TECHNICAL REPORT

ISO/IEC TR 24717

First edition
2009-08-01

Information technology — Programming languages, their environments and system software interfaces — Collection classes for programming language COBOL

Technologies de l'information — Langages de programmation, leurs environnements et interfaces du logiciel système — Classes de collection pour le langage de programmation COBOL

Reference number
ISO/IEC TR 24717:2009(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Tables

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 24717, which is a Technical Report of type 2, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*, in collaboration with INCITS Technical Committee J4, Programming language COBOL.

Introduction

This Technical Report specifies an object-oriented class library for managing collections of object references — a collection class library.

This Technical Report extends the COBOL specification defined in ISO/IEC 1989:2002, *Information technology — Programming languages — COBOL* by providing classes to manage collections.

Annex A forms a normative part of this Technical Report. Annex B, Annex C, Annex D, and the Bibliography are for information only.

Information technology — Programming languages, their environments and system software interfaces — Collection classes for programming language COBOL

1 Scope

This Technical Report specifies the interfaces and behavior of a common class library for managing sets of object references in COBOL. The purpose of this Technical Report is to promote a high degree of portability in implementations of the class library, even though some elements are subject to trial before completion of a final design suitable for standardization.

This specification builds on the syntax and semantics defined in ISO/IEC 1989:2002.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 1989:2002, *Information technology — Programming languages — COBOL*

3 Conformance

This Technical Report is based on ISO/IEC 1989:2002. Conformance to this Technical Report does not require a full implementation of ISO/IEC 1989:2002. The interaction of the features of this Technical Report with features that are not provided by an implementation of ISO/IEC 1989:2002 is processor dependent.

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
collection
set of object references managed by an instance of a collection class

4.2
iterator
object that allows sequencing through all of the object references managed by an instance of a collection class

5 Description techniques

Description techniques and language fundamentals are the same as those described in ISO/IEC 1989:2002. Additionally the class diagrams in Appendix D are presented using Unified Modeling Language (UML).

6 Changes to ISO/IEC 1989:2002

These changes refer to clause and rule numbers in ISO/IEC 1989:2002.

1. 16.1, Base class, insert into BaseFactoryInterface interface definition after 'Procedure Division.'

"

```
    Method-id. ClassName.
    Data division.
    Linkage section.
    01  outName pic n(31).
    Procedure division returning outName.
    End method ClassName.
*>
    Method-id. ExternalClassName.
    Data division.
    Linkage section.
    01  outName.
        02  outNameCategory pic x.
            88  ExternalClassNameDisplay value "X".
            88  ExternalClassNameNational value "N".
        02  outNameNational pic n(1024).
        02  outNameDisplay redefines outNameNational
            pic x(1024).
    Procedure division returning outName.
    End method ExternalClassName.
*>
```

"

2. Add new sections prior to 16.1.1, renumbering 16.1.1 to 16.1.2, etc.

"

16.1.1 ClassName

The ClassName method is a factory method that returns the internal name of the class associated with the factory for which it is invoked.

16.1.1.1 General Rules

1) The ClassName method returns the internal class name of the factory in outName.

NOTE 1 The class name is returned in national characters and might not be suitable for use where an exact case-sensitive class name is required.

NOTE 2 The following code can be used to determine the name of the class of the instance object anObject:

```
Invoke anObject "FactoryObject" returning aFactoryObject
Invoke aFactoryObject "ClassName" returning aClassName
```

16.1.2 ExternalClassName

The ExternalClassName method is a factory method that returns the external name of the class associated with the factory for which it is invoked.

16.1.2.1 General Rules

1) If the external class name specified in the AS clause of the CLASS-ID paragraph is a national literal, the ExternalClassName method sets ExternalClassNameNational to true, and returns the external class name of the factory in outNameNational. Otherwise the external class name of the factory is returned in outNameDisplay and ExternalClassNameDisplay is set to true.

"

3. Add new section.

"16.2 ExceptionInformation class

The ExceptionInformation class provides information on exceptions that may occur in any method that inherits from the BASE class.

```
Interface-id. ExceptionInterface
  Inherits BaseInterface.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface.
*>
Procedure division.
    Method-id. Get Property ExceptionClassName.
    Data division.
    Linkage section.
    01  outName pic n any length.
    Procedure division returning outName.
    End method ExceptionClassName.
*>
    Method-id. Set Property ExceptionClassName.
    Data division.
    Linkage section.
    01  outName pic n any length.
    Procedure division using outName.
    End method ExceptionClassName.
*>
    Method-id. Get Property ExceptionCode.
    Data division.
    Linkage section.
    01  outCode pic n(31).
    Procedure division returning outCode.
    End method ExceptionCode.
*>
    Method-id. Set Property ExceptionCode.
    Data division.
    Linkage section.
    01  outCode pic n(31).
    Procedure division using outCode.
    End method ExceptionCode.
*>
    Method-id. Get Property ExceptionMethodName.
    Data division.
    Linkage section.
    01  outName pic n any length.
    Procedure division returning outName.
    End method ExceptionMethodName.
*>
    Method-id. Set Property ExceptionMethodName.
    Data division.
    Linkage section.
    01  outName pic any length.
    Procedure division using outName.
    End method ExceptionMethodName.
*>
    Method-id. Get Property ExceptionMessage.
    Data division.
    Linkage section.
    01  outMessage pic n any length.
    Procedure division returning outMessage.
    End method ExceptionMessage.
*>
```

```
    Method-id. Set Property ExceptionMessage.
    Data division.
    Linkage section.
    01  outMessage pic n any length.
    Procedure division using outMessage.
    End method ExceptionMessage.
*>
    Method-id. Get Property ExceptionSourceObject.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject.
    End method ExceptionSourceObject.
*>
    Method-id. Set Property ExceptionSourceObject.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division using outObject.
    End method ExceptionSourceObject.
*>
End Interface ExceptionInterface.
```

Properties of the ExceptionInformation class may be used to set or get:

- the method where the exception object was raised,
- the class that contained that method,
- the object on which that method was invoked,
- an exception code that is indicative of the exception that occurred,
- a description of the exception that occurred.

16.2.1 ExceptionClassName property

The ExceptionClassName property is used to set or get the name of the class in which the exception was raised. The maximum size of the method name returned is 1024.

16.2.2 ExceptionCode property

The ExceptionCode property is used to set or get a national character string that indicates the type of exception raised. All exception codes defined by this standard begin with the characters “EO-“.

> NOTE This method is overridden by exception classes that are specific to and associated with the classes that raise the exceptions. Valid exception codes are defined in each class to correspond to those exceptions raised by that class. Exception codes defined for a subclass are in addition to those defined by the class from which the subclass inherits.

16.2.3 ExceptionMethodName property

The ExceptionMethodName property is used to set or get the name of the method in which the exception was raised.

16.2.4 ExceptionMessage property

The ExceptionMessage property is used to set or get a message describing why the exception was raised. The contents of the ExceptionMessage are implementor-defined.

16.2.5 ExceptionSourceObject property

The ExceptionSourceObject property sets or gets an instance object reference or a factory object reference. The object reference returned references the object upon which the method that raised the exception object was invoked.

