddException

Exception thrown when the addition of a name to a marker gives rise to an error.

DECLARATION

public class NameAddException **extends** notio.OperationException

CONSTRUCTORS

- *NameAddException*
  `public` **NameAddException**`( java.lang.String` **message** `)`
  - **Usage**

    * Constructs an exception with the specified message.
  - **Parameters**
    * `message` - The details of the exception.

- *NameAddException*
  `public` **NameAddException**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    * Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    * `message` - the details of the exception.
    * `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    * This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    * This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    * `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`

- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`

- *getMessage*
  `public String` **getMessage**`( )`

- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.30 Class NameDesignator

Class for name designators. Used for references by name.

Declaration

public class NameDesignator
extends notio.Designator
implements java.io.Serializable

Serializable Fields

- private String name
  - Name which is used as the locator in this designator.

Constructors

- *NameDesignator*
  `public` **NameDesignator**`( )`
  - **Usage**
    - Constructs a new NameDesignator unassociated with any name.
- *NameDesignator*
  `public` **NameDesignator**`( java.lang.String` **newName** `)`
  - **Usage**
    - Constructs a new NameDesignator with the specified name.
  - **Parameters**
    - `newName` - the name.

METHODS

- *copy*
  `public Designator` **copy**`( notio.CopyingScheme` **copyScheme**`, java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    - Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
    - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getDesignatorKind*
  `public int` **getDesignatorKind**`( )`
  - **Usage**
    - Returns a constant indicating which kind of designator is. In this case the constant will be: Designator.DESIGNATOR_NAME
  - **Returns** - a constant indicating the kind of the designator.

- *getName*
  `public String` **getName**`( )`
  - **Usage**
    - Returns the name used in this designator.
  - **Returns** - the name.

- *setName*
  `public void` **setName**`( java.lang.String` **newName** `)`
  - **Usage**
    - Sets the name used in this designator.
  - **Parameters**
    - `newName` - the name to be associated with this designator.

METHODS INHERITED FROM CLASS `notio.Designator`

(in 2.2.16, page 92)

- *copy*
  `public abstract Designator` **copy**`( notio.CopyingScheme` **copyScheme**`, java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    - Performs a copy operation on this designator according to the the specified CopyingScheme. The result may be a new designator or simply a reference to this designator depending on the scheme.

  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
    * `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
  - **Returns** - the result of the copy operation.

- *getCaseSensitiveLabels*
`public boolean` **getCaseSensitiveLabels**`( )`

  - **Usage**
    * Returns true if the processing of labels in this designator is case-sensitive.
  - **Returns** - true if the processing of labels in this designator is case-sensitive.

- *getDesignatorKind*
`public abstract int` **getDesignatorKind**`( )`

  - **Usage**
    * Returns a constant indicating which kind of designator this is. The constant is one of: Designator.DESIGNATOR_LITERAL Designator.DESIGNATOR_MARKER Designator.DESIGNATOR_DEFINED Designator.DESIGNATOR_NAME
  - **Returns** - a constant indicating the kind of the designator.

- *getEnclosingReferent*
`public Referent` **getEnclosingReferent**`( )`

  - **Usage**
    * Returns the enclosing referent for this designator or null if there isn't one.
  - **Returns** - the enclosing referent for this designator or null.

- *matchDesignators*
`public static MatchResult` **matchDesignators**`( notio.Designator` **first**`, notio.Designator` **second**`, notio.MatchingScheme` **matchingScheme** `)`

  - **Usage**
    * Compares two designators to decide if they match. The exact semantics of matching are determined by the match control flag.
  - **Parameters**
    * `first` - the first designators being matched.
    * `second` - the second designators being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two designators match according to the scheme's criteria.

- *setCaseSensitiveLabels*
`public void` **setCaseSensitiveLabels**`( boolean` **flag** `)`

  - **Usage**
    * Sets a flag indicating whether or not the processing of labels within this designator is case-sensitive.
  - **Parameters**
    * `flag` - the flag setting for case-sensitivity.

2.2.31 Class Node

The base class for graph nodes. Provides the admittedly tiny amount of common functionality and serves chiefly to make collections of nodes simpler to manage.

Declaration

public abstract class Node
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private Type type
  - The type for this node.
- private Graph enclosingGraph
  - The enclosing graph for this node.
- private String comment
  - The comment associated with this node.

Constructors

- *Node*
  `public` **Node( )**

Methods

- *getComment*
  `public String` **getComment( )**
  - **Usage**
    - Returns the comment string for this node.
  - **Returns** - the comment string associated with this node or null.

- *getEnclosingGraph*
  `public Graph` **getEnclosingGraph( )**
  - **Usage**
    - Returns the graph that encloses this node or null if the node does not currently belong to a graph.
  - **Returns** - the node's enclosing graph.

- *setComment*
  `public void` **setComment(** `java.lang.String` **newComment)**
  - **Usage**
    - Sets the comment string for this node.
  - **Parameters**
    - `newComment` - the new comment string for this node.

2.2.32 Class NodeMapping

A mapping between the nodes in two graphs. The primary purpose for this class is to provide the details of a match between two graphs. It consists of a list of pairs of the two types of nodes. Each pair is a mapping. Since there are some conditions that allow one-to-many mappings, there may be duplicates amongst the first or second elements of the pairs, but no two pairings will be identical.

For example: { a->B, a->C, b->B } is possible but { a->B, a->B } is not.

Declaration

public class NodeMapping
extends java.lang.Object

Constructors

- *NodeMapping*
  `public` **NodeMapping(** `notio.Graph` **newFirstGraph,** `notio.Graph` **newSecondGraph,** `notio.Concept []` **newFirstConcepts,** `notio.Concept []` **newSecondConcepts,** `notio.Relation []` **newFirstRelations,** `notio.Relation []` **newSecondRelations,** `notio.MatchResult []` **newMatchResults**)
  - **Usage**
    - Constructs a node mapping between the two graphs.
  - **Parameters**
    - `newFirstGraph` - the graph being mapped from.
    - `newSecondGraph` - the graph being mapped into.
    - `newFirstConcepts` - the array of concepts that form the first elements of the concept pairs.
    - `newSecondConcepts` - the array of concepts that form the second elements of the concept pairs.
    - `newFirstRelations` - the array of relations that form the first elements of the relation pairs.
    - `newSecondRelations` - the array of relations that form the second elements of the relation pairs.
    - `newMatchResults` - an array of match results from the matching of concepts with nested graphs corresponding to the order of the concept pairs.

Methods

- *getFirstConcepts*
  `public Concept` **getFirstConcepts(**)
  - **Usage**
    - Returns an array that forms the first elements of the Concept pairs.
  - **Returns** - an array of Concepts that map into the second array of Concepts.

- *getFirstGraph*
  `public Graph` **getFirstGraph( )**
  - **Usage**
    - Returns the first graph involved in the mapping.
  - **Returns** - the first graph involved in the mapping.

- *getFirstRelations*
  `public Relation` **getFirstRelations( )**
  - **Usage**
    - Returns an array that forms the first elements of the Relation pairs.
  - **Returns** - an array of Relations that map into the second array of Relations.

- *getMatchResults*
  `public MatchResult` **getMatchResults( )**
  - **Usage**
    - Returns an array of match results that come from matching the graphs nested in concepts. The order of the results corresponds to the order of the concept pairs returned by getFirstConcepts() and getSecondConcepts(). If the a concept did not have a nested graph, or if the matching scheme did not require that the graphs be matched, the corresponding match result will be null. If no match results wree specified when this mapping was constructed, this method will return null.
  - **Returns** - an array of match results that correspond to the pairs of concepts.

- *getSecondConcepts*
  `public Concept` **getSecondConcepts( )**
  - **Usage**
    - Returns an array that forms the second elements of the Concept pairs.
  - **Returns** - an array of Concepts that map into the first array of Concepts.

- *getSecondGraph*
  `public Graph` **getSecondGraph( )**
  - **Usage**
    - Returns the second graph involved in the mapping.
  - **Returns** - the second graph involved in the mapping.

- *getSecondRelations*
  `public Relation` **getSecondRelations( )**
  - **Usage**
    - Returns an array that forms the second elements of the Relation pairs.
  - **Returns** - an array of Relations that map into the first array of Relations.

2.2.33 Class **OperationError**

The base Notio operation error class. This error class allows other throwables to be embedded in it. This provides a clean means for implementation-specific throwables to be thrown from Notio routines. Note that since OperationError is a subclass of Error, applications are not required to explicitly throw or catch it. As such it is reserved for Notio errors that are most probably due to a programming error in the application rather than a reasonable misuse of the API.

