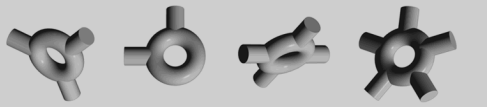
elements



Utilities Package

TARMS Inc.

September 07, 2000

Typeset in LaTeX.

Contents

List of Figures

List of Tables

Package Description

This package contains a number of utility interfaces that can be used throughout the object model.

In many cases, the interfaces will be present in the target language and references to these interfaces will need to be translated into the appropriate interface definitions in the target language. For example the Hashable interface is not strictly needed in Java, as all objects implement hashCode(), a method that can be used in place of hash().

Included in this package is a means for handling identifiers based on standards documents, eg., the ISO 3166 country codes. These identifiers carry information on the relevant standard, as well as the identifier.

Also included in this package is a mechanism for handling *enumerations*. This mechanism is intended for languages without explicit support of enumerations (eg., Java). These classes can be translated into appropriate enumeration types in languages that do support such features (eg., C++). Enumerations have been given the stereotype «Enumeration» and (usually) are subclasses of the Enum §4.1 abstract class.[1]

1 Interfaces

1.1 ClassRepresentation

This interfaces represents a class, for example the “Book” class. This interface is primarily used to determine whether a given object is an instance of the represented class.

[1] There is no associated Enum interface. Enumerations are intended to be provided as an implementation convienience, rather than as part of the overall behavioral model.

1.1.1 Relationships

	Class	Description	Notes
↓	StringClassRepresentationModel §2.4		
↔	DualSpecificationCollection-Model §2.2	classes	

↓:Realized by ↔:Association →:Navigable ◊:Aggregate ♦:Composite

1.1.2 Operations

String className() className

This method returns the name of the class that this object represents.

Boolean isClassOf(Object object) isClassOf

object: Object The object which is being tested to see whether it is the same class as represented by this object.

This method determines whether the class that this interface represents is the class of the given object. If the "class name" of the given object equals this object's className then this method returns true.

The way that this method gets the name of the class of a given object will depend upon the target language. An implementation in Smalltalk or Java can achieve this natively. An implementation in C++ would have to define an abstract superclass with a virtual "classname" method, and make every class that could be represented by this object inherit from it.

Collection<Object> allInstances() allInstances

This method returns all the instances of the class that is represented by this object. This will most likely be achieved via a call to a database.

1.2 Classifier

This interface provides the ability to add key-value pairs to a collection, and to extract the values corresponding to a particular key. The purpose of the Classifier is to hold onto certain information about a particular object that will classify/identify it.

1.2.1 Relationships

	Class	Description	Notes
↓	ClassifierModel §2.1		

↓:Realized by

1.2.2 Operations

add(Object key, DualSpecificationCollection value) add

key: Object This key provides an identifier to which a value can be linked with, in the classifier.

value: DualSpecificationCollection This is the value that will be linked with the given key, and held in the classifier.

This method accepts a key-value pair which will be added to this classifier. The value part of this key-value pair will be a DualSpecificationCollection.

Collection<Object> at(Object key) at

key: Object This key identifies an entry within the dictionary held by this classifier.

This method retrives the value at the given key (which will be a DualSpecificationCollection), sends the message “asInstances” to it, and returns that collection. If the given key is not present in this classifier then null is returned.

Boolean isASupersetOf(Classifier classifier) isASupersetOf

classifier: Classifier A classifier which will be tested to see whether it is a subset of this classifier.

This method determines whether this Classifier encompasses (ie. is a superset of) a given classifier.

If the given classifier contains keys that do not exist in this classifier then return false. Otherwise, for every key in the given classifier obtain the corresponding value (which will be a DualSpecificationCollection) in both the given classifier and this classifier. If all the DualSpecificationCollections from this classifier are supersets of the corresponding DualSpecificationCollections in the given classifier then return true, otherwise return false.

1.3 DualSpecificationCollection

This interface is used to represent a collection of objects. The represented objects can be specified in either of two ways. The first way is to directly specify the object. This is done by inserting the object into the "instances" collection. The second way is to specify a class, of which all instances are considered a part of this collection.

1.3.1 Relationships

	Class	Description	Notes
↓	DualSpecificationCollection-Model §2.2		

↓:Realized by

1.3.2 Operations

Collection<Object> instances() instances

This method returns a collection of the objects that make up part of this Dual-SpecificationCollection.

Collection<ClassRepresentation> classes() classes

This method returns a collection of the classes (ClassRepresentations) that make up part of this DualSpecificationCollection.

Collection<Object> asInstances() asInstances

This method converts this DualSpecificationCollection into a collection of all the instances that it represents. This method returns the union of the following two collections:
1) the collection of objects returned from the "instances" method,
2) the union of all the collections returned from sending the "allInstances" message to each of the ClassRepresentations returned from the "classes" method.

Boolean isASuperSetOf(DualSpecificationCollection dualSpecification-Collection) isASuperSetOf
dualSpecificationCollection: DualSpecificationCollection

This method determines whether this DualSpecificationCollection is a superset of the given dualSpecificationCollection.

If this object's collection of classes is a superset of the given parameter's collection of classes, and the given parameter's collection of instances is either:
a) a subset of this collection's instances, or
b) is of a class that is in this object's collection of classes (check by calling the "isClassOf" method),
then return true, otherwise false.

1.4 Hashable

An object that can generate some sort of hash value for itself. These objects can be inserted into containers that are implemented as hash tables - commonly sets and dictionaries.

1.4.1 Relationships

	Class	Description	Notes
⇓	Comparable §1.5		

⇓:Inherited by

1.4.2 Operations

Integer hash() hash

Hash value for an object. Return a suitable hash value for the object. Hash values can be any computation that consistently returns the same value for the object and is suitably widely dispersed.

1.5 Comparable

An interface for objects with a notion of equality. Most objects are identically equal (same object). Objects, such as mathematical objects, which have a more sophisticated concept of equality should implement this interface.

1.5.1 Relationships

	Class	Description	Notes
⇑	Hashable §1.4		
⇓	PartiallyOrdered §1.6		
⇓	Responsible §1.12		
↓	Enum §4.1		

⇑:Inherits ⇓:Inherited by ↓:Realized by

1.5.2 Operations

Boolean equals(Comparable arg) equals

arg: Comparable The object to compare this object against.