"

7 COBOL Collection Classes

Programmers working with objects quickly find it necessary to manage references to multiple instances of related objects. When managed, these references make up a collection. Collections in COBOL are managed by a collection class as specified in this Technical Report.

Object references are inserted into an instance of the collection class according to the attributes specified when the instance of the collection is created. If a collection is not ordered and not keyed, the object references are inserted sequentially within the collection in the order that they are added. If a collection is ordered, the object references are added using a method of the ordered collection class to position that object reference within the collection relative to the other object references of the collection. If a collection is keyed, each object reference in the collection is associated with a key item when it is added to the collection. Keys are shortcuts to allow retrieval of object references from a collection using a convenient name rather than an object reference. Keys associated with object references do not specify the order of those object references within a collection. If a collection is sorted, a method within each member of the collection is invoked as the object reference is added to the collection. This method returns a national data item that is used to control the sequence in which object references are returned from the collection.

Object references may be retrieved from collections: sequentially; directly using their ordinal positions or, for keyed collections, using their keys; or indirectly using an instance of the Iterator class. The Iterator class provides for retrieval of object references from a collection either in the order they exist within a collection, or in a sequence defined at the creation of the iterator. The iterator may also be used to delete an object from the underlying collection. Collections that contain no object references are empty collections. Attempting to retrieve an object reference from an empty collection will raise an exception object.

Object references that are inserted into or retrieved from instances of the collection class may be restricted to object references that conform to a particular class by using the parameterized forms of the collection class interfaces. Other than these restrictions on the class of the object reference, the methods of these parameterized interfaces function equivalently to the methods of non-parameterized interfaces. These parameterized interfaces, however, do not inherit from the common collection-class interface, but redefine the methods of the collection class within each subclass. The functionalities of these methods are equivalent to the functionalities of the corresponding methods defined in the collection class interface.

When exceptions occur in the invocation of the methods of the collection class, an exception object is created and raised. The exception object contains information that describes the exception and the method in which the exception occurred.

7.1 Collection class

The Collection class is the base class for all collections and includes all the methods necessary to manage the membership of a collection.

The instance methods of the Collection class perform the following functions:

- add object references to a collection
- delete object references from a collection
- compare the membership of two collection instances
- copy an instance of a collection
- count object references in a collection
- return object references from a collection
- create an iterator for a collection

The following is the specification of the formal interfaces supported by the Collection class.

7.1.1 Collection instance interface

The methods associated with the instance objects of the Collection class provide the facilities necessary to establish and maintain a collection.

An instance of the Collection class maintains the ordinal position for each object reference that is added to that collection class. The ordinal position of the first object reference is 1. Object references may be retrieved in sequence or directly by their ordinal position. The collection maintains the ordinal position of the last object reference retrieved from or added to the collection. This object reference is the current reference for this collection.

Collection instance interface

```
Interface-id. CollectionInterface
  Inherits BaseInterface.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface CollectionExInterface
    Interface IteratorInterface.
*>
Procedure division.
    Method-id. AddObject.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    Procedure division using by value inObject
      raising CollectionExInterface.
    End method AddObject.
*>
    Method-id. CompareCollection.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method CompareCollection.
*>
    Method-id. CopyCollection.
    Data division.
    Linkage section.
    01  outObject usage object reference active-class.
    Procedure division returning outObject.
    End method CopyCollection.
*>
    Method-id. CountObjects.
    Data division.
    Linkage section.
    01  outBinary usage binary-long.
    Procedure division returning outBinary.
    End method CountObjects.
*>
    Method-id. CreateIterator.
    Data division.
    Linkage section.
    01  inSequence pic N any length.
    01  outIterator usage object reference IteratorInterface.
    Procedure division using optional inSequence returning outIterator
      raising CollectionExInterface.
    End method CreateIterator.
*>
    Method-id. DeleteAll.
    Procedure division.
    End method DeleteAll.
*>
```

```
    Method-id. DeleteCurrent.
    Procedure division
      raising CollectionExInterface.
    End method DeleteCurrent.
*>
    Method-id. DeleteObject.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    Procedure division using by value inObject
      raising CollectionExInterface.
    End method DeleteObject.
*>
    Method-id. Exists.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method Exists.
*>
    Method-id. Ordinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal.
    End method Ordinal.
*>
    Method-id. ReturnCurrent.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnCurrent.
*>
    Method-id. ReturnFirst.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnFirst.
*>
    Method-id. ReturnLast.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnLast.
*>
    Method-id. ReturnNext.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnNext.
*>
    Method-id. ReturnObject.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outObject usage object reference.
    Procedure division using by value inOrdinal returning outObject
      raising CollectionExInterface.
    End method ReturnObject.
```

```
*>
    Method-id. ReturnPrevious.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnPrevious.
*>
End Interface CollectionInterface.
```

Parameterized collection instance interface

```
Interface-id. ParamCollectionInterface
  Inherits BaseInterface
  Using ParamClass.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface CollectionExInterface
    Interface IteratorInterface
    Class     ParamClass.
*>
Procedure division.
    Method-id. AddObject.
    Data division.
    Linkage section.
    01  inObject usage object reference ParamClass.
    Procedure division using by value inObject
      raising CollectionExInterface.
    End method AddObject.
*>
    Method-id. CompareCollection.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method CompareCollection.
*>
    Method-id. CopyCollection.
    Data division.
    Linkage section.
    01  outObject usage object reference active-class.
    Procedure division returning outObject.
    End method CopyCollection.
*>
    Method-id. CountObjects.
    Data division.
    Linkage section.
    01  outBinary usage binary-long.
    Procedure division returning outBinary.
    End method CountObjects.
*>
    Method-id. CreateIterator.
    Data division.
    Linkage section.
    01  inSequence pic N any length.
    01  outIterator usage object reference IteratorInterface.
    Procedure division using optional inSequence returning outIterator
      raising CollectionExInterface.
    End method CreateIterator.
*>
    Method-id. DeleteAll.
    Procedure division.
    End method DeleteAll.
*>
```

```
    Method-id. DeleteCurrent.
    Procedure division
      raising CollectionExInterface.
    End method DeleteCurrent.
*>
    Method-id. DeleteObject.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising CollectionExInterface.
    End method DeleteObject.
*>
    Method-id. Exists.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method Exists.
*>
    Method-id. Ordinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal.
    End method Ordinal.
*>
    Method-id. ReturnCurrent.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnCurrent.
*>
    Method-id. ReturnFirst.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnFirst.
*>
    Method-id. ReturnLast.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnLast.
*>
    Method-id. ReturnNext.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnNext.
*>
    Method-id. ReturnObject.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outObject usage object reference paramClass.
    Procedure division using by value inOrdinal returning outObject
      raising CollectionExInterface.
    End method ReturnObject.
```

```
*>
    Method-id. ReturnPrevious.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising CollectionExInterface.
    End method ReturnPrevious.
*>
End Interface ParamCollectionInterface.
```

7.1.1.1 AddObject method

The AddObject method adds the specified object reference to the collection and assigns the object reference an ordinal position equal to 1 greater than that of the highest ordinal position associated with the collection. If the object reference is NULL, a CollectionException object whose ExceptionCode method returns EO-NULL is raised and the object reference is not added to the collection. If resources do not exist to add the object reference, a CollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and the object reference is not added to the collection.