Declaration

public class OperationError
extends java.lang.Error

Serializable Fields

- private Throwable subThrowable
  - An throwable enclosed within this one.

Constructors

- *OperationError*
  `public` **OperationError**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an error with the specified message.
  - **Parameters**
    - `message` - the details of the error.
- *OperationError*
  `public` **OperationError**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an error with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the error.
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

Methods

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**

    * This method retrieves arbitrary throwables embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable(** `java.lang.Throwable` **newSubThrowable)**
  - **Usage**
    * This method allows arbitrary throwables to be embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    * `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

METHODS INHERITED FROM CLASS `java.lang.Error`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace(** **)**
- *getLocalizedMessage*
  `public String` **getLocalizedMessage(** **)**
- *getMessage*
  `public String` **getMessage(** **)**
- *printStackTrace*
  `public void` **printStackTrace(** **)**
- *printStackTrace*
  `public void` **printStackTrace(** `java.io.PrintStream` **)**
- *printStackTrace*
  `public void` **printStackTrace(** `java.io.PrintWriter` **)**
- *toString*
  `public String` **toString(** **)**

2.2.34 CLASS OperationException

The base Notio operation exception class. This exception class allows other throwables to be embedded in it. This provides a clean means for implementation-specific throwables to be thrown from Notio routines.

DECLARATION

---

public class OperationException **extends** java.lang.Exception

Serializable Fields

- private Throwable subThrowable
  - An throwable enclosed within this one.

Constructors

- *OperationException*
  `public` **OperationException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - the details of the exception.

- *OperationException*
  `public` **OperationException**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

Methods

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.35 CLASS ParserException

Exception thrown when some parser's operation gives rise to an error.

DECLARATION

public class ParserException
extends notio.TranslationException

SERIALIZABLE FIELDS

- private int beginColumn
  - The column in which the current token begins.
- private int endColumn
  - The column in which the current token ends.
- private int beginLine
  - The line in which the current token begins.
- private int endLine
  - The line in which the current token ends.
- private String offendingToken
  - A string containing the offending token.

- private String expectedTokens
  - An array of strings containing the possible expected tokens.
- private boolean beforeFlag
  - A flag indicating the error occurred sometime before the specified token or location.
- private boolean afterFlag
  - A flag indicating the error occurred sometime after the specified token or location.

CONSTRUCTORS

- *ParserException*
  `public` **ParserException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.

- *ParserException*
  `public` **ParserException**`( java.lang.String` **message**`, java.lang.String` **newOffendingToken**`, int` **newBeginLine**`, int` **newEndLine**`, int` **newBeginColumn**`, int` **newEndColumn**`, java.lang.String []` **newExpectedTokens**`, boolean` **newBeforeFlag**`, boolean` **newAfterFlag** `)`
  - **Usage**
    - Constructs an exception with the specified message and details of the token that triggered this exception.
  - **Parameters**
    - `message` - the details of the exception.
    - `newOffendingToken` - the offending token.
    - `newBeginLine` - the line on which the offending token begins.
    - `newEndLine` - the line on which the offending token ends.
    - `newBeginColumn` - the column at which the offending token begins.
    - `newEndColumn` - the column at which the offending token ends.
    - `newExpectedTokens` - an array containing tokens that were expected in place of the offending token.
    - `newBeforeFlag` - a flag indicating that the cause of the exception is actually somewhere before the specified token or location.
    - `newAfterFlag` - a flag indicating that the cause of the exception is actually somewhere after the specified token or location.

- *ParserException*
  `public` **ParserException**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.

  * `newSubThrowable` - the sub-throwable to be embedded in this exception.

- *ParserException*
  `public` **ParserException(** `java.lang.String` **message,** `java.lang.Throwable` **newSubThrowable,** `java.lang.String` **newOffendingToken,** `int` **newBeginLine,** `int` **newEndLine,** `int` **newBeginColumn,** `int` **newEndColumn,** `java.lang.String []` **newExpectedTokens,** `boolean` **newBeforeFlag,** `boolean` **newAfterFlag)**
  - **Usage**
    * Constructs an exception with the specified message, sub-throwable, and details of the token that triggered this exception.
  - **Parameters**
    * `message` - the details of the exception.
    * `newSubThrowable` - the sub-throwable to be embedded in this exception.
    * `newOffendingToken` - the offending token.
    * `newBeginLine` - the line on which the offending token begins.
    * `newEndLine` - the line on which the offending token ends.
    * `newBeginColumn` - the column at which the offending token begins.
    * `newEndColumn` - the column at which the offending token ends.
    * `newExpectedTokens` - an array containing tokens that were expected in place of the offending token.
    * `newBeforeFlag` - a flag indicating that the cause of the exception is actually somewhere before the specified token or location.
    * `newAfterFlag` - a flag indicating that the cause of the exception is actually somewhere after the specified token or location.

Methods

- *getBeginColumn*
  `public int` **getBeginColumn( )**
  - **Usage**
    * Returns the beginning column of the offending token related to this exception or 0 if none has been specified. Note that it is possible to have a beginning column but still have null returned by getOffendingToken(). Columns are numbered starting at 1.
  - **Returns** - the beginning column of this exception or 0.

- *getBeginLine*
  `public int` **getBeginLine( )**
  - **Usage**
    * Returns the beginning line of the offending token related to this exception or 0 if none has been specified. Note that it is possible to have a beginning line but still have null returned by getOffendingToken(). Lines are numbered starting at 1.
  - **Returns** - the beginning line of this exception or 0.

- *getEndColumn*
  `public int` **getEndColumn( )**

  - **Usage**
    * Returns the ending column of the offending token related to this exception or 0 if none has been specified. Note that it is possible to have a ending column but still have null returned by getOffendingToken(). Columns are numbered starting at 1.
  - **Returns** - the ending column of this exception or 0.

- *getEndLine*
`public int` **`getEndLine( )`**

  - **Usage**
    * Returns the ending line of the offending token related to this exception or 0 if none has been specified. Note that it is possible to have a ending line but still have null returned by getOffendingToken(). Lines are numbered starting at 1.
  - **Returns** - the ending line of this exception or 0.

- *getExpectedTokens*
`public String` **`getExpectedTokens( )`**

  - **Usage**
    * Returns an array containing tokens that would have been accept in place of the offending token related to this exceptionm, or null if none have been specified.
  - **Returns** - the array of expected tokens or null.

- *getOccurranceMessage*
`public String` **`getOccurranceMessage( )`**

  - **Usage**
    * Returns a string containing information about the token and/or location involved in this exception, or null if no details are available. The standard getMessage() method in this exception should only return an explanation of the error. It will frequently be desirable to provide both messages, but on occasion only one will be desired or the formatting must be tightly controlled. The message will always be a single line. If more control is needed, the methods for accessing individual details about the exception should be used to construct a message. Parsing the results of this method is strongly discouraged since no guarantees are provided as to its format.
  - **Returns** - a string containing the message, or null.

- *getOffendingToken*
`public String` **`getOffendingToken( )`**

  - **Usage**
    * Returns the offending token related to this exception or null if none has been specified.
  - **Returns** - the offending token or null.

METHODS INHERITED FROM CLASS `notio.TranslationException`

(in 2.2.45, page 179)

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.36 CLASS Referent

A class for storing the referent of a concept. A referent consists of a quantifier, a designator, and a descriptor (nested graph). A null quantifier indicates that the enclosing concept is existentially quantified. A null designator indicates that the enclosing concept is unspecified. A null descriptor indicates that the enclosing concept is undescribed. A combination of a null quantifier, null designator, and null descriptor indicates that the enclosing concept is a generic concept. A concept that has no referent at all (null) is also considered to be a generic concept.

Declaration

public class Referent
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private Designator designator
  - The designator for this referent.
- private Macro quantifier
  - The quantifier for this referent.
- private Graph descriptor
  - The descriptor for this referent.
- private CoreferenceSet corefSets
  - The coreference sets to which this referent belongs.
- private Concept enclosingConcept
  - The concept that encloses this referent.

Constructors

- *Referent*
  `public` **Referent**`( )`
  - **Usage**
    - Constructs a new referent with no quantifier, designator or descriptor. This is automatically a 'generic' referent. The quantifier is initialized to null, indicating existential quantification. The designator is initialized to null, indicating an unspecified referent. The decriptor is initialized to null, indicating an undescribed referent.