The equality relationship. Returns true if two objects are equal, false otherwise. Any Comparable object can be supplied as an argument; implementors are expected to perform type-checking to avoid runtime errors.

This signature may have several different names, depending upon target language. Examples are = (Smalltalk), == (C++) and equals (Java).

Boolean notEquals(Comparable arg) notEquals

arg: Comparable The object to compare this object against.

The inequality relationship.

$$\neg(a = b) \Leftrightarrow a \text{ !=} b$$

This signature may have several different names, depending on target language. Examples are != (Smalltalk) and != (C++)

Boolean strictlyEquals(Comparable arg) strictlyEquals

arg: Comparable The object to compare this object against.

Raises: IncomparableException

A strict version of the equality relationship. A strict version of equality, where IncomparableException is raised if the two objects are not of the same type, as opposed to returning false.

Integer hash() hash

Hash value for an object. An additional restriction on hash functions for Comparable is

$$(a = b) \Rightarrow (hash(a) = hash(b))$$

1.6 PartiallyOrdered

An interface for objects with a partial "less-than" relationship. Some instances of these objects may not be comparable. For example, sets are partially ordered, using subset as an ordering. $\{1, 2\} \leq \{1, 2, 3\}$. However neither $\{1, 3\} \leq \{1, 2\}$ or $\{1, 2\} \leq \{1, 3\}$.

1.6.1 Relationships

	Class	Description	Notes
⇑	Comparable §1.5		
⇓	TotallyOrdered §1.7		

⇑:Inherits ⇓:Inherited by

1.6.2 Operations

Boolean lessThanOrEqualTo(PartiallyOrdered arg) lessThanOrEqualTo

arg: PartiallyOrdered The object to compare this object against.

The less than or equal to relationship. Compares the object against arg, returning true if this object is less than or equal to arg.

If two objects that implement PartiallyOrdered, but are incomparable, are compared using lessThanOrEqualTo, then false is returned.

Depending on target language, this operation may use the <= symbol.

1.7 TotallyOrdered

A refinement of partial ordering, where $\neg a \leq b$ implies that $a > b$.

1.7.1 Relationships

	Class	Description	Notes
⇑	PartiallyOrdered §1.6		
⇓	Datestamp §1.8		
⇓	Identifiable §1.10		
↓	OrderedEnum §4.2		

⇑:Inherits ⇓:Inherited by ↓:Realized by

1.7.2 Operations

Boolean lessThanOrEqualTo(TotallyOrdered arg) lessThanOrEqualTo
arg: TotallyOrdered The object to compare this object against.
Raises: IncomparableException

The less than or equal to relationship. Similar to the lessThanOrEqualTo defined for PartiallyOrdered. However, if two incomparable objects are compared, an IncomparableException is raised.

Depending on target language, this operation may use the <= symbol.

Boolean greaterThan(TotallyOrdered arg) greaterThan
arg: TotallyOrdered The object to compare this object against.
Raises: IncomparableException

The strictly greater than relationship.

$$\neg(a \leq b) \Leftrightarrow (a > b)$$

Depending on target language, this operation may use the > symbol.

Boolean greaterThanOrEqualTo(TotallyOrdered arg)
arg: TotallyOrdered The object to compare this object against. greaterThanOrEqualTo
Raises: IncomparableException

The greater than or equal to relationship.

$$(a \geq b) \Leftrightarrow (b \leq a)$$

Depending on target language, this operation may use the >= symbol.

Boolean lessThan(TotallyOrdered arg) lessThan
arg: TotallyOrdered The object to compare this object against.
Raises: IncomparableException

The strictly less than relationship.

$$\neg(a \leq b) \Leftrightarrow (b < a)$$

Depending on target language, this operation may use the < symbol.

1.8 Datestamp

Datestamps hold information about the date that some event occurred upon. Generally, Datestamps provide no date arithmetic beyond the ability to compare dates and provide a readable representation.

Datestamps and Timestamps may already have an implementation in the target language.

1.8.1 Relationships

	Class	Description	Notes
⇑	ValueSemantics §1.19		
⇑	TotallyOrdered §1.7		
⇓	Timestamp §1.9		

⇑:Inherits ⇓:Inherited by

1.8.2 Operations

print(OutputStream stream) print

stream: OutputStream The stream to print the date stamp onto.

Print the date. A user-readable version of the date is added to the output stream. The current local date printing conventions are used.

«Static Method» Datestamp currentProcessingDate() currentProcessingDate

System processing date. Returns the current processing date. The current processing date is not necessarily the physical current date. It is, instead, the notional processing date for the system.

Boolean equals(Comparable arg) equals

arg: Comparable The object to compare this object against.

The equality relationship. Two Datestamps are equal if they represent the same calendar date.

Boolean lessThanOrEqualTo(PartiallyOrdered arg) lessThanOrEqualTo
arg: PartiallyOrdered The object to compare this object against.

The less than or equal to relationship. Datestamps are totally ordered by the calendar dates that they represent.

1.9 Timestamp

Timestamps extend Datestamp interfaces to include the time of an event as well as its date. Timestamps are accurate to — at least — the nearest second.

Timestamps are always associated with some time zone.

1.9.1 Relationships

	Class	Description	Notes
⇑	Datestamp §1.8		

⇑:Inherits

1.9.2 Operations

Boolean equals(Comparable arg) equals
arg: Comparable The object to compare this object against.

Equality relationship. Timestamps are generally compared for equality and ordering independent of time zone. Eg. 6-Apr-1999 14:05 EST (Australia/NSW) is equal to 6-Apr-1999 04:05 GMT.

Boolean lessThanOrEqualTo(Datestamp arg) lessThanOrEqualTo
arg: Datestamp The object to compare this object against.
Raises: IncomparableException

Less than or equal to relationship. Timestamps are generally compared for equality and ordering independent of time zone. Eg. 6-Apr-1999 14:05 EST (Australia/NSW) is less than 6-Apr-1999 04:10 GMT.

Boolean strictlyEquals(Comparable arg) strictlyEquals
arg: Comparable The object to compare this object against.
Raises: IncomparableException

Strict equality relationship. The strictlyEquals method compares time zones as well as the actual times. Eg. 6-Apr-1999 14:05 EST (Australia/NSW) is not

strictly equal to 6-Apr-1999 04:05 GMT.

print(OutputStream stream) print

stream: OutputStream The stream to print onto.