NOTE Adding an object reference invalidates any iterators associated with this collection.

7.1.1.2 CompareCollection method

The CompareCollection method compares the membership of the collection to the membership of another collection and returns either 1 (true) or 0 (false) indicating whether those collections reference the same objects. Key values need not match. Collections that contain multiple references to an object are equal only if the same numbers of multiple references to that object exist in both collections.

NOTE The order in which object references were added to the two collections does not matter.

7.1.1.3 CopyCollection method

The CopyCollection method creates a new collection of the same class with identical content and returns an object reference to the new collection. Related iterator are not copied. The current object of the new collection is the first object of that collection.

NOTE The sequence of the new collection is the same as that of the old collection. If this is a sorted collection, the sequencing method of the new collection is the same as that of the old.

7.1.1.4 CountObjects method

The CountObject method returns the number of object references managed by this instance of the Collection class. Duplicate references to an object are counted as distinct object references.

7.1.1.5 CreateIterator method

The CreateIterator method creates and returns an iterator instance for this collection as specified in 7.5, Iterator class. An optional sequencing method parameter may be specified. If this parameter does not refer to a valid method as defined in 7.1.2, Sequencing method, a CollectionException object whose ExceptionCode method returns EO-INVALID-SEQUENCING-METHOD is raised and CreateIterator returns the NULL object reference. If the sequencing method parameter is omitted, object references are returned by the iterator in the sequence that they were added to the collection. The current reference of the iterator is set as described in 7.5.2.5, ReturnFirst.

NOTE 1 There may be more than one iterator associated with a collection. These iterators may differ in sequence.

NOTE 2 If the collection is an ordered collection and the sequencing method is omitted, the sequence in which object references are returned by the iterator can differ from the sequence in which object references are returned by the collection.

NOTE 3 An iterator may also be created by the NewIterator factory method of the Iterator class. A parameterized iterator may only be created by the NewIterator factory class.

If resources do not exist to create the iterator, a CollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and CreateIterator returns the NULL object reference.

7.1.1.6 DeleteAll method

The DeleteAll method removes all object references from the collection.

NOTE Deleting all object references invalidates all iterators associated with this collection. After deleting all objects, there is no current reference.

7.1.1.7 DeleteCurrent method

The DeleteCurrent method removes the current reference from the collection. If there is no current reference, a CollectionException object whose ExceptionCode method returns EO-NO-CURRENT-OBJECT-REFERENCE is raised.

The ordinal positions of all object references with ordinal positions greater than the current reference are decremented by 1. The object reference with the ordinal position that becomes equal to the ordinal position of the object reference that has been deleted becomes the current reference. If the ordinal position of the object reference deleted is the highest ordinal position for this collection, there is no current reference.

NOTE Deleting an object reference invalidates all iterators associated with this collection.

7.1.1.8 DeleteObject method

The DeleteObject method removes the specified object reference from the collection. If the input object reference is not part of this collection, a CollectionException object whose ExceptionCode method returns EO-NOT-IN-COLLECTION is raised.

If the specified object reference is a member of the collection more than once, all object references are removed from the collection. For each object reference deleted, the ordinal positions of all object references with ordinal positions greater than that object reference are decremented by 1. If the current reference is deleted, the object reference that occurs next in sequence becomes the current reference.

NOTE Deleting an object reference invalidates all iterators associated with this collection.

7.1.1.9 Exists method

The Exists method tests to see if the specified object reference is a member of the set of object references managed by the collection class instance. If the object reference is a member of the collection, the method returns 1 (true); otherwise the method returns 0 (false).

7.1.1.10 Ordinal method

The Ordinal method returns the ordinal position of the current reference. If there is no current reference, this method returns 0.

7.1.1.11 ReturnCurrent method

The ReturnCurrent method returns the current reference. If there is no current reference, a CollectionException object whose ExceptionCode method returns EO-NO-CURRENT-OBJECT-REFERENCE is raised and the method returns the NULL object reference.

7.1.1.12 ReturnFirst method

The ReturnFirst method returns the object reference with an ordinal position of 1 from the collection. If there are no object references in this collection, a CollectionException object whose ExceptionCode method returns EO-EMPTY is raised and the method returns the NULL object reference.

7.1.1.13 ReturnLast method

The ReturnLast method returns the object reference with the highest ordinal position from the collection. If there are no object references in this collection, a CollectionException object whose ExceptionCode method returns EO-EMPTY is raised and the method returns the NULL object reference.

7.1.1.14 ReturnNext method

The ReturnNext method returns the object reference with an ordinal position 1 greater than the ordinal position of the current reference. If the current reference is the last object reference in the collection, the method returns the NULL object reference and a CollectionException object whose ExceptionCode method returns EO-END-OF-COLLECTION is raised. If there is no current reference, the ReturnNext method returns the first object reference in the collection.

7.1.1.15 ReturnObject method

The ReturnObject method returns the object reference whose ordinal position within the collection is equal to inOrdinal. If inOrdinal is less than one or greater than the number of object references in the collection, the method returns the NULL object reference and a CollectionException object whose ExceptionCode method returns EO-NOT-IN-COLLECTION object is raised.

7.1.1.16 ReturnPrevious method

The ReturnPrevious method returns the object reference with an ordinal position 1 less than the ordinal position of the current reference. If the current reference is the first object reference in the collection, the method returns the NULL object reference and a CollectionException object whose ExceptionCode method returns EO-BEGINNING-OF-COLLECTION is raised. If there is no current reference, The ReturnPrevious method returns the last object reference in the collection.

7.1.2 Sequencing method

Sequencing methods are used by the CreateIterator method of the Collection class and by the NewSortedCollection method of the SortedCollection class to determine the sequence in which the object references are returned by the resultant iterator or sorted collection. If the sequencing methods for two object references return data items that compare equal, those object references are returned in the sequence that they were added to the collection. Rules for comparison are specified in 8.8.4.1.1, Relation conditions, in ISO/IEC 1989-2002 Programming Language COBOL.

```
Method-id. method-name.
Data division.
Linkage section.
01  outString pic N any length.
Procedure division returning outString.
End method method-name.
```

Method-name shall be the name that is passed as a parameter to the CreateIterator method of the Collection class, to the NewSortedCollection method of the SortedCollection class, or to the NewIterator method of the Iterator class. Each object in the collection shall define a method with the name method-name. That method shall conform to the above specification.

The maximum length for outString is 1024 national characters.

7.2 OrderedCollection class

The OrderedCollection class inherits from the Collection Class. The OrderedCollection class contains additional methods for inserting objects relative to other objects within the collection. Unlike the Collection class, adding object references to an instance of the OrderedCollection using the methods of the OrderedCollection class can change the ordinal position of other object references within the collection.

NOTE Adding an object reference invalidates any iterators associated with this collection.

