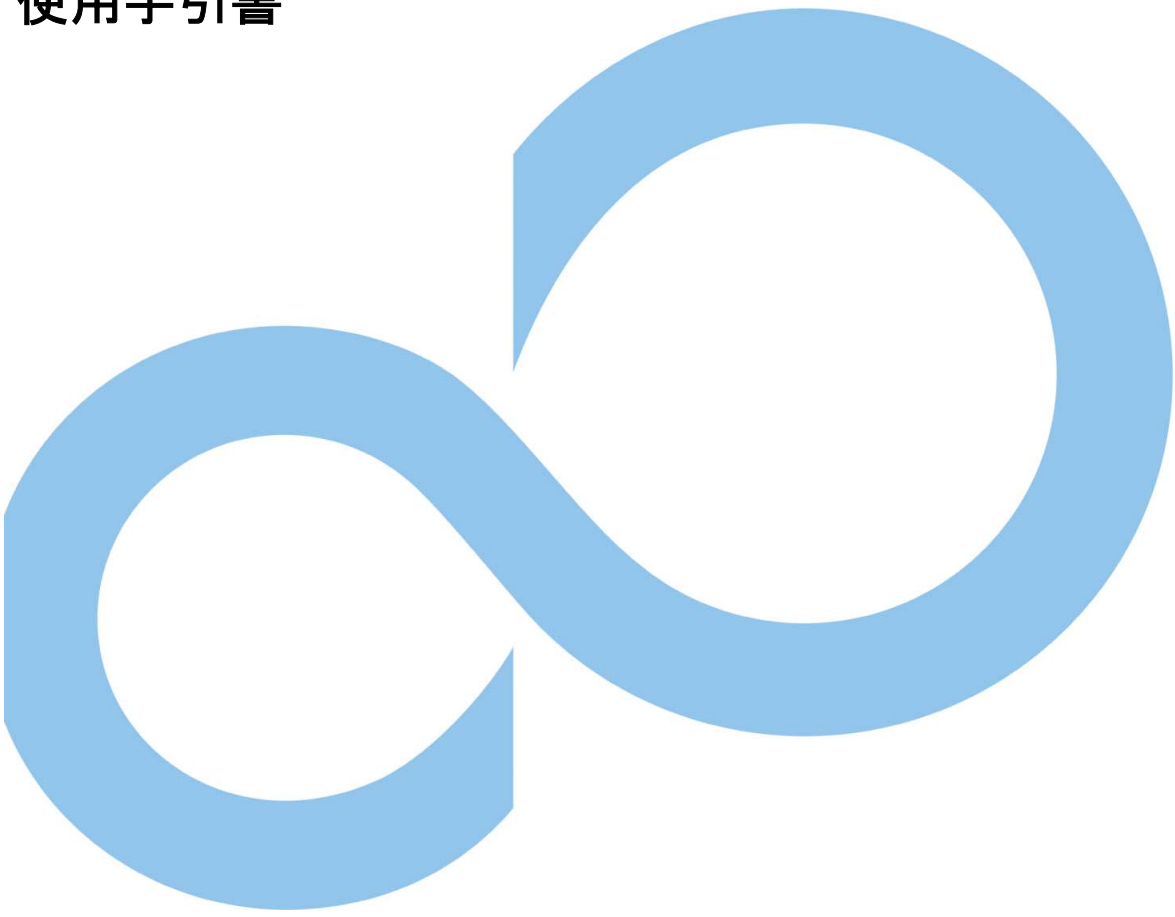
Microsoft® Windows® 95
Microsoft® Windows® 98
Microsoft® Windows® Me
Microsoft® Windows NT®
Microsoft® Windows® 2000
Microsoft® Windows® XP
Microsoft® Windows Server™ 2003