Print the Timestamp. A user-readable version of the date and time is added to the stream. The current local date and time printing conventions are used.

String timeZone() timeZone

Timestamp's time zone. Returns a string description of the Timestamp's time zone.

Integer offsetFromGMT() offset-FromGMT

Timezone's offset from GMT. Returns the number of seconds that this time zone is offset from GMT. Positive amounts indicate that the time zone is ahead of GMT, negative amounts indicate that the time zone is behind GMT.

Note that this method must take account of daylight saving and any other time adjustments associated with the time zone.

Note that the Smalltalk TimeZone class is not adequate to handle Southern hemisphere daylight saving time.

«Static Method» Timestamp localStamp() localStamp

Current local time. Returns the current time, using the local time zone.

«Static Method» Timestamp gmtStamp() gmtStamp

Current GMT time. Returns the current time, using the GMT time zone.

«Static Method» Timestamp now() now

Current time. Returns the current time using whatever default time zone the system uses.

1.10 Identifiable

An Identifiable is an object that belongs to a family of objects, each member of the family having a unique identification string. The identification string is usually some standardized code for the object — eg., the RIC codes for organizations, or the ISO currency names.

1.10.1 Relationships

	Class	Description	Notes
⇑	TotallyOrdered §1.7		
⇓	StandardizedIdentifier §1.11		

⇑:Inherits ⇓:Inherited by

1.10.2 Operations

String identifier() identifier

The identifier. Return the unique identifier string that this object has.

Boolean equal(Comparable arg) equal

arg: Comparable The object to compare against.

Equality relationship. Two Identifiables are equal if the identifiers that they have are equal. Interfaces and classes that implement Identifiable usually have further equality conditions.

Boolean lessThanOrEqualTo(Comparable arg) lessThanOrEqualTo

arg: Comparable The object to compare against.

Raises: IncomparableException

The less than or equal to relationship. Return true if the result of identifier() for this object is less than or equal to the result of identifier() for arg. Return false otherwise.

Integer hash() hash

Hash value for an object. Return the standard hash for the string returned by identifier().

1.11 StandardizedIdentifier

An identifier based on some international or national standard. Examples of standardized identifiers include the ISO 639 language codes and the ISO 3166 country codes.[1, 2]

For the most part, identifiers based on standards need to obey the general requirements of identifiers generally: unique within the object family and having an ordering based on the identifier string. Standardized identifiers allow more information about the identification scheme.

1.11.1 Relationships

	Class	Description	Notes
⇑	Identifiable §1.10		
↓	StandardizedIdentifierModel §2.3		

⇑:Inherits ↓:Realized by

1.11.2 Operations

String standard() standard

The relevant standard. Return the name of the standard that this identifier comes from. The name is returned in the format that the standards body itself adopts. (eg. ISO-639-1).

As a special case, return nil to indicate that the identifier comes from no recognized standard. If standard() returns nil, then body() should also return nil.

String body() body

The standards body. Return the name or common abbreviation of the standards body that defines the standard. Examples include ISO, ANSI, ASI, etc.

As a special case, return nil to indicate that the identifier comes from no recognized standard. If body() returns nil, then standard() should also return nil.

1.12 Responsible

The Responsible interface denotes an abstract assignment of responsibility. A specific piece of user reference data will normally be used here. Responsibles are generally only compared for equality or printed so that the party responsible for some change can be identified.

Responsibles may refer to groups of parties, for example the back office group.

1.12.1 Relationships

	Class	Description	Notes
⇑	Comparable §1.5		

⇑:Inherits

1.12.2 Operations

Domain domain() domain

Owning domain. Returns the domain that this responsible party is associated with.

String toString() toString

This method returns a string based, user readable representation of the "Responsible" object. This string will include all information necessary to identify this object to a reader.

1.13 InputStream

A stream that can be interrogated to produce a sequence of characters. InputStreams are generally used for parsing purposes. Parsing failures produce a ParseException exception.

1.14 Object

Object represents a top level interface from which all other interfaces inherit. This is provided so that we can have a type that encompasses all objects. Inheritance to this interface is not shown from all other interfaces because we do not wish to close off the inheritance hierarchy, and also because the existence and implementation of a top level object may differ between target languages.

1.15 OutputStream

A stream which can have a textual description written to it.

The print(OutputStream) method is implemented on anything that can produce a textual description. The printString() method, implemented on anything that implements the print method, returns a string representation of the output.

In general, the presence of the print and printString methods can be assumed in all classes. The presence of these method signatures indicates that some additional definition has been provided.

1.16 Reportable

The Reportable interface provides an interface to error and validation reporting machinery.

There are three severity classes of Reportable: Nulls, Errors and Warnings. Null errors represent no problems. Errors represent a problem that will interfere with the operation of the system and, therefore, cannot be manually overridden.

Warnings represent something that probably represents an error, but which the system can handle and can, therefore, be manually overridden. Severity classes are ordered, with $Null < Warning < Error$.

Reportable objects are composable: two Reportables can be combined into a single object that represent a combination of the two errors. The severity class of the composition is the maximum severity class of its components.

1.16.1 Relationships

	Class	Description	Notes
$\downarrow$	ReportableModel		
$\leftrightarrow$	ReportableModelComposite	components 0..n	$\Diamond$

$\downarrow$:Realized by $\leftrightarrow$:Association $\rightarrow$:Navigable $\Diamond$:Aggregate $\blacklozenge$:Composite

1.16.2 Operations

Boolean isError() isError

This Reportable is an error. Returns true if this object has an Error severity class.

Boolean isWarning() isWarning

This Reportable is a warning. Returns true if this object has a Warning severity class.

Boolean isNull() isNull

This Reportable has no problems. Returns true if this object has a Null severity class.

$$\neg isError \wedge \neg isWarning \Leftrightarrow isNull$$

Reportable errors() errors

Errors only. Returns a Reportable containing only the errors contained in this object. If there are no errors, then a Reportable with a severity class of Null is returned.

Reportable warnings() warnings

Warnings only. Returns a Reportable containing only the warnings contained in this object. If there are no errors, then a Reportable with a severity class of Null is returned.