Object references may be added to an instance of the OrderedCollection class using the AddObject method of the Collection class. This is equivalent to using the AddLast method of the OrderedCollection class.

7.2.1 OrderedCollection instance interface

Ordered collection instance interface

```
Interface-id. OrderedCollectionInterface
  Inherits CollectionInterface.
Environment division.
Configuration section.
Repository.
    Interface CollectionInterface
    Interface OrderedCollectionExInterface.
*>
Procedure division.
    Method-id. AddAfter.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddAfter.
*>
    Method-id. AddBefore.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddBefore.
*>
    Method-id. AddFirst.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddFirst.
*>
    Method-id. AddLast.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddLast.
*>
End Interface OrderedCollectionInterface.
```

Parameterized ordered collection instance Interface

```
Interface-id. ParamOrderedCollInterface
  Inherits BaseInterface
  Using paramClass.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface OrderedCollectionExInterface
    Interface IteratorInterface
    Class     ParamClass.
*>
Procedure division.
    Method-id. AddAfter.
    Data division.
```

```
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddAfter.
*>
    Method-id. AddBefore.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddBefore.
*>
    Method-id. AddFirst.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddFirst.
*>
    Method-id. AddLast.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddLast.
*>
*>  The following methods are described in
*>      2.1, Collection Class Interface
*>
    Method-id. AddObject.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method AddObject.
*>
    Method-id. CompareCollection.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method CompareCollection.
*>
    Method-id. CopyCollection.
    Data division.
    Linkage section.
    01  outObject usage object reference active-class.
    Procedure division returning outObject.
    End method CopyCollection.
*>
    Method-id. CountObjects.
    Data division.
    Linkage section.
    01  outBinary usage binary-long.
    Procedure division returning outBinary.
    End method CountObjects.
*>
    Method-id. CreateIterator.
    Data division.
    Linkage section.
    01  inSequence pic N any length.
    01  outIterator usage object reference IteratorInterface.
```

```
    Procedure division using optional inSequence returning outIterator
      raising OrderedCollectionExInterface.
    End method CreateIterator.
*>
    Method-id. DeleteAll.
    Procedure division.
    End method DeleteAll.
*>
    Method-id. DeleteCurrent.
    Procedure division
      raising OrderedCollectionExInterface.
    End method DeleteCurrent.
*>
    Method-id. DeleteObject.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising OrderedCollectionExInterface.
    End method DeleteObject.
*>
    Method-id. Exists.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method Exists.
*>
    Method-id. Ordinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal.
    End method Ordinal.
*>
    Method-id. ReturnCurrent.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising OrderedCollectionExInterface.
    End method ReturnCurrent.
*>
    Method-id. ReturnFirst.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising OrderedCollectionExInterface.
    End method ReturnFirst.
*>
    Method-id. ReturnLast.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising OrderedCollectionExInterface.
    End method ReturnLast.
*>
    Method-id. ReturnNext.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising OrderedCollectionExInterface.
    End method ReturnNext.
*>
```

```
    Method-id. ReturnObject.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outObject usage object reference paramClass.
    Procedure division using by value inOrdinal returning outObject
      raising OrderedCollectionExInterface.
    End method ReturnObject.
*>
    Method-id. ReturnPrevious.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising OrderedCollectionExInterface.
    End method ReturnPrevious.
*>
End Interface ParamOrderedCollInterface.
```

7.2.1.1 AddAfter method

The AddAfter method adds an object reference to the collection and assigns the object reference an ordinal position equal to 1 greater than the ordinal position of the current reference. The ordinal position of any object reference with an ordinal position greater than the ordinal position of the current reference is incremented by 1. If the object reference is NULL, an OrderedCollectionException object whose ExceptionCode method returns EO-NULL is raised and the object reference is not added to the collection; otherwise, if there is no current reference, an OrderedCollectionException object whose ExceptionCode method returns EO-NO-CURREN-OBJECT-REFERENCE is raised and the object reference is not added to the collection.

If resources do not exist to add the object reference, an OrderedCollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and the object reference is not added to the collection.

7.2.1.2 AddBefore method

The AddBefore method adds an object reference to the collection and assigns the object reference an ordinal position equal to the ordinal position of the current reference. The ordinal position of the current reference and of any object reference with an ordinal position greater than the ordinal position of the current reference is incremented by 1. If the object reference is NULL, an OrderedCollectionException object whose ExceptionCode method returns EO-NULL is raised and the object reference is not added to the collection; otherwise, if there is no current reference, an OrderedCollectionException object whose ExceptionCode method returns EO-NO-CURRENT-OBJECT-REFERENCE is raised and the object reference is not added to the collection.

If resources do not exist to add the object reference, an OrderedCollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and the object reference is not added to the collection.

7.2.1.3 AddFirst method

The AddFirst method adds an object reference to the collection and assigns the object reference an ordinal position equal to 1. The ordinal position of all other object references within the collection is incremented by 1. If there are no object references in the collection, the object reference is added to the collection. If the object reference is NULL, an OrderedCollectionException object whose ExceptionCode method returns EO-NULL is raised and the object reference is not added to the collection.

If resources do not exist to add the object reference, an OrderedCollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and the object reference is not added to the collection.

7.2.1.4 AddLast method

The AddLast method adds an object reference to the collection and assigns the object reference an ordinal position equal to 1 greater than the ordinal position of the highest ordinal position associated with collection. If there are no object references in the collection, the object reference is assigned ordinal position 1. If the object reference is NULL, an OrderedCollectionException object whose ExceptionCode method returns EO-NULL is raised and the object reference is not added to the collection.

If resources do not exist to add the object reference, an OrderedCollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and the object reference is not added to the collection.

7.3 KeyedCollection class

The KeyedCollection class inherits from the Collection class.

The KeyedCollection class associates a key value with each object reference that is a member of the collection. An object reference may be returned from the collection by invoking the ReturnKeyedObject method passing the object reference's key value as an argument.

Object references cannot be added to a KeyedCollection using the AddObject method of the Collection class.

7.3.1 KeyedCollection instance interface

Keyed collection interface

```
Interface-id. KeyedCollectionInterface
  Inherits CollectionInterface.
Environment division.
Configuration section.
Repository.
    Interface CollectionInterface
    Interface KeyedCollectionExInterface.
*>
Procedure division.
    Method-id. AddKeyed.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    01  inKey pic N any length.
    Procedure division using by value inObject
      by reference inKey
      raising KeyedCollectionExInterface.
    End method AddKeyed.
*>
    Method-id. ReturnKeyFromCurrent.
    Data division.
    Linkage section.
    01  outKey pic N any length.
    Procedure division returning outKey
      raising KeyedCollectionExInterface.
    End method ReturnKeyFromCurrent.
*>
    Method-id. ReturnKeyFromOrdinal.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outKey pic N any length.
    Procedure division using by value inOrdinal returning outKey
      raising KeyedCollectionExInterface.
    End method ReturnKeyFromOrdinal.
*>
```

```
    Method-id. ReturnKeyedObject.
    Data division.
    Linkage section.
    01  inKey pic N any length.
    01  outObject usage object reference.
    Procedure division using inKey returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnKeyedObject.
*>
End Interface KeyedCollectionInterface.
```