- *Referent*
  `public` **Referent**`( notio.Designator` **newDesignator** `)`
  - **Usage**
    - Constructs a new referent with the specified designator. The designator may be null, indicating an unspecified referent. The quantifier is initialized to null, indicating existential quantification. The decriptor is initialized to null, indicating an undescribed referent.
  - **Parameters**
    - `newDesignator` - a single designator for this referent.

- *Referent*
  `public` **Referent(** `notio.Designator` **newDesignator,** `notio.Graph` **newDescriptor)**
  - **Usage**
    - Constructs a new referent with the specified designator and descriptor. The designator may be null, indicating an unspecified referent. The descriptor may be null, indicating an undescribed referent. The quantifier is initialized to null, indicating existential quantification.
  - **Parameters**
    - `newDesignator` - the designator for this referent.
    - `newDescriptor` - the descriptor for this referent.

- *Referent*
  `public` **Referent(** `notio.Graph` **newDescriptor)**
  - **Usage**
    - Constructs a new referent with the specified descriptor. The descriptor may be null, indicating an undescribed referent. The quantifier is initialized to null, indicating existential quantification. The designator is initialized to null, indicating an unspecified referent.
  - **Parameters**
    - `newDescriptor` - the descriptor for this referent.

- *Referent*
  `public` **Referent(** `notio.Macro` **newQuantifier)**
  - **Usage**
    - Constructs a new referent with the specified quantifier. The quantifier may be null, indicating existential quantification. The designator is initialized to null, indicating an unspecified referent. The decriptor is initialized to null, indicating an undescribed referent.
  - **Parameters**
    - `newQuantifier` - the quantifier for this referent.

- *Referent*
  `public` **Referent(** `notio.Macro` **newQuantifier,** `notio.Designator` **newDesignator)**
  - **Usage**
    - Constructs a new referent with the specified quantifier and designator. The quantifier may be null, indicating existential quantification. The designator may be null, indicating an unspecified referent. The decriptor is initialized to null, indicating an undescribed referent.
  - **Parameters**
    - `newQuantifier` - the quantifier for this referent.
    - `newDesignator` - the designator for this referent.

- *Referent*
  `public` **Referent(** `notio.Macro` **newQuantifier,** `notio.Designator` **newDesignator,** `notio.Graph` **newDescriptor)**

  - **Usage**
    * Constructs a new referent with the specified quantifier, designator, and descriptor. The quantifier may be null, indicating existential quantification. The designator may be null, indicating an unspecified referent. The descriptor may be null, indicating an undescribed referent.
  - **Parameters**
    * `newQuantifier` - the quantifier for this referent.
    * `newDesignator` - the designator for this referent.
    * `newDescriptor` - the descriptor for this referent.

- *Referent*
  `public` **Referent**`( notio.Macro` **newQuantifier**`, notio.Graph` **newDescriptor** `)`
  - **Usage**
    * Constructs a new referent with the specified quantifier and descriptor. The quantifier may be null, indicating existential quantification. The descriptor may be null, indicating an undescribed referent. The designator is initialized to null, indicating an unspecified referent.
  - **Parameters**
    * `newQuantifier` - the quantifier for this referent.
    * `newDescriptor` - the descriptor for this referent.

METHODS

- *copy*
  `public Referent` **copy**`( notio.CopyingScheme` **copyScheme** `)`
  - **Usage**
    * Performs a copy operation on this referent according to the the specified CopyingScheme. The result may be a new referent or simply a reference to this referent depending on the scheme. Coreference sets will be copied as required by the CopyingScheme.
  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
  - **Returns** - the result of the copy operation.

- *copy*
  `public Referent` **copy**`( notio.CopyingScheme` **copyScheme**`, java.util.Hashtable` **substitutionTable** `)`
  - **Usage**
    * Performs a copy operation on this referent according to the the specified CopyingScheme. The result may be a new referent or simply a reference to this referent depending on the scheme. Coreference sets will be copied as required by the CopyingScheme.
  - **Parameters**
    * `copyScheme` - the copying scheme used to control the copy operation.
    * `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.

  - **Returns** - the result of the copy operation.

- *getDescriptor*
`public Graph` **getDescriptor( )**
  - **Usage**
    * Returns this referent's descriptor. Null indicates an undescribed referent.
  - **Returns** - this referent's descriptor or null.

- *getDesignator*
`public Designator` **getDesignator( )**
  - **Usage**
    * Returns this referent's designator. Null indicates an unspecified referent.
  - **Returns** - this referent's designator or null.

- *getEnclosingConcept*
`public Concept` **getEnclosingConcept( )**
  - **Usage**
    * Returns the concept that encloses this referent.
  - **Returns** - the concept that encloses this referent.

- *getQuantifier*
`public Macro` **getQuantifier( )**
  - **Usage**
    * Returns this referent's quantifier. Null indicates existential quantification.
  - **Returns** - this concept's quantifier.

- *isContext*
`public boolean` **isContext( )**
  - **Usage**
    * Returns true if this referent forms a context. A referent forms a context if it has a descriptor.
  - **Returns** - true if this referent forms a context.

- *matchReferents*
`public static MatchResult` **matchReferents(** `notio.Referent` **first,** `notio.Referent` **second,** `notio.MatchingScheme` **matchingScheme)**
  - **Usage**
    * Compares two referents to decide if they match. The exact semantics of matching are determined by the matching scheme.
  - **Parameters**
    * `first` - the first referent being matched.
    * `second` - the second referent being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two referents match according to the scheme's criteria.

- *setDescriptor*
  `public void` **setDescriptor(** `notio.Graph` **newDescriptor)**
  - **Usage**
    - Sets this referent's descriptor. Null indicates undescribed.
  - **Parameters**
    - `newDescriptor` - this referent's new descriptor.

- *setDesignator*
  `public void` **setDesignator(** `notio.Designator` **newDesignator)**
  - **Usage**
    - Sets this concept's designator. Null indicates an unspecified referent.
  - **Parameters**
    - `newDesignator` - this referent's new designator or null.

- *setQuantifier*
  `public void` **setQuantifier(** `notio.Macro` **newQuantifier)**
  - **Usage**
    - Sets this referent's quantifier. Null indicates existential quantification.
  - **Parameters**
    - `newQuantifier` - this referent's new quantifier.

2.2.37 Class Relation

The conceptual relation node class. This class encapsulates all available information about a conceptual relation within a graph. It consists of a type and/or arguments (concepts connected to the arcs of the relation). The arguments can be divided into input and output arguments as desired, though the default is that the last (and possibly only) argument is the sole output arc. Relations with no arguments (zero valence) are allowed. Note that the terms 'arc' and 'argument' are used interchangably throughout the documentation.

Conceptually, the *N* arguments of a relation are all stored in one array, with zero being the index of the first argument. The first *k* elements of this array are considered to be inputs and the remaining *N* - *k* are outputs. This allows them to be both ordered and directed at the same time. The boundary between inputs and outputs is defined using the "output start index". This is the index into the array of arguments at which the first output argument occurs. If there are no outputs, the output start index is 1 greater than the number of inputs. Using setArguments() sets the entire array. Using setInputArgument() and setOutputArgument() sets the same array plus the index that defines where the outputs begin. That index can be set and found explicitly by calling setOutputStartIndex() and getOutputStartIndex(). In the end, it's all just one array and an index that shows where the outputs start. The default output start index for relations is *N* - *1*, as specified by the standard.

Declaration

public class Relation
extends notio.Node
implements java.io.Serializable

CONSTRUCTORS

- *Relation*
  `public` **Relation( )**
  - **Usage**
    - Constructs a new relation that has no type and no arguments. The number of arguments is automatically set to zero.

- *Relation*
  `public` **Relation(** `notio.Concept []` **newArguments)**
  - **Usage**
    - Constructs a relation with no type and the specified arguments. For a non-zero valence, the last argument is assumed to be the sole output argument.
  - **Parameters**
    - `newArguments` - the concepts related by this relation.

- *Relation*
  `public` **Relation(** `notio.Concept []` **newArguments,** `int` **newOutputStartIndex)**
  - **Usage**
    - Constructs a relation with no type and the specified arguments with output arguments starting at the specified index.
  - **Parameters**
    - `newArguments` - the concepts related by this relation.
    - `newOutputStartIndex` - the index into the newArguments array at which the output arcs start.

- *Relation*
  `public` **Relation(** `notio.RelationType` **newType)**
  - **Usage**
    - Constructs a relation with the given type. The valence is taken from the specified type and the arguments are initialized to null. If the type's valence is unspecified, a zero-length argument array is used for construction. For a specified non-zero valence, the last argument is assumed to be the sole output argument.
  - **Parameters**
    - `newType` - the type for this relation.