Reportable composedWith(Reportable arg) composedWith
arg: Reportable

Compose two Reportables. Compose this Reportable object with another and produce a new Reportable that is the combination of both.

print(OutputStream stream) print
stream: OutputStream The stream to print onto.

Print the error. Provide a string description of the error(s) or warning(s).

Reportables with Null severity should print nothing.

Composed Reportables should lay the composed errors and warnings out in a suitably readable format.

1.17 TextStream

A stream which can have a rich text description written to it. The prettyPrint(TextStream) method can be implemented on anything that can produce such a description. By default, the print(OutputStream) method is used, without additional font and layout information.

In general, the presence of the print and prettyPrint methods can be assumed in all classes. The presence of these method signatures indicates that some additional definition has been provided.

1.18 Validatable

A Validatable object is one that can be interrogated for internal consistency.

Validation would be performed as a final check on entry or modification of data. It would also be performed when data is received from an external system.

Validation would test things like the presence of essential information; that specific pieces of information have valid values; and that different pieces of information are consistent. For example, if an integer field can only take values of +1 or -1, then validation would test this. It could also test that a deal has a legitimate counterparty. It can check security and business rules.

In the future, Validatable objects may need to be extended to provide a context mechanism. Currently, however, Validatable objects are expected to be able to confirm their validity by examining their internal state.

A validatable object can hold an association to another validatable object. If the first object changes then the validation on the associated object should be validated as well; care must be taken when doing these cross-association validations as the validation of the associated object may cause a validation loop if it calls back to the original object's validation method.

1.18.1 Relationships

	Class	Description	Notes
⇓	Enum §4.1		
↓	StandardizedIdentifierModel §2.3		
↓	Enum §4.1		

⇓:Inherited by ↓:Realized by

1.18.2 Operations

Reportable validate() validate

Validate the object. Return a Reportable giving details of any internal errors or warnings within the object. Classes that implement Validatable will have an explicit set of validation criteria.

1.19 ValueSemantics

Value semantics indicates that instances of the object in question act as values. Operations on values create new instances of values, rather than directly modify the values themselves. Examples of objects showing value semantics are: Booleans, Numbers, Strings (normally), Dates, Times, etc.

Attributes may only be objects that have value semantics.

1.19.1 Relationships

	Class	Description	Notes
⇓	Datestamp §1.8		
↓	StandardizedIdentifierModel §2.3		
↓	Enum §4.1		

⇓:Inherited by ↓:Realized by

2 Classes

2.1 ClassifierModel

This class is a concrete realization of the Classifier interface.

2.1.1 Relationships

	Class	Description	Notes
↑	Classifier §1.2		

↑:Realizes

2.1.2 Attributes

dictionary: Dictionary Holds a dictionary that contains all the key-value pairs that represent the classification of something.

2.2 DualSpecificationCollectionModel

This class is a concrete realization of the ClassifierValue interface.

2.2.1 Relationships

	Class	Description	Notes
↑	DualSpecificationCollection §1.3		
↔	ClassRepresentation §1.1	classes 0..n	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

2.2.2 Attributes

instances: Collection<Object>

2.3 StandardizedIdentifierModel

An implementation of the StandardizedIdentifier interface. Instances of this class obey value semantics making them suitable for use as attributes.

No particular restrictions are placed upon this model, although the validation protocol can be extended to provide verification that the identifier is permitted by the standard.

2.3.1 Relationships

	Class	Description	Notes
↑	StandardizedIdentifier §1.11		
↑	Validatable §1.18		
↑	ValueSemantics §1.19		

↑:Realizes

2.3.2 Attributes

body: String The standards body.

standard: String The relevant standard.

identifier: String The relevant identifier.

2.3.3 Operations

Reportable validate() validate

Validate the identifier.

This operation has been included as a hook for later expansion. It is likely that more extensive validation of standards-based identifiers is likely in future, or can be part of an implementation of this class.

- The Identifer cannot be null.
- The Body cannot be null.
- The Standard cannot be null.

Currently, a Null Reportable §1.16 object can be returned.

2.4 StringClassRepresentationModel

This class is a concrete realization of the ClassRepresentation interface.

2.4.1 Relationships

	Class	Description	Notes
↑	ClassRepresentation §1.1		

↑:Realizes

2.4.2 Attributes

className: String

2.5 UnorderedEnum

2.5.1 Relationships

	Class	Description	Notes
⇑	Enum §4.1		

⇑:Inherits

3 Exceptions

3.1 EnumOutOfRangeException

An exception raised when an enumeration is asked to provide an element not in the acceptable range of elements.

3.1.1 Operations

Class enum() enum

The class of the enumeration that detected a range error.

3.2 IncomparableException

This exception is raised whenever an attempt is made to compare two objects that are strictly incomparable, For example, 1 and "A String" are not strictly comparable.

3.2.1 Operations

Comparable comparer() comparer

The comparer.

The object that was doing the comparison. That is, the receiver of whatever message raised the exception.

Comparable comparee() comparee

The comparison object.

The object that the receiver was comparing against when this exception was raised.

3.3 NotFoundException

An exception raised whenever a collection cannot find and object or key.

3.3.1 Operations

Object key() key

Absent key.

The object or key that was being searched for.

3.4 ParseException

A ParseException is raised if an input stream being parsed contains an unexpected sequence.

3.4.1 Operations

InputStream stream() stream

The stream which contains the unexpected item.

4 Enumerations

4.1 Enum

The top-level class for all enumeration types. Enumerations are assumed to implement ValueSemantics, making them suitable candidates for use as attributes.

Subclasses implement enumerations by providing static methods that supply the instances of the enumeration. For example, a color-space enumeration might have static operations called red(), blue() and green() that return the individual instances of the enumeration. These operations should always return the same instance when invoked.

4.1.1 Relationships

	Class	Description	Notes
⇑	Validatable §1.18		
↑	ValueSemantics §1.19		
↑	Validatable §1.18		
↑	Comparable §1.5		
⇓	UnorderedEnum §2.5		
⇓	OrderedEnum §4.2		

⇑:Inherits ⇓:Inherited by ↑:Realizes

4.1.2 Attributes

name: String The name of the enumeration element.

4.1.3 Operations

String name() name

The name of the enumeration.

Return a string that uniquely identifies this element of the enumeration (class and name uniquely indentifies the element across all enumerations).