Parameterized keyed collection interface

```
Interface-id. ParamKeyedCollectionInterface
  Inherits BaseInterface
  Using paramClass.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface KeyedCollectionExInterface
    Interface IteratorInterface
    Class     ParamClass.
*>
Procedure division.
    Method-id. AddKeyed.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    01  inKey pic N any length.
    Procedure division using by value inObject
      by reference inKey
      raising KeyedCollectionExInterface.
    End method AddKeyed.
*>
    Method-id. ReturnKeyFromCurrent.
    Data division.
    Linkage section.
    01  outKey pic N any length.
    Procedure division returning outKey
      raising KeyedCollectionExInterface.
    End method ReturnKeyFromCurrent.
*>
    Method-id. ReturnKeyFromOrdinal.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outKey pic N any length.
    Procedure division using by value inOrdinal returning outKey
      raising KeyedCollectionExInterface.
    End method ReturnKeyFromOrdinal.
*>

    Method-id. ReturnKeyedObject.
    Data division.
    Linkage section.
    01  inKey pic N any length.
    01  outObject usage object reference paramClass.
    Procedure division using inKey returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnKeyedObject.
*>
*>
*>  The following methods are described in
*>      2.1, Collection Class Interface
*>
    Method-id. CompareCollection.
    Data division.
```

```
    Linkage section.
    01  inObject usage object reference.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method CompareCollection.
*>
    Method-id. CopyCollection.
    Data division.
    Linkage section.
    01  outObject usage object reference active-class.
    Procedure division returning outObject.
    End method CopyCollection.
*>
    Method-id. CountObjects.
    Data division.
    Linkage section.
    01  outBinary usage binary-long.
    Procedure division returning outBinary.
    End method CountObjects.
*>
    Method-id. CreateIterator.
    Data division.
    Linkage section.
    01  inSequence pic N any length.
    01  outIterator usage object reference IteratorInterface.
    Procedure division using optional inSequence returning outIterator
      raising KeyedCollectionExInterface.
    End method CreateIterator.
*>
    Method-id. DeleteAll.
    Procedure division.
    End method DeleteAll.
*>
    Method-id. DeleteCurrent.
    Procedure division
      raising KeyedCollectionExInterface.
    End method DeleteCurrent.
*>
    Method-id. DeleteObject.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising KeyedCollectionExInterface.
    End method DeleteObject.
*>
    Method-id. Exists.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method Exists.
*>
    Method-id. Ordinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal.
    End method Ordinal.
*>
    Method-id. ReturnCurrent.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnCurrent.
```

```
*>
    Method-id. ReturnFirst.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnFirst.
*>
    Method-id. ReturnLast.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnLast.
*>
    Method-id. ReturnNext.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnNext.
*>
    Method-id. ReturnObject.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outObject usage object reference paramClass.
    Procedure division using by value inOrdinal returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnObject.
*>
    Method-id. ReturnPrevious.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising KeyedCollectionExInterface.
    End method ReturnPrevious.
*>
End Interface ParamKeyedCollectionInterface.
```

7.3.1.1 AddKeyed method

The AddKeyed method is equivalent to the AddObject method of the Collection class; however, the AddKeyed method of the KeyedCollection class requires an additional parameter, the key value. The key value is a national data item that may be used to retrieve the object reference from the collection. If the specified key value duplicates a key value that is already part of the collection, a KeyedCollectionException object whose ExceptionCode method returns EO-DUPLICATE-KEY is raised and the collection is unchanged. All other rules for the AddKeyed method are as specified in 7.1.2.1, AddObject method.

NOTE The addition of an object reference invalidates all iterators associated with this collection.

If resources do not exist to add the object reference, a KeyedCollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and the object reference is not added to the collection.

7.3.1.2 ReturnKeyFromCurrent method

The ReturnKeyFromCurrent method returns the key value that was associated with the current reference when that object reference was added to the KeyedCollection. If there is no current reference, a KeyedCollectionException object whose ExceptionCode method returns EO-NO-CURRENT-OBJECT-REFERENCE is raised and the returned data item is a zero-length national data

item. If the size of the returning data item is not large enough to contain the key value, a KeyedCollectionException object whose ExceptionCode method returns EO-TRUNCATED-KEY is raised and the key value is truncated to the size of the returning data item.

7.3.1.3 ReturnKeyFromOrdinal method

The ReturnKeyFromOrdinal method returns the key value that was associated with the object reference at the ordinal position specified in inOrdinal. If there is no object reference at the specified ordinal position, a KeyedCollectionException object whose ExceptionCode method returns EO-INVALID-ORDINAL is raised and the returned data item is a zero-length national data item. If the size of the returning data item is not large enough to contain the key value, a KeyedCollectionException object whose ExceptionCode method returns EO-TRUNCATED-KEY is raised and the returned key value is truncated to the size of the returning data item.

7.3.1.4 ReturnKeyedObject method

The ReturnKeyedObject method returns the object reference that was associated with the key value when that object reference was added to the KeyedCollection. If there is no matching key value associated with a member of the collection, a KeyedCollectionException object whose ExceptionCode method returns EO-NOT-IN-COLLECTION is raised and the ReturnKeyedObject method returns the NULL object reference. When comparing inKey with the key value associated with each member of the KeyedCollection, the shorter of the two comperands is extended to the size of the longer operand and padded with national spaces.

7.3.2 Overridden methods

These methods are defined in the Collection instance interface and are overridden in a class that implements KeyedCollectionInterface.

7.3.2.1 AddObject method

The AddObject method of the KeyedCollection class always raises a KeyedCollectionException object whose ExceptionCode method returns EO-NO-KEY and the collection is unchanged.

NOTE 1 The override of the AddObject method prevents the addition of object references to a keyed collection without a key. All object references in a keyed collection are added using the AddKeyed method.

NOTE 2 The parameterized keyed collection class does not contain the AddObject method.

7.4 SortedCollection class

The SortedCollection class inherits from the Collection class.

The SortedCollection class sequences object references in a collection by a national character value returned from a sequencing method associated with each object reference in that collection.

The addition of object references to a Sorted Collection can change the ordinal position of other object references within the collection.