- *Relation*
  `public` **Relation(** `notio.RelationType` **newType,** `notio.Concept []` **newArguments)**
  - **Usage**
    - Constructs a relation with the given type and arguments. The last argument is assumed to be the sole output argument. The number of arguments must conform to the valence of the relation type unless it is unspecified in which case the valence is set to the number of arguments in the array.

- **Parameters**
  - `newType` - the type for this relation.
  - `newArguments` - the concepts related by this relation.

- *Relation*
  `public` **Relation(** `notio.RelationType` **newType,** `notio.Concept []` **newInputArguments,** `notio.Concept []` **newOutputArguments)**
  - **Usage**
    - Constructs a relation with the given type and the specified input and output arguments. The number of arguments must conform to the valence of the relation type unless it is unspecified in which case the valence is set to the number of arguments in the array.
  - **Parameters**
    - `newType` - the type for this relation.
    - `newInputArguments` - the input concepts related by this relation.
    - `newOutputArguments` - the output concepts related by this relation.

- *Relation*
  `public` **Relation(** `notio.RelationType` **newType,** `notio.Concept []` **newArguments,** `int` **newOutputStartIndex)**
  - **Usage**
    - Constructs a relation with the given type and arguments and output argument start index. The number of arguments must conform to the valence of the relation type unless it is unspecified in which case the valence is set to the number of arguments in the array.
  - **Parameters**
    - `newType` - the type for this relation.
    - `newArguments` - the concepts related by this relation.
    - `newOutputStartIndex` - the index into the newArguments array at which the output arcs start.

METHODS

- *copy*
  `public Relation` **copy(** `notio.CopyingScheme` **copyScheme)**
  - **Usage**
    - Performs a copy operation on this relation according to the the specified CopyingScheme. The result may be a new node or simply a reference to this relation depending on the scheme.
  - **Parameters**
    - `copyScheme` - the copying scheme used to control the copy operation.
  - **Returns** - the result of the copy operation.

- *copy*
  `public Relation` **copy(** `notio.CopyingScheme` **copyScheme,** `java.util.Hashtable` **substitutionTable)**

- **Usage**
  - Performs a copy operation on this relation according to the the specified CopyingScheme. The result may be a new node or simply a reference to this relation depending on the scheme.
- **Parameters**
  - `copyScheme` - the copying scheme used to control the copy operation.
  - `substitutionTable` - a hashtable containing copied objects available due to earlier copy operations.
- **Returns** - the result of the copy operation.

- *getArguments*
`public Concept` **`getArguments( )`**
  - **Usage**
    - Returns all of the concepts related by this relation. Input arguments are followed by output arguments. In a 'classic' relation, the last argument is always the sole output argument. If the relation has a type whose valence has changed since its definition, the arguments will not be adjusted. A call to isComplete() should be made to effect this adjustment.
  - **Returns** - the arguments to this relation.

- *getInputArguments*
`public Concept` **`getInputArguments( )`**
  - **Usage**
    - Returns the input arguments for this relation. If the relation has a type whose valence has changed since its definition, the arguments will not be adjusted. A call to isComplete() should be made to effect this adjustment.
  - **Returns** - an array, possibly empty, containing input arguments for this relation.

- *getOutputArguments*
`public Concept` **`getOutputArguments( )`**
  - **Usage**
    - Returns the output arguments for this relation. If the relation has a type whose valence has changed since its definition, the arguments will not be adjusted. A call to isComplete() should be made to effect this adjustment.
  - **Returns** - an array, possibly empty, containing output arguments for this relation.

- *getOutputStartIndex*
`public int` **`getOutputStartIndex( )`**
  - **Usage**
    - Returns the index of the first output argument within the array returned by getArguments(). This method will return -1 if the relation has a valence of zero. If there are inputs, but no outputs, the output start index will be 1 greater than the number of inputs. This means that any code using the output start index to index into the array returned by getArguments() should check first to see if the array is long enough.
  - **Returns** - the index of the first output argument within the array returned by getArguments().

  - **See Also**
    - `notio.Relation.getArguments()`

- *getType*
  `public RelationType` **getType( )**
  - **Usage**
    - Returns this relation's type.
  - **Returns** - this relation's type.

- *getValence*
  `public int` **getValence( )**
  - **Usage**
    - This method returns the valence (number of arguments) that this relation has defined. This number includes null arguments. If this relation's type has a defined valence, the number returned by this method will equal the valence of the type.
  - **Returns** - the number of arguments this relation has defined (including nulls).

- *isComplete*
  `public boolean` **isComplete( )**
  - **Usage**
    - Returns true if this relation's arguments are completely specified (are all non-null). If the relation has a type, its valence will be checked to ensure that the relation is complete according to the type. This check is made in case the valence of the type has changed. If it has changed, the arguments will be adjusted accordingly.
  - **Returns** - true if this relation's arguments are all non-null.

- *matchRelations*
  `public static boolean` **matchRelations(** `notio.Relation` **first,** `notio.Relation` **second,** `notio.MatchingScheme` **matchingScheme)**
  - **Usage**
    - Compares two relations to decide if they match. The exact semantics of matching are determined by the matching scheme.
  - **Parameters**
    - `first` - the first relation being matched.
    - `second` - the second relation being matched.
    - `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two concepts match according to the scheme's criteria.

- *relatesConcept*
  `public boolean` **relatesConcept(** `notio.Concept` **concept)**
  - **Usage**
    - Returns true if the specified concept is an argument of this relation.
  - **Parameters**
    - `concept` - the concept being checked for.
  - **Returns** - true if the specified concept is an argument of this relation.

- *replaceArgument*
  `public void` **replaceArgument(** `notio.Concept` **oldConcept,** `notio.Concept` **newConcept)**
  - **Usage**
    - Replaces all occurrences of the specified argument with a new one regardless of whether it is an input or output argument. Replacing nulls with non-nulls or vice-versa is allowed.
  - **Parameters**
    - `oldConcept` - the concept being replaced.
    - `newConcept` - the replacement concept.

- *replaceInputArgument*
  `public void` **replaceInputArgument(** `notio.Concept` **oldConcept,** `notio.Concept` **newConcept)**
  - **Usage**
    - Replaces all occurrences of the specified input argument with a new one. Replacing nulls with non-nulls or vice-versa is allowed. Replacing an argument with null effectively removes that argument.
  - **Parameters**
    - `oldConcept` - the input argument being replaced.
    - `newConcept` - the replacement argument.

- *replaceOutputArgument*
  `public void` **replaceOutputArgument(** `notio.Concept` **oldConcept,** `notio.Concept` **newConcept)**
  - **Usage**
    - Replaces all occurrences of the specified output argument with a new one. Replacing nulls with non-nulls or vice-versa is allowed. Replacing an argument with null effectively removes that argument.
  - **Parameters**
    - `oldConcept` - the output argument being replaced.
    - `newConcept` - the replacement argument.

- *restrictTo*
  `public Relation` **restrictTo(** `notio.RelationType` **subType)**
  - **Usage**
    - Returns a new node identical to this but restricted to the new type.
  - **Parameters**
    - `subType` - the type to be restricted to.
  - **Returns** - the new restricted relation.
  - **Exceptions**
    - `notio.RestrictionException` - if subType is not a real subtype of the current type

---

- *setArgument*
  `public void` **setArgument(** `int` **index,** `notio.Concept` **newConcept)**

- **Usage**
  - Sets the specified argument to the specified concept. The index of the first argument (often labelled "1" in diagrams) is zero. If the argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the specified index exceeds that valence, a IllegalArgumentException is thrown. If the valence is undefined and a previously unused argument index is specified, the valence of this particular relation will be increased to accomodate the index. No other relations nor the relation's type are affected by this increase. All currently specified arguments are preserved and any newly created but unspecified arguments are set to null. If the valence of the relation's type has changed since the arguments were specified, the arguments will be adjusted as if isComplete() had been called. Setting an argument to null effectively removes that argument.
- **Parameters**
  - `index` - the index of the argument being set.
  - `newConcept` - the concept to be used as an argument.