Integer hash() hash

Hash value for an object.

Return the hash of the name of the element.

Boolean equals(Comparable arg) equals

arg: Comparable The object to compare this object against.

The equality relationship.

Enumerations use distingushed instances to recognise each element of the enumeration. Equality, therefore, is usually the same as object identity.

Returns true if two objects are equal, false otherwise. Any Comparable object can be supplied as an argument; implementors are expected to perform type-checking to avoid runtime errors.

«Static Method» Collection<Enum> elements() elements

The elements of the enumeration.

Return a collection of all the elements of the enumeration.

Reportable validate() validate

Validate the Enum
an Enum is invalid if:

- Its name is null

4.2 OrderedEnum

An enumeration that has some order.

The elements of an ordered enumeration are associated with an integer, the *order*, which gives the place of the element in the enumeration. The orders associated with the elements of an enumeration do not need to be sequential, just unique.

4.2.1 Relationships

	Class	Description	Notes
⇑	Enum §4.1		
↑	TotallyOrdered §1.7		

⇑:Inherits ↑:Realizes

4.2.2 Attributes

order: Integer The order of the element in the enumeration.

4.2.3 Operations

Boolean lessThanOrEqualTo(TotallyOrdered arg) lessThanOrEqualTo
arg: TotallyOrdered The object to compare this object against.
Raises: IncomparableException

The less than or equal to relationship.

Raise an IncomparableException if the argument is not of the same enumeration class as this element. Otherwise, return true if $this.order \leq arg.order$ and false otherwise.

OrderedEnum pred(Integer count) pred
count: Integer The number of elements to move back. The default value is 1.
Raises: EnumOutOfRangeException

Predecessor of this element.

If count is 0 then return this element. If count is less than zero then return the result of succ() with the negation of count as an argument.

Otherwise, let p be the predecessor of this element: the element with an order less than the order of this element and closest to the order of this element. If there is no such element, raise an EnumOutOfRangeException.

If count is 1 then return p. If count is greater than 1 then return $p.pred(count - 1)$.

Note that the implementation of this operation can use methods considerably more efficient than those implied in the definition.

OrderedEnum succ(Integer count) succ

count: Integer The number of elements to move forward. The default value is 1.

Raises: EnumOutOfRangeException

Successor of this element.

If count is 0 then return this element. If count is less than zero then return the result of pred() with the negation of count as an argument.

Otherwise, let p be the successor of this element: the element with an order greater than the order of this element and closest to the order of this element. If there is no such element, raise an EnumOutOfRangeException.

If count is 1 then return p. If count is greater than 1 then return $p.succ(count - 1)$.

Note that the implementation of this operation can use methods considerably more efficient than those implied in the definition.

OrderedEnum cyclicPred(Integer count) cyclicPred

count: Integer The number of elements to move back. The default value is 1.

Raises: EnumOutOfRangeException

Cyclic predecessor of this element.

If count is 0 then return this element. If count is less than zero then return the result of cyclicSucc() with the negation of count as an argument.

Otherwise, let p be the predecessor of this element: the element with an order less than the order of this element and closest to the order of this element. If there is no such element, then p is the result of the last() operation for this class.

If count is 1 then return p. If count is greater than 1 then return $p.cyclicPred(count - 1)$.

Note that the implementation of this operation can use methods considerably

more efficient than those implied in the definition.

OrderedEnum cyclicSucc(Integer count) cyclicSucc
count: Integer The number of elements to move forward. The default value is 1.
Raises: EnumOutOfRangeException

Cyclic successor of this element.

If count is 0 then return this element. If count is less than zero then return the result of cyclicPred() with the negation of count as an argument.

Otherwise, let p be the successor of this element: the element with an order greater than the order of this element and closest to the order of this element. If there is no such element, let p the result of the first() operation for this class.

If count is 1 then return p. If count is greater than 1 then return $p.cyclicSucc(count-1)$.

Note that the implementation of this operation can use methods considerably more efficient than those implied in the definition.

«Static Method» Collection<Enum> elements() elements

The elements of the enumeration.

Return a collection sorted into order (smallest first) of all the elements of the enumeration.

«Static Method» OrderedEnum first() first

The smallest element in the enumeration.

Return the element with the least order.

«Static Method» OrderedEnum last() last

The largest element in the enumeration.

Return the element with the greatest order.

5 Associations

Table 1: Utilities— Associations

Association Role	Class	Card.	Notes
classes	ClassRepresentation §1.1	0..n	→

Table 1: . . . continued

Association Role	Class	Card.	Notes
	DualSpecificationCollection-Model §2.2		

→:Navigable ◊:Aggregate ⧫:Composite

5.1 classes

Role: *Navigable* ClassRepresentation, 0..n.
Role: DualSpecificationCollectionModel.