7.4.1 SortedCollection factory interface

Sorted collection factory interface

```
Interface-id. SortedCollFactoryInterface
    Inherits BaseFactoryInterface.
Environment division.
Configuration section.
Repository.
    Interface BaseFactoryInterface
*>
```

```
Procedure division.
    Method-id. NewSortedCollection.
    Data division.
    Linkage section.
    01  inSequence pic N any length.
    01  outCollection usage object reference active-class.
    Procedure division using inSequence returning outCollection
      raising SortedCollectionExInterface.
    End method NewSortedCollection.
*>
End Interface SortedCollFactoryInterface.
```

Parameterized sorted collection factory interface

```
Interface-id. ParamSortedColFactoryInterface
    Inherits BaseFactoryInterface
    Expands using paramClass.
Environment division.
Configuration section.
Repository.
    Interface BaseFactoryInterface
    Interface SortedCollectionExInterface
*>
Procedure division.
    Method-id. NewSortedCollection.
    Data division.
    Linkage section.
    01  inSequence pic N any length.
    01  outCollection usage object reference active-class.
    Procedure division using inSequence returning outCollection
      raising SortedCollectionExInterface.
    End method NewSortedCollection.
*>
End Interface ParamSortedColFactoryInterface.
```

7.4.1.1 NewSortedCollection method

The NewSortedCollection method creates and returns a SortedCollection. The invocation of NewSortedCollection shall specify a sequencing method argument as described in 7.1.2, Sequencing method. The name of the sequencing method is passed as an argument to the NewSortedCollection. Each time an object reference is added to a sorted collection, the sequencing method is invoked to obtain the sequencing value. Once a sequencing value has been associated with an object reference in a collection, that sequencing value remains constant as long as that object reference is a member of that collection.

If resources do not exist to create the SortedCollection, a SortedCollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and NewSortedCollection returns the NULL object reference.

7.4.2 Overridden factory methods

The method below is defined in BaseFactoryInterface and is overridden in a class which implements SortedCollectionFactoryInterface.

7.4.2.1 New method

The New method of the SortedCollection class returns the NULL object reference and raises a SortedCollectionException object whose ExceptionCode method returns EO-NEW.

7.4.3 SortedCollection instance interface

Sorted collection instance interface

```
Interface-id. SortedCollectionInterface
  Inherits CollectionInterface.
Environment division.
Configuration section.
Repository.
    Interface CollectionInterface
    Interface SortedCollectionExInterface.
End Interface SortedCollectionInterface.
```

Parameterized sorted collection instance interface

```
Interface-id. ParamSortedCollectionInterface
  Inherits BaseInterface
  Using paramClass.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface SortedCollectionExInterface
    Interface IteratorInterface
    Class     ParamClass.
*>
*>
*>  The following methods are described in
*>      2.1, Collection Class Interface
*>
    Method-id. CompareCollection.
    Data division.
    Linkage section.
    01  inObject usage object reference.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method CompareCollection.
*>
    Method-id. CopyCollection.
    Data division.
    Linkage section.
    01  outObject usage object reference active-class.
    Procedure division returning outObject.
    End method CopyCollection.
*>
    Method-id. CountObjects.
    Data division.
    Linkage section.
    01  outBinary usage binary-long.
    Procedure division returning outBinary.
    End method CountObjects.
*>
    Method-id. CreateIterator.
    Data division.
    Linkage section.
    01  inSequence pic N any length.
    01  outIterator usage object reference IteratorInterface.
    Procedure division using optional inSequence returning outIterator
      raising SortedCollectionExInterface.
    End method CreateIterator.
*>
    Method-id. DeleteAll.
    Procedure division.
    End method DeleteAll.
*>
    Method-id. DeleteCurrent.
    Procedure division
```

```
      raising SortedCollectionExInterface.
    End method DeleteCurrent.
*>
    Method-id. DeleteObject.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    Procedure division using by value inObject
      raising SortedCollectionExInterface.
    End method DeleteObject.
*>
    Method-id. Exists.
    Data division.
    Linkage section.
    01  inObject usage object reference paramClass.
    01  outBoolean usage bit pic 1.
    Procedure division using by value inObject returning outBoolean.
    End method Exists.
*>
    Method-id. Ordinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal.
    End method Ordinal.
*>
    Method-id. ReturnCurrent.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising SortedCollectionExInterface.
    End method ReturnCurrent.
*>
    Method-id. ReturnFirst.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising SortedCollectionExInterface.
    End method ReturnFirst.
*>
    Method-id. ReturnLast.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising SortedCollectionExInterface.
    End method ReturnLast.
*>
    Method-id. ReturnNext.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising SortedCollectionExInterface.
    End method ReturnNext.
*>
    Method-id. ReturnObject.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outObject usage object reference paramClass.
    Procedure division using by value inOrdinal returning outObject
      raising SortedCollectionExInterface.
    End method ReturnObject.
*>
```

```
    Method-id. ReturnPrevious.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising SortedCollectionExInterface.
    End method ReturnPrevious.
*>
End Interface ParamSortedCollectionInterface.
```

7.4.4 Overridden instance methods

The method below is defined in CollectionInterface and is overridden in a class that implements the SortedCollectionInterface.

7.4.4.1 AddObject method

The AddObject method of the SortedCollection class adds the specified object reference to the collection and assigns the object reference an ordinal position that places the object reference in sequence by the national character value returned by the sequencing method of the object whose reference is being added. The ordinal position of any object reference with an ordinal position greater than the ordinal position of the current reference is incremented by 1. If the specified object reference is NULL a SortedCollectionException object whose SortedExceptionCode method returns EO-NULL is raised and the object reference is not added to the collection; otherwise, if there is no sequencing method in the object being added, or the method does not conform to the specification in 7.1.2, Sequencing method, a SortedCollectionException object whose ExceptionCode method returns EO-INVALID-SEQUENCING-METHOD is raised and the object reference is not added to the collection.

If resources do not exist to add the object reference, a SortedCollectionException object whose ExceptionCode method returns EO-RESOURCE is raised and the object reference is not added to the collection.

7.5 Iterator class

An iterator is an object used to access the object references that are members of a collection. Iterator instances are created using the CreateIterator method described in 7.1.1.5, CreateIterator method, or the NewIterator method described in 7.5.1.1, NewIterator method. Except for changes made by the DeleteCurrent method of the iterator, any change in the membership of the underlying collection invalidates that iterator. Any reference to an invalidated iterator raises an IteratorException object whose ExceptionCode method returns EO-INVALIDATED-ITERATOR.

The iterator accesses the collected object references sequentially as specified in 7.1.1.5, CreateIterator method. The iterator instance methods that return object references from an iterator return those object references in the sequence specified at the creation of the iterator.

NOTE If there is no sequencing method defined at the creation of the iterator, object references are returned in the order that they were added to the associated collection. If the associated collection is an ordered collection, this sequence might differ from the sequence that would be returned using the methods of the associated collection.

The current reference for an iterator is the last object reference returned by any of its instance methods. If there are multiple iterators associated with the associated collection, each iterator has its own current reference.

NOTE The current reference for an iterator is not related to the current reference in the associated collection.

If the iterator is associated with a collection that contains no object references, invocation of any of that iterator's instance methods raises an IteratorException object whose ExceptionCode method returns EO-EMPTY. If that method returns an object reference, that object reference is the NULL object reference.