- *setArguments*
`public void` **setArguments**`( notio.Concept [] newConcepts )`

  - **Usage**
    - Sets the arguments according to the specified array of concepts. If an argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the number of arguments in the array exceeds that valence, a IllegalArgumentException is thrown. Otherwise, this method behaves like a series of calls to setArgument(), with the array index used as the argument's index.
  - **Parameters**
    - `newConcepts` - the array of concepts to be used as arguments.
  - **See Also**
    - `notio.Relation.setArgument`

- *setInputArgument*
`public void` **setInputArgument**`( int index, notio.Concept newConcept )`

  - **Usage**
    - Sets the specified argument to the specified concept. If the argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the specified index exceeds that valence, a IllegalArgumentException is thrown. If the valence is undefined and a previously unused argument index is specified, the valence of this particular relation will be increased to accomodate the index. No other relations nor the relation's type are affected by this increase. All currently specified arguments are preserved and any newly created but unspecified arguments are set to null. If the valence of the relation's type has changed since the arguments were specified, the arguments will be adjusted as if isComplete() had been called. Setting an argument to null effectively removes that argument.
  - **Parameters**
    - `index` - the index of the input argument being set (the nth input argument).
    - `newConcept` - the concept to be used as an argument.

- *setOutputArgument*
  `public void` **setOutputArgument(** `int` **index,** `notio.Concept` **newConcept)**
  - **Usage**
    - Sets the specified argument to the specified concept. If the argument has already been set, it will be replaced with the new value. If the valence of the relation type is defined and the specified index exceeds that valence, a IllegalArgumentException is thrown. If the valence is undefined and a previously unused argument index is specified, the valence of this particular relation will be increased to accomodate the index. No other relations nor the relation's type are affected by this increase. All currently specified arguments are preserved and any newly created but unspecified arguments are set to null. If the valence of the relation's type has changed since the arguments were specified, the arguments will be adjusted as if isComplete() had been called. Setting an argument to null effectively removes that argument.
  - **Parameters**
    - `index` - the index of the output argument being set (the nth output argument).
    - `newConcept` - the concept to be used as an argument.

- *setOutputStartIndex*
  `public void` **setOutputStartIndex(** `int` **newOutputStartIndex)**
  - **Usage**
    - Sets the index of the first output argument within the array returned by getArguments(). For relations with a valence of zero, the only valid start index is -1. For all other valences, the start index must be within the array's range unless there are no outputs, in which case the index may be 1 greater than the length of the array. Note that this method should rarely be needed since the start index can be specified in a constructor and it is unlikely to change. This method is provided chiefly for completeness.
  - **Parameters**
    - `newOutputStartIndex` - the index of the first output argument within the array returned by getArguments().
  - **See Also**
    - `notio.Relation.getArguments()`

- *setType*
  `public void` **setType(** `notio.RelationType` **newType)**
  - **Usage**
    - Sets the type for this relation. If a type has already been set, it will be replaced. If the new type has a valence that is different from the previous type and arguments have already been specified for this relation, the arguments will be updated as if isComplete() had been called.
  - **Parameters**
    - `newType` - the new type for this relation.
  - **See Also**
    - `notio.Relation.isComplete` (in 2.2.37, page 157)

METHODS INHERITED FROM CLASS `notio.Node`

(in 2.2.31, page 137)

- *getComment*
  `public String` **getComment**`( )`
  - **Usage**
    - Returns the comment string for this node.
  - **Returns** - the comment string associated with this node or null.

- *getEnclosingGraph*
  `public Graph` **getEnclosingGraph**`( )`
  - **Usage**
    - Returns the graph that encloses this node or null if the node does not currently belong to a graph.
  - **Returns** - the node's enclosing graph.

- *setComment*
  `public void` **setComment**`( java.lang.String` **newComment** `)`
  - **Usage**
    - Sets the comment string for this node.
  - **Parameters**
    - `newComment` - the new comment string for this node.

2.2.38 CLASS RelationAddError

Error thrown when the addition of a relation gives rise to an error.

DECLARATION

public class RelationAddError
extends notio.OperationError

CONSTRUCTORS

- *RelationAddError*
  `public` **RelationAddError**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an error with the specified message.
  - **Parameters**
    - `message` - The details of the error.

- *RelationAddError*
  `public` **RelationAddError**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**

    * Constructs an error with the specified message and sub-throwable.
  - **Parameters**
    * `message` - the details of the error.
    * `newSubThrowable` - the sub-throwable to be embedded in this error.

METHODS INHERITED FROM CLASS `notio.OperationError`

(in 2.2.33, page 141)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    * This method retrieves arbitrary throwables embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    * This method allows arbitrary throwables to be embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    * `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

METHODS INHERITED FROM CLASS `java.lang.Error`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.39 Class **RelationType**

The relation type class. This class encapsulates all available information about a relation type. The type can be defined by a label and/or a type definition or valence. Relation types provide the type information for both the relations and actors.

Declaration

public class RelationType
extends notio.Type
implements java.io.Serializable

Serializable Fields

- private String label
  - The type label for this type.
- private RelationTypeDefinition typeDefinition
  - The definition of this relation type.
- private int valence
  - The valence of this relation.
- private String comment
  - The comment associated with this type.

Constructors

- *RelationType*
  `public` **RelationType**`( )`
  - **Usage**
    - Constructs a RelationType with no label or type definition. The valence of the relation is automatically set to be unspecified. Note that this method does not add the type to any hierarchy.

- *RelationType*
  `public` **RelationType**`( notio.RelationTypeDefinition` **newDefinition** `)`
  - **Usage**
    - Constructs an unlabelled RelationType with the specified type definition. The valence of the type is automatically determined from the type definition. Note that this method does not add the type to any hierarchy.
  - **Parameters**
    - `newDefinition` - the type definition for this type.

- *RelationType*
  `public` **RelationType(** `java.lang.String` **newLabel)**
  - **Usage**
    - Constructs a labelled RelationType with the specified type label and no type definition. The valence of the relation is automatically set to be unspecified. Note that this method does not add the type to any hierarchy.
  - **Parameters**
    - `newLabel` - the type label for this type.

- *RelationType*
  `public` **RelationType(** `java.lang.String` **newLabel,** `int` **newValence)**
  - **Usage**
    - Constructs a labelled RelationType with the specified type label and valence. Note that this method does not add the type to any hierarchy.
  - **Parameters**
    - `newLabel` - the type label for this type.
    - `newValence` - the valence for this type.

- *RelationType*
  `public` **RelationType(** `java.lang.String` **newLabel,**
  `notio.RelationTypeDefinition` **newDefinition)**
  - **Usage**
    - Constructs a labelled RelationType with the specified type label and type definition. The valence of the type is automatically determined from the type definition. Note that this method does not add the type to any hierarchy.
  - **Parameters**
    - `newLabel` - the type label for this type.
    - `newDefinition` - the type definition for this type.

METHODS

- *getComment*
  `public String` **getComment( )**
  - **Usage**
    - Returns the comment string for this type.
  - **Returns** - the comment string associated with this type or null.

- *getImmediateSubTypes*
  `public RelationType` **getImmediateSubTypes( )**
  - **Usage**
    - Returns the immediate subtypes of this type.
  - **Returns** - an array of immediate subtypes of this type.

- *getImmediateSuperTypes*
  `public RelationType` **getImmediateSuperTypes( )**

  - **Usage**
    * Returns the immediate supertypes of this type.
  - **Returns** - an array of immediate supertypes of this type.

- *getLabel*
`public String` **getLabel( )**
  - **Usage**
    * Returns the type label for this type.
  - **Returns** - the string that is the label for this type.

- *getProperSubTypes*
`public RelationType` **getProperSubTypes( )**
  - **Usage**
    * Returns all subtypes of this type.
  - **Returns** - an array of subtypes of this type.

- *getSuperProperTypes*
`public RelationType` **getSuperProperTypes( )**
  - **Usage**
    * Returns all supertypes of this type.
  - **Returns** - an array of supertypes of this type.

- *getTypeDefinition*
`public RelationTypeDefinition` **getTypeDefinition( )**
  - **Usage**
    * Returns the relation type definition for this type (if any).
  - **Returns** - the type definition for this type or null if there isn't one.

- *getValence*
`public int` **getValence( )**
  - **Usage**
    * Returns the valence for this type, or -1 if the valence is undefined.
  - **Returns** - the valence for this type, or -1 if the valence is undefined.

- *hasProperSubType*
`public boolean` **hasProperSubType(** `notio.RelationType` **queryType)**
  - **Usage**
    * Tests whether the specified type is a proper subtype of this type.
  - **Parameters**
    * `queryType` - the type being tested.
  - **Returns** - true if queryType is a proper subtype of this type, false otherwise.