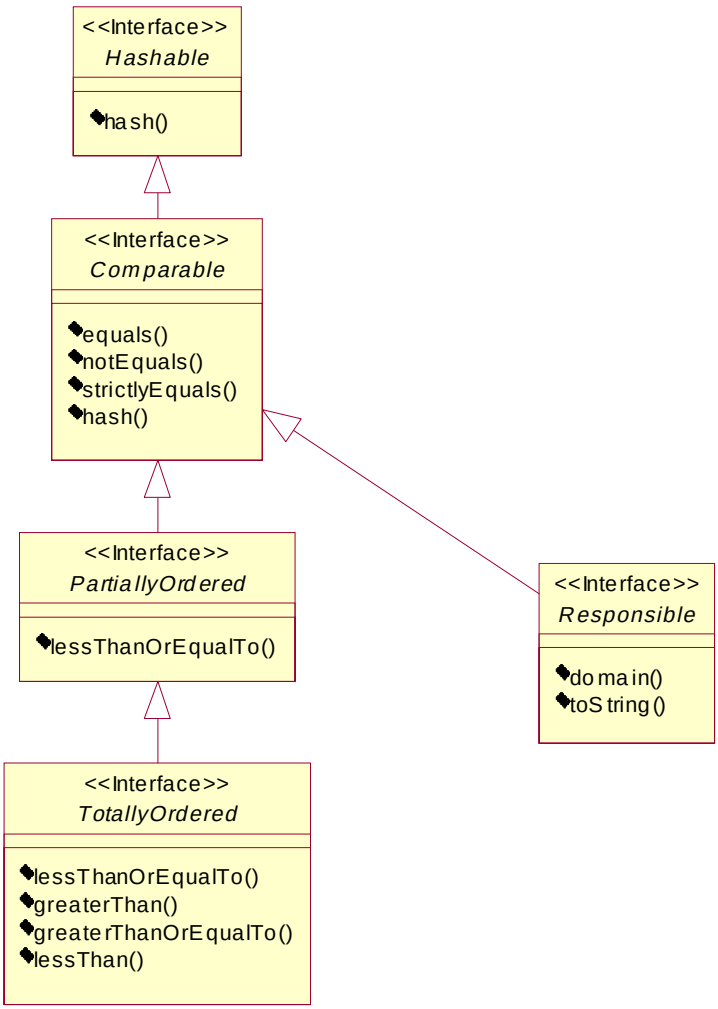


Figure 1: Class Diagram— Comparison

Figure 2: Class Diagram— Validation

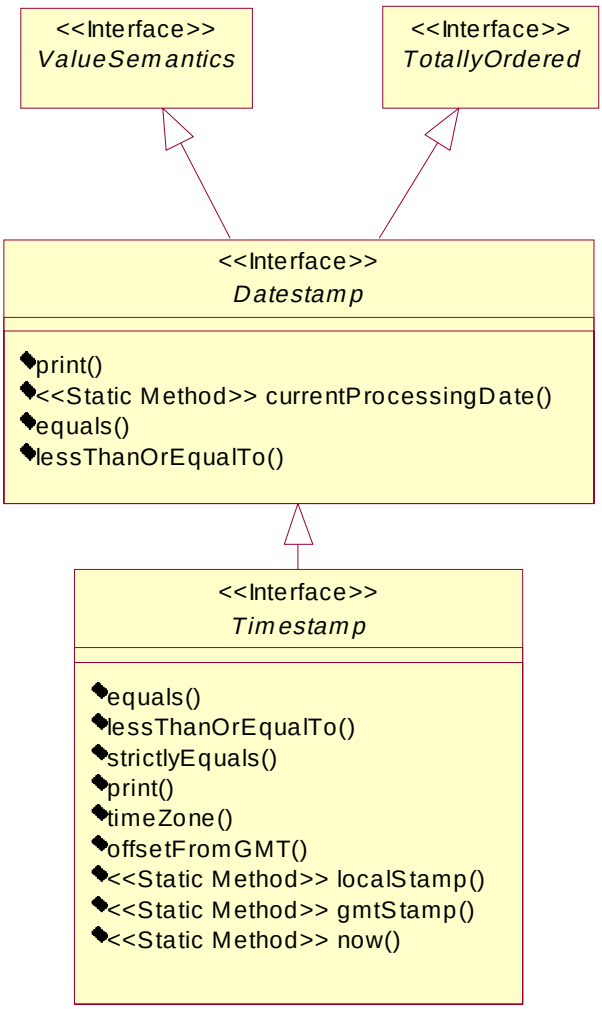


Figure 3: Class Diagram— Stamps

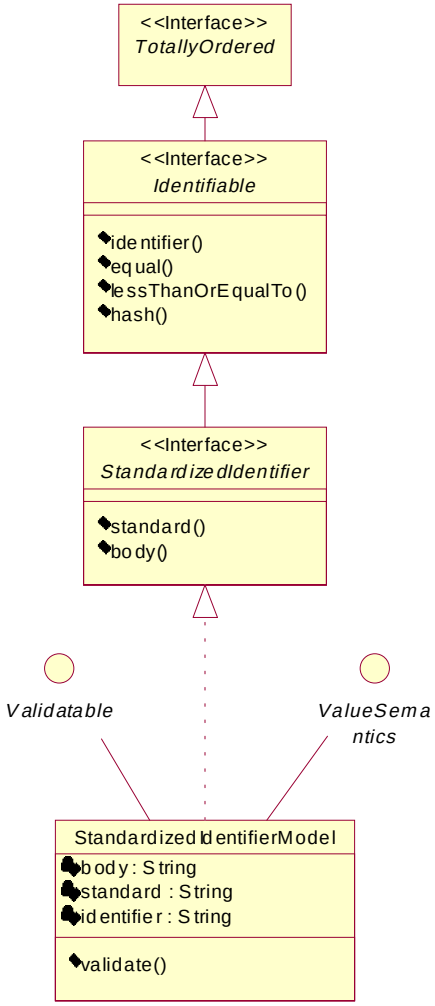


Figure 4: Class Diagram— Identification

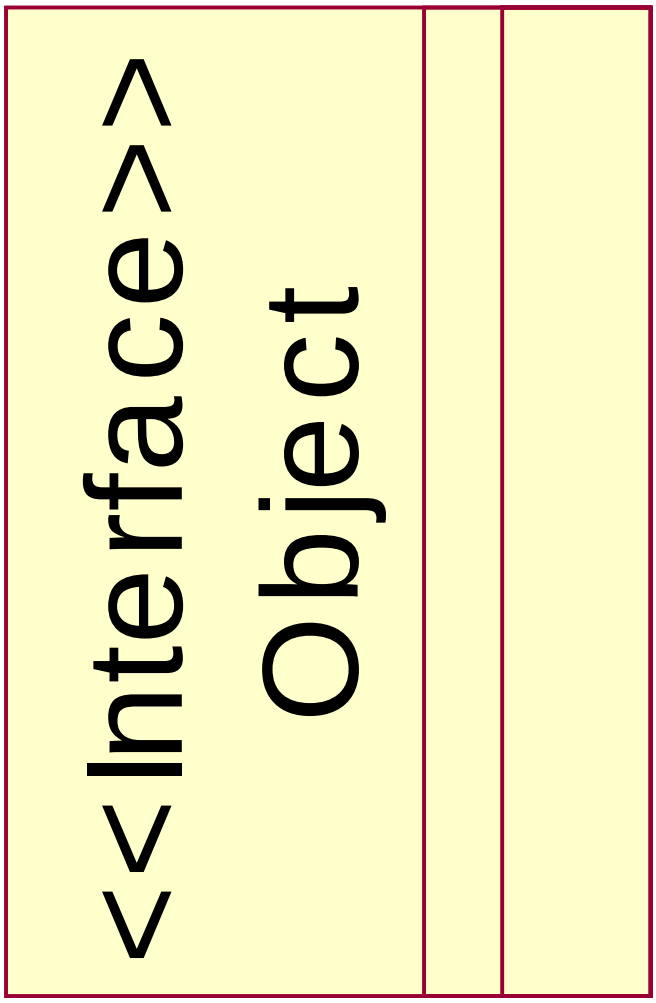


Figure 5: Class Diagram— Object

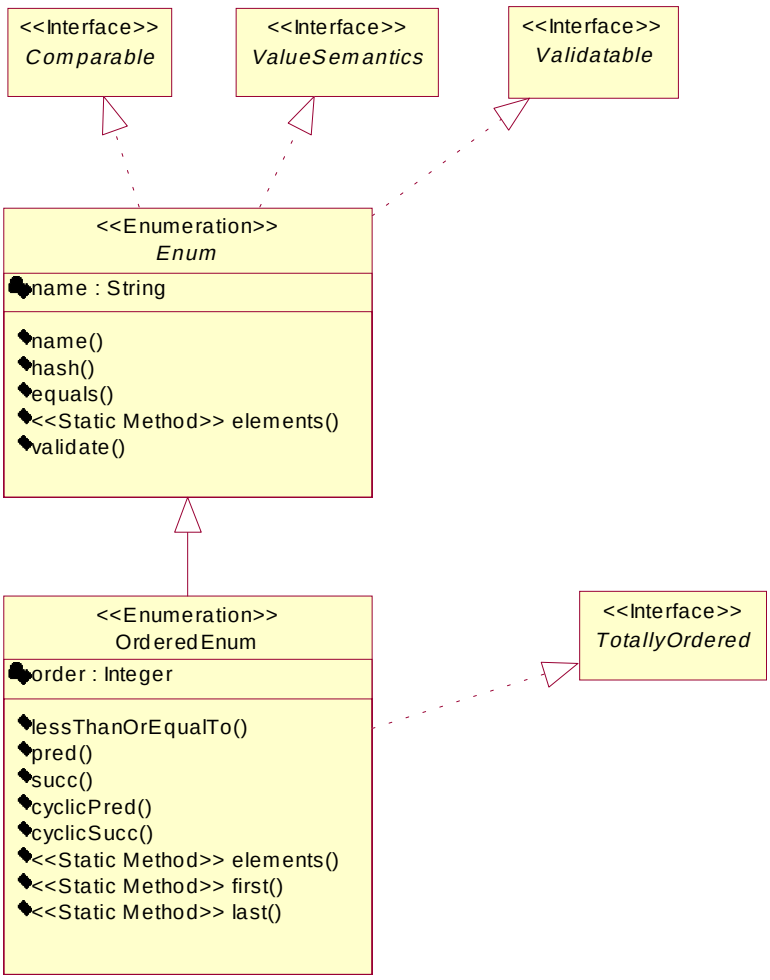


Figure 6: Class Diagram— Enumerations

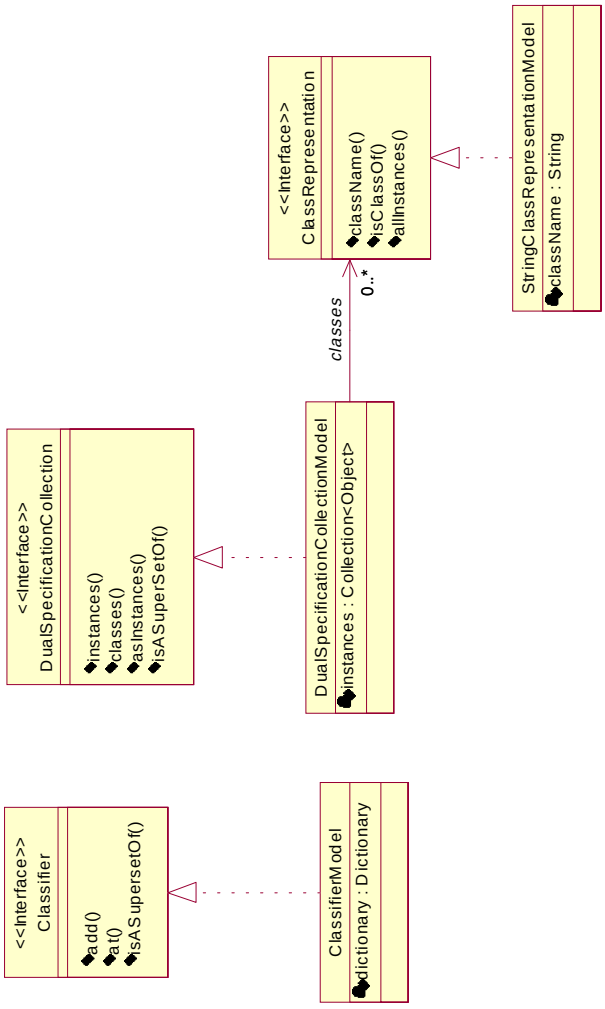


Figure 7: Class Diagram— Classifier

6 Extensions to the Utilities Implementation Package

The Utilities package contains interface definitions for common behaviors in the elements model. In most cases, the interfaces described in Utilities will be supplied as library classes, or inbuilt behavior, by the implementation language.

Some behaviors will be missing, however. The Utilities implementation package contains concrete implementations of some of the interfaces described in the Utilities package. These implementations can be used as needed.

6.1 ReportableModel

A concrete implementation of the Reportable interface. Subclasses provide concrete behavior.

6.1.1 Relationships

	Class	Description	Notes
↑	Reportable §1.16		
⇓	ReportableModelNull		
⇓	ReportableModelPrimitive		
⇓	ReportableModelComposite		

⇓:Inherited by ↑:Realizes

6.1.2 Operations

Reportable composedWith(Reportable arg) composedWith
arg: Reportable

Compose two reportables.

Return a ReportableModelComposite containing this reportable and arg as components.

6.2 ReportableModelComposite

A reportable error or warning containing more than one error or warning.

The multiple errors or warnings are collected into an aggregation. Note that an instance of this class must have two or more components; instances with fewer components are reduced to an instance of ReportableModelPrimitive or of ReportableModelNull.