7.5.1 Iterator factory interface

Iterator factory interface

```
Interface-id. IteratorFactoryInterface
  Inherits BaseInterface.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface IteratorExceptionInterface
    Interface Collection.
*>
Procedure division.
    Method-id. NewIterator.
    Data division.
    Linkage section.
    01  inCollection usage object reference Collection.
    01  inSequence pic N(31).
    01  outIterator usage object reference active-class.
    Procedure division using inCollection
      optional inSequence returning outIterator
      raising IteratorExceptionInterface.
    End method NewIterator.
*>
End IteratorFactoryInterface.
```

Parameterized Iterator factory interface

```
Interface-id. IteratorFactoryInterface
  Inherits BaseInterface
  Using paramClass.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface IteratorExceptionInterface
    Interface Collection
    Class     paramClass.
*>
Procedure division.
    Method-id. NewIterator.
    Data division.
    Linkage section.
    01  inCollection usage object reference Collection.
    01  inSequence pic N(31).
    01  outIterator usage object reference active-class.
    Procedure division using inCollection
      optional inSequence returning outIterator
      raising IteratorExceptionInterface.
    End method NewIterator.
*>
End IteratorFactoryInterface.
```

7.5.1.1 NewIterator method

The NewIterator method creates and returns an iterator instance for the collection inCollection as specified in 7.5, Iterator class. An optional sequencing method parameter may be specified. If this parameter does not refer to a valid method as defined in 7.1.2, Sequencing method, an IteratorException object whose ExceptionCode method returns EO-INVALID-SEQUENCING-METHOD is raised and NewIterator returns the NULL object reference. If the sequencing method parameter is omitted, object references are returned by the iterator in the sequence that they were added to the collection. The current reference of the iterator is set as described in 7.5.2.5, ReturnFirst method.

NOTE 1 There may be more than one iterator associated with a collection. These iterators may differ in sequence.

NOTE 2 If the collection is an ordered collection and the sequencing method is omitted, the sequence in which object references are returned by the iterator can differ from the sequence in which object references are returned by the collection.

If resources do not exist to create the iterator, an IteratorException object whose ExceptionCode method returns EO-RESOURCE is raised and NewIterator returns the NULL object reference.

A parameterized iterator may only be created using the NewIterator method.

7.5.2 Iterator instance interface

Iterator Interface

```
Interface-id. IteratorInterface
  Inherits BaseInterface.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface IteratorExceptionInterface.
*>
Procedure division.
    Method-id. CollectionOrdinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal
      raising IteratorExceptionInterface.
    End method CollectionOrdinal.
*>
    Method-id. DeleteCurrent.
    Procedure division
      raising IteratorExceptionInterface.
    End method DeleteCurrent.
*>
    Method-id. IteratorOrdinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal
      raising IteratorExceptionInterface.
    End method IteratorOrdinal.
*>
    Method-id. ReturnCurrent.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnCurrent.
*>
    Method-id. ReturnFirst.
    Data division.
    Linkage section.
```

```
    01  outObject usage object reference.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnFirst.
*>
    Method-id. ReturnLast.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnLast.
*>
    Method-id. ReturnNext.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnNext.
*>
    Method-id. ReturnOrdinal.
    Data division.
    Linkage section.
    01  inOrdinal usage binary-long.
    01  outObject usage object reference.
    Procedure division using by value inOrdinal
      returning outObject
      raising IteratorExceptionInterface.
    End method ReturnOrdinal.
*>
    Method-id. ReturnPrevious.
    Data division.
    Linkage section.
    01  outObject usage object reference.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnPrevious.
*>
End Interface IteratorInterface.
```

Parameterized Iterator Interface

```
Interface-id. ParamIteratorInterface
  Inherits BaseInterface
  Using paramClass.
Environment division.
Configuration section.
Repository.
    Interface BaseInterface
    Interface IteratorExceptionInterface.
    Class     paramClass.
*>
Procedure division.
    Method-id. CollectionOrdinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal
      raising IteratorExceptionInterface.
    End method ReturnOrdinal.
*>
    Method-id. DeleteCurrent.
    Procedure division
      raising IteratorExceptionInterface.
    End method CollectionOrdinal.
*>
```

```
    Method-id. IteratorOrdinal.
    Data division.
    Linkage section.
    01  outOrdinal usage binary-long.
    Procedure division returning outOrdinal
      raising IteratorExceptionInterface.
    End method IteratorOrdinal.
*>
    Method-id. ReturnCurrent.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnCurrent.
*>
    Method-id. ReturnFirst.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnFirst.
*>
    Method-id. ReturnLast.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnLast.
*>
    Method-id. ReturnNext.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnNext.
*>
    Method-id. ReturnOrdinal.
    Data division.
    Linkage section.
    01  outBinary usage binary-long.
    Procedure division returning outBinary
      raising IteratorExceptionInterface.
    End method ReturnOrdinal.
*>
    Method-id. ReturnPrevious.
    Data division.
    Linkage section.
    01  outObject usage object reference paramClass.
    Procedure division returning outObject
      raising IteratorExceptionInterface.
    End method ReturnPrevious.
*>
End Interface ParamIteratorInterface.
```


7.5.2.1 CollectionOrdinal method

The CurrentOrdinal method returns the ordinal position of the current reference in the sequence of object references referenced by the associated collection. If there is no current reference, an IteratorException object whose ExceptionCode method returns EO-INVALID-ORDINAL is raised and the method returns the NULL object reference.

7.5.2.2 DeleteCurrent method

The DeleteCurrent method removes the current reference from the associated collection. The next sequential object reference referenced by the iterator becomes the current reference. If the object reference removed was the last object reference in the sequence referenced by the iterator, there is no current reference. If the DeleteCurrent method is invoked when there is no current reference, an IteratorException object whose ExceptionCode method returns EO-NO-CURRENT-OBJECT-REFERENCE is raised. If the object reference deleted is the current reference for the associated collection, the current reference is set as described in 7.1.1.7, DeleteCurrent method.

NOTE Deleting an object reference invalidates any other iterators associated with this collection.

7.5.2.3 IteratorOrdinal method

The IteratorOrdinal method returns the ordinal position of the current reference in the sequence of object references referenced by the iterator. If there is no current reference, an IteratorException object whose ExceptionCode method returns EO-INVALID-ORDINAL is raised and the method returns the NULL object reference.

7.5.2.4 ReturnCurrent method

The ReturnCurrent method returns the object reference that is the current reference. If there is no current reference when the ReturnCurrent method is invoked, an IteratorException object whose ExceptionCode method returns EO-NO-CURRENT-OBJECT-REFERENCE is raised and the method returns the NULL object reference.

7.5.2.5 ReturnFirst method

The ReturnFirst method returns the first object reference in the sequence of object references referenced by the iterator. If there are no object references within the collection associated with the iterator, an IteratorException object whose ExceptionCode method returns EO-EMPTY is raised and the method returns the NULL object reference.

7.5.2.6 ReturnLast method

The ReturnLast method returns the last object reference in the sequence of object references referenced by the iterator. If there are no object references within the collection associated with the iterator, an IteratorException object whose ExceptionCode method returns EO-EMPTY is raised and the method returns the NULL object reference.

7.5.2.7 ReturnNext method

The ReturnNext method returns the next object reference in the sequence of object references referenced by the iterator. If the current reference is the last sequential object reference referenced by the iterator, an IteratorException object whose ExceptionCode method returns EO-END-OF-COLLECTION is raised and the method returns the NULL object reference.