- *hasProperSuperType*
`public boolean` **hasProperSuperType(** `notio.RelationType` **queryType)**

  - **Usage**
    * Tests whether the specified type is a proper supertype of this type.
  - **Parameters**
    * `queryType` - the type being tested.
  - **Returns** - true if queryType is a proper supertype of this type, false otherwise.

- *hasSubType*
  `public boolean` **hasSubType**`( notio.RelationType` **queryType** `)`

  - **Usage**
    * Tests whether the specified type is a subtype of this type.
  - **Parameters**
    * `queryType` - the type being tested.
  - **Returns** - true if queryType is a subtype of this type, false otherwise.

- *hasSuperType*
  `public boolean` **hasSuperType**`( notio.RelationType` **queryType** `)`

  - **Usage**
    * Tests whether the specified type is a supertype of this type.
  - **Parameters**
    * `queryType` - the type being tested.
  - **Returns** - true if queryType is a supertype of this type, false otherwise.

- *matchRelationTypes*
  `public static boolean` **matchRelationTypes**`( notio.RelationType` **first**`, notio.RelationType` **second**`, notio.MatchingScheme` **matchingScheme** `)`

  - **Usage**
    * Compares two relation types to decide if they match. The exact semantics of matching are determined by the match control flag.
  - **Parameters**
    * `first` - the first relation type being matched.
    * `second` - the second relation type being matched.
    * `matchingScheme` - the matching scheme that determines how the match is performed.
  - **Returns** - true if the two relation types match according to the scheme's criteria.

- *setComment*
  `public void` **setComment**`( java.lang.String` **newComment** `)`

  - **Usage**
    * Sets the comment string for this type.
  - **Parameters**
    * `newComment` - the new comment string for this type.

- *setLabel*
  `public void` **setLabel**`( java.lang.String` **newLabel** `)`

  - **Usage**

    * Sets the type label for this type. If the type already had a label, it is replaced. If a null is used, the label is removed from the type.
  - **Parameters**
    * `newLabel` - the string that is the label for this type.
  - **Exceptions**
    * `notio.TypeChangeError` - if the type belongs to a hierarchy and the new label is already in use within it.

- *setTypeDefinition*
  `public void` **setTypeDefinition**`( notio.RelationTypeDefinition` **newDefinition** `)`

  - **Usage**
    * Sets the type definition for this type. If the type already has a definition, it is replaced. If null is used, the type definition will be removed. Specifying a type definition will override any existing valence information with the valence of the type definition.
  - **Parameters**
    * `newDefinition` - the new type definition for this type.

- *setValence*
  `public void` **setValence**`( int` **newValence** `)`

  - **Usage**
    * Sets the valence for this type. A value of -1 indicates that the valence is undefined. The valence is the number of arcs (or arguments) that this relation type possesses. Changing the valence has no effect on existing relations using this type except that they many return a different value when their isComplete() method is called. It is up to the application to ensure that all such relations are corrected by the additional or removal of arguments. In most cases, it is expected that this method will be used to change a valence from unspecified to a specific value, rather than from one specific value to another.
  - **Parameters**
    * `newValence` - the new valence for this type.
  - **Exceptions**
    * `notio.TypeChangeError` - if a type definition has been specified for this type.
  - **See Also**
    * `notio.Relation.isComplete()`

METHODS INHERITED FROM CLASS `notio.Type`

- *getLabel*
  `public abstract String` **getLabel**`( )`

  - **Usage**
    * Returns the type label for this type.
  - **Returns** - the type label for this type.

2.2.40 Class **RelationTypeDefinition**

The relation type definition class. This class provides the functionality of a lambda expression used for describing a relation type.

Declaration

public class RelationTypeDefinition
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private Abstraction abstraction
  - The abstraction used in this definition.
- private ConceptType signature
  - The signature used in this definition.

Constructors

- *RelationTypeDefinition*
  `public` **RelationTypeDefinition**`( notio.Concept [] newParameters, notio.Graph newRelator )`
  - **Usage**
    - Constructs a new relation type definition with the specified relator graph and the array of formal parameter concepts. The formal parameter concepts must be part of the graph and have a null referent. The signature of this definition is derived from the formal parameters.
  - **Parameters**
    - `newParameters` - the array of formal parameter concepts used in this definition.
    - `newRelator` - the relator graph for this definition.

- *RelationTypeDefinition*
  `public` **RelationTypeDefinition**`( notio.ConceptType [] newSignature )`
  - **Usage**
    - Constructs a new relation type definition with the specified signature.
  - **Parameters**
    - `newSignature` - the array of concept types that form the signature of this definition.

METHODS

- *getFormalParameters*
  `public Concept` **`getFormalParameters( )`**
  - **Usage**
    - Returns the formal parameter concepts for this definition or null if no formal parameters have been specified.
  - **Returns** - the formal parameter concepts for this definition.

- *getRelator*
  `public Graph` **`getRelator( )`**
  - **Usage**
    - Returns the relator graph for this definition or null if no relator has been specified.
  - **Returns** - the relator graph for this definition.

- *getSignature*
  `public ConceptType` **`getSignature( )`**
  - **Usage**
    - Returns the signature for this definition. The signature consists of the concept types of the formal parameters. No relator graph need have been specified in order to have a signature.
  - **Returns** - the signature for this definition.

- *getValence*
  `public int` **`getValence( )`**
  - **Usage**
    - Returns the valence (number of formal parameters) for this definition.
  - **Returns** - the valence for this definition.

2.2.41 CLASS RelationTypeHierarchy

The relation type hierarchy class.

DECLARATION

public class RelationTypeHierarchy
extends notio.TypeHierarchy
implements java.io.Serializable

FIELDS

- public static final String UNIVERSAL_TYPE_LABEL
  - The predefined type label for the universal type.
- public static final String ABSURD_TYPE_LABEL
  - The predefined type label for the absurd type.

CONSTRUCTORS

- *RelationTypeHierarchy*
  `public` **RelationTypeHierarchy**`( )`
  - **Usage**
    - Constructs a new RelationTypeHierarchy.

METHODS

- *addSubTypesToType*
  `public void` **addSubTypesToType**`( notio.RelationType` **subjectType**`, notio.RelationType [] newSubTypes )`
  - **Usage**
    - Adds the specified list of subtypes as sub types to the subject type.
  - **Parameters**
    - `subjectType` - the type having subtypes added.
    - `newSubTypes` - the array of types to be added as subtypes.
  - **Exceptions**
    - `notio.TypeChangeError` - if specified subtypes are not in hierarchy, or if addition of subtypes creates a type order conflict.

- *addSubTypeToType*
  `public void` **addSubTypeToType**`( notio.RelationType` **subjectType**`, notio.RelationType` **newSubType** `)`
  - **Usage**
    - Adds the specified subtype to the subject type.
  - **Parameters**
    - `subjectType` - the type having a subtype added.
    - `newSubType` - the subtype to be added.
  - **Exceptions**
    - `notio.TypeChangeError` - if specified subtype is not in hierarchy or, if addition of the subtype creates a type order conflict.

- *addSuperTypesToType*
  `public void` **addSuperTypesToType**`( notio.RelationType` **subjectType**`, notio.RelationType [] newSuperTypes )`

- **Usage**
  - Adds the specified list of supertypes as super types to the subject type.
- **Parameters**
  - `subjectType` - the type having supertypes added.
  - `newSuperTypes` - the array of types to be added as supertypes.
- **Exceptions**
  - `notio.TypeChangeError` - if specified supertypes are not in hierarchy, or if addition of supertypes creates a type order conflict.

- *addSuperTypeToType*
  `public void` **addSuperTypeToType**`( notio.RelationType` **subjectType,** `notio.RelationType` **newSuperType** `)`
  - **Usage**
    - Adds the specified supertype to the subject type.
  - **Parameters**
    - `subjectType` - the type having a supertype added.
    - `newSuperType` - the supertype to be added.
  - **Exceptions**
    - `notio.TypeChangeError` - if specified supertype is not in hierarchy or, if addition of the supertype creates a type order conflict.

- *addTypeToHierarchy*
  `public void` **addTypeToHierarchy**`( notio.RelationType` **newType** `)`
  - **Usage**
    - Adds a new type to this hierarchy with the Universal type as its only supertype, and the Absurd type as its only subtype. Note that unlike the other versions of this method, this does not throw notio.TypeChangeError since the Universal and Absurd types are always present in the hierarchy.
  - **Parameters**
    - `newType` - the type being added.
  - **Exceptions**
    - `notio.TypeAddError` - if type label is already in use by another type.