6.2.1 Relationships

	Class	Description	Notes
⇑	ReportableModel		
↔	Reportable §1.16	components 2..n	→

⇑:Inherits ↔:Association →:Navigable ◊:Aggregate ♦:Composite

6.2.2 Operations

Boolean isError() isError

This reportable is an error?

Return true if there exists a component which returns true to isError, return false otherwise.

Boolean isWarning() isWarning

This reportable is a warning?

Return true if no component returns true to isError, return false otherwise.

Boolean isNull() isNull

This reportable is null?

Return false.

Reportable errors() errors

Return errors only.

Collect the errors returned from the components into a single collection. If the resulting collection is empty, return an instance of ReportableModelNull. If the resulting collection contains one element, return that element. Otherwise, return a ReportableModelComposite with the collected elements as components.

Reportable warnings() warnings

Return warnings only.

Collect the warnings returned from the components into a single collection. If the resulting collection is empty, return an instance of ReportableModelNull. If the resulting collection contains one element, return that element. Otherwise, return a ReportableModelComposite with the collected elements as components.

print(OutputStream stream) print

stream: OutputStream The stream to print onto.

Print the error.

For each error in the composite, print the reportable onto the stream, interspersing a new line between each error. For each warning in the composite, print the reportable onto the stream, interspersing a new line between each warning. Errors always come before warnings.

6.3 ReportableModelNull

An instance of this class is used if no errors can be found. This class contains no state and can be used as a singleton.

6.3.1 Relationships

	Class	Description	Notes
⇑	ReportableModel		

⇑:Inherits

6.3.2 Operations

Boolean isError() isError

This reportable is an error?

Return false.

Boolean isWarning() isWarning

This reportable is a warning?

Return false.

Boolean isNull() isNull

This reportable is null?

Return true.

Reportable errors() errors

Return errors only.

Return this object.

Reportable warnings() warnings

Return warnings only.

Return this object.

print(OutputStream stream) print

stream: OutputStream The stream to print onto.

Print the error.

Do nothing.

6.4 ReportableModelPrimitive

A reportable error containing a single error or warning.

6.4.1 Relationships

	Class	Description	Notes
⇑	ReportableModel		

⇑:Inherits

6.4.2 Attributes

severity: Enumeration = error The severity of the reportable, one of { warning, error }

description: String A description of the error or warning.

6.4.3 Operations

Boolean isError() isError

This reportable is an error?

Return true if severity is set to error, otherwise return false.

Boolean isWarning() isWarning

This reportable is a warning?

Return true if severity is set to warning, otherwise return false.

Boolean isNull() isNull

This reportable is null?

Return false.

Reportable errors() errors

Return errors only.

If severity is set to error, return this object. Otherwise return an instance of ReportableModelNull.

Reportable warnings() warnings

Return warnings only.

If severity is set to warning, return this object. Otherwise return an instance of ReportableModelNull.

print(OutputStream stream) print

stream: OutputStream The stream to print onto.

Print the error.

Print the description on the stream.

6.5 Associations

Table 2: Utilities Implementation— Associations

Association Role	Class	Card.	Notes
components			
component	Reportable §1.16	2..n	→
composite	ReportableModelComposite	0..n	◊

→:Navigable ◊:Aggregate ♦:Composite

6.5.1 components

Role: component *Navigable* Reportable, 2..n.

Role: composite *Aggregate* ReportableModelComposite, 0..n.

The component reportables that make up a composite reportable.

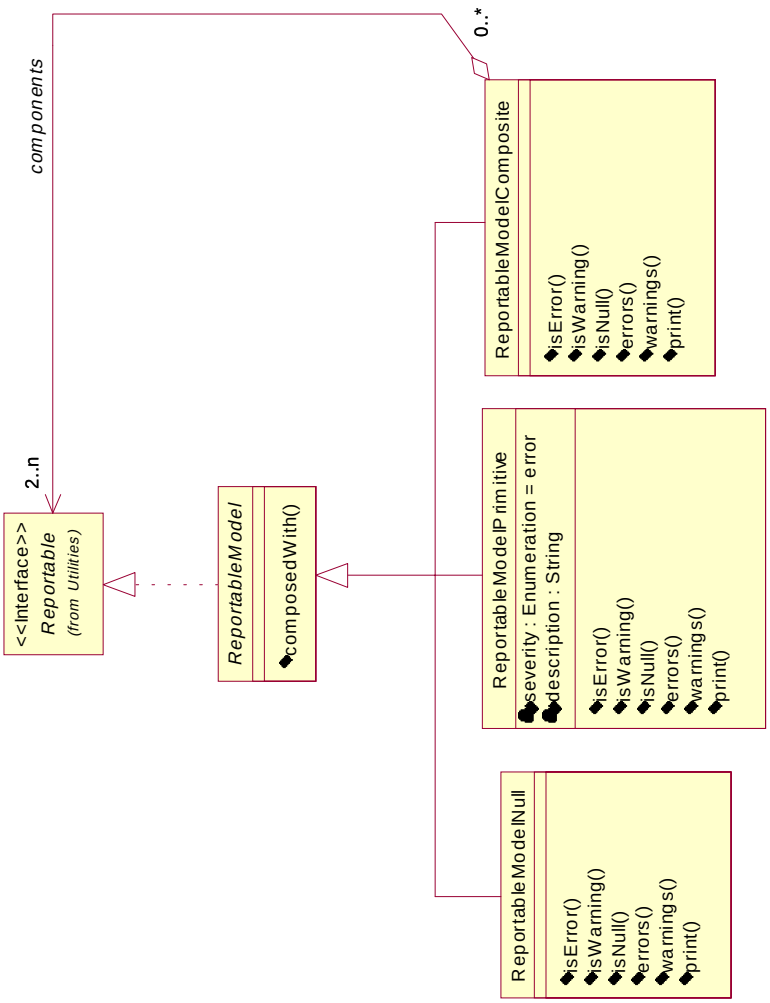


Figure 8: Class Diagram— Utilities Implementation

References

[1] International Organization for Standardization (ISO). *Code for the Representation of Names of Languages*, number ISO 639, 1988. http://www.iso.ch/cate/d4766.html.

[2] International Organization for Standardization (ISO). *Codes for the Representation of Names of Countries and Their Subdivisions*, number ISO 3166, 1997–9. http://www.din.de/gremien/nas/nabd/iso3166ma/index.html.