7.5.2.8 ReturnOrdinal method

The ReturnObject method returns the object reference whose ordinal position within the iterator is equal to inOrdinal. If inOrdinal is less than one or greater than the number of object references in the collection, the method returns the NULL object reference and a CollectionException object whose ExceptionCode method returns EO-NOT-IN-COLLECTION object is raised.

7.5.2.9 ReturnPrevious method

The ReturnPrevious method returns the previous object reference in the sequence of object references referenced by the iterator. If the current reference is the first sequential object reference referenced by the iterator, an IteratorException object whose ExceptionCode method returns EO-BEGINNING-OF-DATA is raised and the method returns the NULL object reference.

7.6 Collection Exception classes

The Collection Exception class inherits from the ExceptionInformation class and provides specific information on exceptions that can occur in the Collection Class methods. Each subclass of the collection class implements a subclass of the Collection Exception class to provide specific returned values for the ExceptionCode method.

7.6.1 Collection Exception interfaces

```
Interface-id. CollectionExInterface
  Inherits ExceptionInterface.
Environment division.
Configuration section.
Repository.
    Interface ExceptionInterface.
End Interface CollectionExInterface.

Interface-id. KeyedCollectionExInterface
  Inherits CollectionExInterface.
Environment division.
Configuration section.
Repository.
    Interface CollectionExInterface.
End Interface KeyedCollectionExInterface.

Interface-id. OrderedCollectionExInterface
  Inherits CollectionExInterface.
Environment division.
Configuration section.
Repository.
    Interface CollectionExInterface.
End Interface OrderedCollectionExInterface.

Interface-id. SortedCollectionExInterface
  Inherits CollectionExInterface.
Environment division.
Configuration section.
Repository.
    Interface CollectionExInterface.
End Interface SortedCollectionExInterface.

Interface-id. IteratorExceptionInterface
  Inherits CollectionExInterface.
Environment division.
Configuration section.
Repository.
    Interface CollectionExInterface.
End Interface IteratorExceptionInterface.
```

7.6.2 ExceptionCode property values

The ExceptionCode property contains exception code values set by the methods of the collection classes and the iterator classes for each type of exception that can occur.

7.6.2.1 Collection class ExceptionCode property values

The Collection class methods set the ExceptionCode property to one of the following exception code values:

ExceptionCode Value	Exception
EO-BEGINNING-OF-COLLECTION	A ReturnPrevious method was invoked, but the CurrentObject was the first object in the collection or a ReturnObject method was invoked with an ordinal position less than 1
EO-EMPTY	There are no object references in the collection
EO-END-OF-COLLECTION	A ReturnNext method was invoked, but the CurrentObject was the last object in the collection or a ReturnObject method was invoked with an ordinal position greater than the ordinal position of the last object reference in the collection
EO-INVALID-ORDINAL	The ordinal specified is not valid for this collection
EO-INVALID-SEQUENCING-METHOD	The class or interface of an object reference in the collection does not define the Sequencing method specified as a parameter to the CreateIterator method
EO-NO-CURRENT-OBJECT-REFERENCE	CurrentObject does not reference an object
EO-NOT-IN-COLLECTION	The object reference is not a member of the collection
EO-NULL	Object reference is null
EO-RESOURCE	Resources do not exist to complete the method

Table 1 — Collection class ExceptionCode property values

7.6.2.2 OrderedCollection class ExceptionCode property values

The OrderedCollection class methods set the ExceptionCode property to one of the following exception code values:

ExceptionCode Value	Exception
EO-RESOURCE	Resources do not exist to complete the method

Table 2 — OrderedCollection class ExceptionCode property values

7.6.2.3 KeyedCollection class ExceptionCode property values

The KeyedCollection class methods set the ExceptionCode property to the one of the following exception code values:

ExceptionCode Value	Exception
EO-DUPLICATE-KEY	The AddKeyed method was invoked with a key that duplicates a key contained in this collection.
EO-INVALID-KEY	The ReturnKeyedObject method was invoked with a key that this collection does not contain
EO-NO-KEY	The AddObject method was invoked
EO-RESOURCE	Resources do not exist to complete the method
EO-TRUNCATED-KEY	The size allowed for the returning data item from the ReturnKeyFromCurrent and ReturnKeyFromOrdinal methods is not large enough to contain the key of the object specified in the method

Table 3 — KeyedCollection class ExceptionCode property values

7.6.2.4 SortedCollection class ExceptionCode property values

The SortedCollection class methods set the ExceptionCode property to one of the following exception code values:

ExceptionCode Value	Exception
EO-NEW	The New factory method was invoked
EO-INVALID-SEQUENCING-METHOD	The class or interface of an object reference in the collection does not define the Sequencing method specified as a parameter to the NewSortedCollection method
EO-RESOURCE	Resources do not exist to complete the method

Table 4 — SortedCollection class ExceptionCode property values

7.6.2.5 Iterator class ExceptionCode property values

The Iterator class methods set the ExceptionCode property to one of the following exception code values:

ExceptionCode Value	Exception
EO-INVALID-SEQUENCING-METHOD	The class or interface of an object reference in the collection does not define the Sequencing method specified as a parameter to the NewIterator method
EO-INVALIDATED-ITERATOR	The membership of the collection associated with the iterator has changed externally to the iterator
EO-RESOURCE	Resources do not exist to complete the method

Table 5 — Iterator class ExceptionCode property values

Annex A
(normative)

Language element lists

A.1 Implementor-defined element list

The following is a list of the language elements within this Technical Report that depend on implementor definition to complete the specification of the elements. Each element is defined as required, optional, or conditionally required. Furthermore, each element is defined as requiring (or not requiring) user documentation. These terms have the following meaning:

Required: The element shall be provided by the implementor. When the element is part of a feature that is optional or processor-dependent, the item is not required if the optional or processor-dependent feature is not implemented.

Optional: The element may be provided at the implementor's option.

Conditionally required: If the associated feature or language element is implemented then this element is also required.

Documentation required: If the element is provided by the implementor, the implementor's user documentation shall document the element or shall reference other documentation that fulfills this requirement.

[1] ExceptionMessage (content when a CollectionException object is raised). This item is required. This item shall be documented in the implementor's user documentation. (16.2.4 ExceptionMessage method)

A.2 Undefined language element list

The following are language elements within this Technical Report that are explicitly undefined:

None.

Annex B
(informative)

Unresolved technical issues

B.1 General

Some technical issues have proven difficult to resolve and have not been addressed in the current design. Those issues are presented here in order to obtain feedback from reviewers and early implementors.

1) Because ISO/IEC 1989:2002 does not support method overloading, and this feature was too complicated to include in this Technical Report, some methods that required a variable number of parameters are named uniquely. If method overloading is added to the next edition of ISO/IEC 1989, the following methods should be renamed:

Class	Method	Renamed to
KeyedCollectionClass	AddKeyed	AddObject
SortedCollectionClass	NewSortedCollection	New

The old methods should then be excluded from that edition.

2) This technical report does not address the issues of localization with respect to the collating sequences that are used to determine the order of object references within a sorted collection, or a sequenced iterator. Optional parameters that allow for localized collating sequences should be added to the NewSortedCollection method, the CreateIterator method, and the NewIterator method.