- *addTypeToHierarchy*
  `public void` **addTypeToHierarchy**`( notio.RelationType` **newType,** `notio.RelationType [] supertypes, notio.RelationType [] subtypes )`
  - **Usage**
    - Adds a new type to the hierarchy with the specified supertypes and subtypes.
  - **Parameters**
    - `newType` - the type being added.
    - `supertypes` - the array of supertypes to be used, or null (assumes universal type as only supertype).
    - `subtypes` - the array of subtypes to be used, or null (assumes absurd type as only subtype).
  - **Exceptions**
    - `notio.TypeAddError` - if type label is already in use by another type.

    * `notio.TypeChangeError` - if specified supertypes or subtypes are not in hierarchy, or if addition of type creates a type order conflict.

- *addTypeToHierarchy*
  `public void` **addTypeToHierarchy(** `notio.RelationType` **newType,** `notio.RelationType` **supertype,** `notio.RelationType` **subtype)**
  - **Usage**
    * Adds a new type to this hierarchy with the specified supertype and subtype.
  - **Parameters**
    * `newType` - the type being added.
    * `supertype` - the single supertype to be used, or null (assumes universal type as subtype).
    * `subtype` - the single subtype to be used, or null (assumes absurd type as subtype).
  - **Exceptions**
    * `notio.TypeAddError` - if type label is already in use by another type.
    * `notio.TypeChangeError` - if specified supertypes or subtypes are not in hierarchy, or if addition of a type creates a type order conflict.

- *getCaseSensitiveLabels*
  `public boolean` **getCaseSensitiveLabels( )**
  - **Usage**
    * Returns true if the processing of type labels in this hierarchy is case-sensitive.
  - **Returns** - true if the processing of type labels in this hierarchy is case-sensitive.

- *getImmediateSubTypesOf*
  `public RelationType` **getImmediateSubTypesOf(** `notio.RelationType` **subjectType)**
  - **Usage**
    * Returns the immediate subtypes of the subject type in no particular order.
  - **Parameters**
    * `subjectType` - the type whose immediate subtypes will be returned.
  - **Returns** - an array of the immediate subtypes of the subject type, possibly empty.

- *getImmediateSuperTypesOf*
  `public RelationType` **getImmediateSuperTypesOf(** `notio.RelationType` **subjectType)**
  - **Usage**
    * Returns the immediate subtypes of the subject type in no particular order.
  - **Parameters**
    * `subjectType` - the type whose immediate subtypes will be returned.
  - **Returns** - an array of the immediate subtypes of the subject type, possibly empty.

- *getProperSubTypesOf*
  `public RelationType` **getProperSubTypesOf(** `notio.RelationType` **subjectType)**
  - **Usage**

    * Returns all the subtypes of the subject type, not just the immediate subtypes, in no particular order.
  - **Parameters**
    * `subjectType` - the type whose subtypes will be returned.
  - **Returns** - an array of the subtypes of the subject type, possibly empty.

- *getProperSuperTypesOf*
  `public RelationType` **getProperSuperTypesOf(** `notio.RelationType` **subjectType)**
  - **Usage**
    * Returns all the supertypes of the subject type, not just the immediate supertypes, in no particular order.
  - **Parameters**
    * `subjectType` - the type whose supertypes will be returned.
  - **Returns** - an array of the supertypes of the parent type, possibly empty.

- *getTypeByLabel*
  `public RelationType` **getTypeByLabel(** `java.lang.String` **label)**
  - **Usage**
    * Looks up and returns the type object associated with the specified label.
  - **Parameters**
    * `label` - the label used to lookup.
  - **Returns** - the type associated with the label or null if non-existant.

- *getUnlabelledTypes*
  `public RelationType` **getUnlabelledTypes( )**
  - **Usage**
    * Returns all unlabelled types in this hierarchy in no particular order.
  - **Returns** - an array of the unlabelled types in this hierarchy, possibly empty.

- *isProperSubTypeOf*
  `public boolean` **isProperSubTypeOf(** `notio.RelationType` **subject,** `notio.RelationType` **object)**
  - **Usage**
    * Determines whether subject is a subtype of object.
  - **Parameters**
    * `subject` - the potential subtype being tested.
    * `object` - the potential supertype being tested.
  - **Returns** - true if subject is a subtype of object.

- *isProperSuperTypeOf*
  `public boolean` **isProperSuperTypeOf(** `notio.RelationType` **subject,** `notio.RelationType` **object)**
  - **Usage**
    * Determines whether subject is a proper supertype of object.

  - **Parameters**
    * `subject` - the potential supertype being tested.
    * `object` - the potential subtype being tested.
  - **Returns** - true if subject is a supertype of object.

- *isSubTypeOf*
`public boolean` **isSubTypeOf(** `notio.RelationType` **subject,** `notio.RelationType` **object)**
  - **Usage**
    * Determines whether subject is a subtype of object.
  - **Parameters**
    * `subject` - the potential subtype being tested.
    * `object` - the potential supertype being tested.
  - **Returns** - true if subject is a subtype of object.

- *isSuperTypeOf*
`public boolean` **isSuperTypeOf(** `notio.RelationType` **subject,** `notio.RelationType` **object)**
  - **Usage**
    * Determines whether subject is a supertype of object.
  - **Parameters**
    * `subject` - the potential supertype being tested.
    * `object` - the potential subtype being tested.
  - **Returns** - true if subject is a supertype of object.

- *removeTypeFromHierarchy*
`public void` **removeTypeFromHierarchy(** `notio.RelationType` **deadType)**
  - **Usage**
    * Removes a type from this hierarchy. Removing a type from its hierarchy does not remove it from any of the nodes in which it may be used. This must be done manually if it is necessary. In most cases it is expected that types are being replaced rather than actually removed entirely. In this case, it is much easier to simply the change the characteristics of the existing type. If a type is being replaced by more than one type, it will naturally be necessary to perform at least part of the replacement manually. If replacements must be performed, the utility method replaceTypeInGraph() may be used.
  - **Parameters**
    * `deadType` - the type being removed.
  - **Exceptions**
    * `notio.TypeRemoveError` - if specified type is not in hierarchy
  - **See Also**
    * `notio.RelationTypeHierarchy.replaceTypeInGraph`

- *replaceTypeInGraph*
`public static void` **replaceTypeInGraph(** `notio.Graph` **graph,** `notio.RelationType` **oldType,** `notio.RelationType` **newType)**

  - **Usage**
    * Traverses the specified graph, replacing all occurrences of the specified type with a replacement type. Either of these types may be null. If the old type is null, untyped relations will be set to the new type. If the new type is null, relations using the old type will become untyped. In most cases where a non-null type is replacing a non-null type, it is probably more efficient to modify the old type directly and avoid the use of this method.

- *setCaseSensitiveLabels*
  `public void` **setCaseSensitiveLabels**`( boolean` **flag** `)`
  - **Usage**
    * Sets a flag indicating whether or not the processing of type labels within this hierarchy is case-sensitive.
  - **Parameters**
    * `flag` - the flag setting for case-sensitivity.

METHODS INHERITED FROM CLASS `notio.TypeHierarchy`

2.2.42 CLASS **RestrictionException**

Exception thrown when an invalid restrict operation is attempted.

DECLARATION

---

public class RestrictionException **extends** notio.OperationException

CONSTRUCTORS

- *RestrictionException*
  `public` **RestrictionException**`( java.lang.String` **message** `)`
  - **Usage**
    * Constructs an exception with the specified message.
  - **Parameters**
    * `message` - The details of the exception.

- *RestrictionException*
  `public` **RestrictionException**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    * Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    * `message` - the details of the exception.
    * `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.43 CLASS SimpleMatchGenerator

A generator class used to extract SimpleMatches from a MatchResult one at a time.

DECLARATION

public class SimpleMatchGenerator
extends java.lang.Object

CONSTRUCTORS

- *SimpleMatchGenerator*
  `public` **SimpleMatchGenerator**`( notio.MatchResult` **matchResult** `)`
  - **Usage**
    - Constructs a new generator that will process the specified MatchResult.
  - **Parameters**
    - `matchResult` - the MatchResult to be processed.

METHODS

- *getNextSimpleMatch*
  `public MatchResult` **getNextSimpleMatch**`( )`
  - **Usage**
    - Gets the next simple match from this generator.
  - **Returns** - the next simple match from this generator or null if there are no more.
- *getRemainingMatches*
  `public MatchResult` **getRemainingMatches**`( )`
  - **Usage**
    - Gets all remaining simple matches from this generator.
  - **Returns** - an array of all simple matches remaining in this generator, possibly empty.
- *resetGenerator*
  `public void` **resetGenerator**`( )`
  - **Usage**
    - Resets the generator that will generate from the beginning again.

2.2.44 CLASS TranslationContext

A class for holding information related to translation so it can be given to different parsers and generators to ensure consistent representations. A TranslationContext consists of a set of TranslationInfoUnits. Translators can add, retrieve, and remove TranslationInfoUnits from a context. The context has no understanding about the nature of the units it carries and only a particular translator knows which units it needs and understands. Translators must agree on a TranslationInfoUnit name and sub-interface in order to share information. Note that the term 'context' here does not refer to the conceptual graph idea of a context.

Declaration

public class TranslationContext
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private Hashtable unitTable
  - A hashtable mapping unit names to the TranslationInfoUnits that form this context.

Constructors

- *TranslationContext*
  `public` **TranslationContext**`( )`
  - **Usage**
    - Constructs a new TranslationContext containing no TranslationInfoUnits.

Methods

- *addUnit*
  `public void` **addUnit**`( notio.TranslationInfoUnit` **newUnit** `)`
  - **Usage**
    - Adds the specified unit to this context under the unit's name. The name is determined by a call to getUnitName() in the unit. If the unit is already part of this context, nothing happens. If another unit has been added to this context under the same name, it is replaced by the new unit.
  - **Parameters**
    - `newUnit` - the unit to be added.
  - **Exceptions**
    - `java.IllegalArgumentException` - if the unit's name is null.
  - **See Also**
    - `notio.TranslationInfoUnit.getUnitName` (in 2.1.6, page 48)

- *getUnit*
  `public TranslationInfoUnit` **getUnit**`( java.lang.String` **unitName** `)`
  - **Usage**
    - Returns the unit corresponding to the specified name, or null if no unit with that name is currently in this context.
  - **Parameters**
    - `unitName` - the name of the unit being retrieved.

- *removeUnit*
  `public void` **removeUnit(** `notio.TranslationInfoUnit` **deadUnit)**
  - **Usage**
    - Removes the specified unit from this context. If the unit is not part of this context, nothing happens.
  - **Parameters**
    - `deadUnit` - the unit to be removed.

- *resetUnits*
  `public void` **resetUnits( )**
  - **Usage**
    - Calls resetUnit() in all units in this context. This effectively clears all information relating to earlier translation sessions from this context.
  - **See Also**
    - `notio.TranslationInfoUnit.resetUnit` (in 2.1.6, page 48)

2.2.45 Class TranslationException

The base class for all exceptions thrown during translation (parsing or generating). Contains fields for translation-specific details.

Declaration

public class TranslationException
extends notio.OperationException

Constructors

- *TranslationException*
  `public` **TranslationException(** `java.lang.String` **message)**
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.

- *TranslationException*
  `public` **TranslationException(** `java.lang.String` **message,** `java.lang.Throwable` **newSubThrowable)**
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.
    - `newSubThrowable` - the sub-throwable to be embedded in this exception.

METHODS INHERITED FROM CLASS `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.46 CLASS TypeAddError

Error thrown when the addition of a type gives rise to an error.

Declaration

public class TypeAddError
extends notio.OperationError

Constructors

- *TypeAddError*
  `public` **TypeAddError(** `java.lang.String` **message)**
  - **Usage**
    - Constructs an error with the specified message.
  - **Parameters**
    - `message` - The details of the error.

- *TypeAddError*
  `public` **TypeAddError(** `java.lang.String` **message,** `java.lang.Throwable` **newSubThrowable)**
  - **Usage**
    - Constructs an error with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the error.
    - `newSubThrowable` - the sub-throwable to be embedded in this error.

Methods inherited from class `notio.OperationError`

(in 2.2.33, page 141)

- *getSubThrowable*
  `public Throwable` **getSubThrowable(** **)**
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable(** `java.lang.Throwable` **newSubThrowable)**
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

Methods inherited from class `java.lang.Error`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.47 CLASS TypeChangeError

Error thrown when a change to a type gives rise to an error.

DECLARATION

public class TypeChangeError
extends notio.OperationError

CONSTRUCTORS

- *TypeChangeError*
  `public` **TypeChangeError**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an error with the specified message.
  - **Parameters**
    - `message` - The details of the error.
- *TypeChangeError*
  `public` **TypeChangeError**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an error with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the error.
    - `newSubThrowable` - the sub-throwable to be embedded in this TypeChangeError.

METHODS INHERITED FROM CLASS `notio.OperationError`

(in 2.2.33, page 141)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

METHODS INHERITED FROM CLASS `java.lang.Error`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.48 CLASS TypeExpansionException

Exception thrown when a type expansion gives rise to an error.

Declaration

public class TypeExpansionException
extends notio.OperationException

Constructors

- *TypeExpansionException*
  `public` **TypeExpansionException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - The details of the exception.

- *TypeExpansionException*
  `public` **TypeExpansionException**`( java.lang.String` **message**`,`
  `java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an exception with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the exception.
    - `newSubThrowable` - the sub-throwable to be embedded in this exception.

Methods inherited from class `notio.OperationException`

(in 2.2.34, page 142)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.

- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationExceptions and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationException.

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.49 CLASS TypeRemoveError

Error thrown when the removal of a type gives rise to an error.

DECLARATION

public class TypeRemoveError
extends notio.OperationError

CONSTRUCTORS

- *TypeRemoveError*
  `public` **TypeRemoveError**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an error with the specified message.
  - **Parameters**
    - `message` - The details of the error.
- *TypeRemoveError*
  `public` **TypeRemoveError**`( java.lang.String` **message**`, java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - Constructs an error with the specified message and sub-throwable.
  - **Parameters**
    - `message` - the details of the error.
    - `newSubThrowable` - the sub-throwable to be embedded in this error.

METHODS INHERITED FROM CLASS `notio.OperationError`

(in 2.2.33, page 141)

- *getSubThrowable*
  `public Throwable` **getSubThrowable**`( )`
  - **Usage**
    - This method retrieves arbitrary throwables embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API.
  - **Returns** - the sub-throwable or null if none is present.
- *setSubThrowable*
  `public void` **setSubThrowable**`( java.lang.Throwable` **newSubThrowable** `)`
  - **Usage**
    - This method allows arbitrary throwables to be embedded inside OperationErrors and thrown along with them. The embedded throwables can give further details of what happened or be used to pass implementation-specific throwables through the standard Notio API. This method will replace any existing sub-throwable.
  - **Parameters**
    - `newSubThrowable` - the sub-throwable to be embedded in this OperationError.

METHODS INHERITED FROM CLASS `java.lang.Error`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.50 CLASS UnimplementedFeatureException

This exception is thrown whenever a feature or operation is unavailable in a given Notio implementation. It is included primarily so that partial implementations may be safely distributed. Since it is a subclass of java.lang.RuntimeException, this exception need not be declared in the throws clause of any method and need not be explicitly caught. Use of this exception should be kept to a minimum and it should never be used to report any other type of error.

DECLARATION

public class UnimplementedFeatureException
extends java.lang.RuntimeException

CONSTRUCTORS

- *UnimplementedFeatureException*
  `public` **UnimplementedFeatureException**`( java.lang.String` **message** `)`
  - **Usage**
    - Constructs an exception with the specified message.
  - **Parameters**
    - `message` - the details of the exception.

METHODS INHERITED FROM CLASS `java.lang.RuntimeException`

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public native Throwable` **fillInStackTrace**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter )`
- *toString*
  `public String` **toString**`( )`

2.2.51 CLASS UnimplementedMacro

Class used to create a dummy for unimplemented macros.

Declaration

public class UnimplementedMacro
extends java.lang.Object
implements Macro, java.io.Serializable

Serializable Fields

- private String name
  - The name by which this macro is known.

Constructors

- *UnimplementedMacro*
  `public` **UnimplementedMacro**`( java.lang.String` **newName** `)`
  - **Usage**
    - Constructs an dummy macro with the specified name.
  - **Parameters**
    - `newName` - the name by which this macro is known.

Methods

- *executeMacro*
  `public Object` **executeMacro**`( java.lang.Object [] args )`
  - **Usage**
    - UnimplementedMacros are dummy macros that do nothing when implemented. Perhaps they should throw an UnimplementedOperationException exception?
  - **Parameters**
    - `args` - these are completely ignored.
  - **Returns** - null since this macro doesn't do anything.

- *getName*
  `public String` **getName**`( )`
  - **Usage**
    - Returns the name of the macro.
  - **Returns** - the name of the macro.