B1JW-6281-01Z2

コレクションクラスライブラリ for COBOL V7.0

使用手引書



FMVシリーズ,FMV-DESKPOWER,FMV-BIBLO

NetCOBOL

FUJITSU

まえがき

製品の呼び名について

本書では、各製品を次のように略記しています。あらかじめご了承ください。

- 「Microsoft(R) Windows(R) 95 operating system」
  →「Windows(R) 95」または「Windows」
- 「Microsoft(R) Windows(R) 98 operating system」
  →「Windows(R) 98」または「Windows」
- 「Microsoft(R) Windows(R) Millennium Edition」
  →「Windows(R) Me」または「Windows」
- 「Microsoft(R) Windows NT(R) Workstation operating system Version 4.0」
  →「Windows NT(R)」または「Windows」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0」
  →「Windows NT(R)」または「Windows」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0, Terminal Server Edition」
  →「Windows NT(R)」または「Windows」
- 「Microsoft(R) Windows NT(R) Server Network operating system, Enterprise Edition Version 4.0」
  →「Windows NT(R)」または「Windows」
- 「Microsoft(R) Windows(R) 2000 Professional operating system」
  →「Windows(R) 2000」または「Windows」
- 「Microsoft(R) Windows(R) 2000 Server operating system」
  →「Windows(R) 2000」または「Windows」
- 「Microsoft(R) Windows(R) 2000 Advanced Server operating system」
  →「Windows(R) 2000」または「Windows」
- 「Microsoft(R) Windows(R) XP Professional operating system」
  →「Windows(R) XP」または、「Windows」
- 「Microsoft(R) Windows(R) XP Home Edition operating system」
  →「Windows(R) XP」または、「Windows」
- 「Microsoft(R) Windows Server™ 2003, Standard Edition」
  →「Windows Server™ 2003」または、「Windows Server™ 2003 Standard Edition」
- 「Microsoft(R) Windows Server™ 2003 Enterprise Edition」
  →「Windows Server™ 2003」または、「Windows Server™ 2003 Enterprise Edition」

本書の目的

本書は、コレクションクラスライブラリを利用したCOBOLプログラムの作成から実行までに必要な情報を記載しています。

本書に特に記載されていないプログラム上の規則および使用方法については、NetCOBOLの規則および使用方法が適用されます。"COBOL文法書"および"NetCOBOL 使用手引書"を参照してください。

本書の対象読者

本書は、コレクションクラスライブラリを利用したCOBOLプログラムを開発される方を対象としています。

前提知識

本書を読むにあたって、以下の知識が必要です。

- COBOLの文法に関する基本的な知識
- COBOLのオブジェクト指向プログラミングに関する基本的な知識

本書の位置づけ

NetCOBOLシリーズにおける本書の位置付け、および関連マニュアルについては、“NetCOBOL 解説書”を参照してください。

表記上の約束

各メソッドの説明は次のように記載されています。

X.X　メソッド名

説明

メソッドの機能を簡単に説明しています。

記述形式

INVOKE *object* "**METHOD-NAME**"
USING *USING-PARAMETER*
RETURNING *RETURNING-PARAMETER*

引数

USING-PARAMETER［属性: 後述］

INVOKE文のUSING引数として指定するデータ項目について説明します。

復帰値

RETURNING-PARAMETER［属性: 後述］

INVOKE文のRETURNING引数として指定するデータ項目について説明します。

解説

メソッドの機能について解説します。

使用例

COBOLプログラムの使用例を記載します。

OBJECT

メソッドを実行するオブジェクトを表します。

METHOD-NAME

実行するメソッド名を表します。

属性

コレクションクラスライブラリの各メソッドのパラメータ(*1)の型(PICTURE句の文字列)を表します。プログラムには、記載されている属性に適合するデータ項目を指定します。適合については、"COBOL文法書"および"NetCOBOL 使用手引書"の『適合』を参照してください。

属性が"PIC X ANY LENGTH" となっているパラメータについては、任意の長さを持つ英数字のデータ項目を記述してください。

属性が "OBJECT REFERENCE CLASS OF SELF"となっているパラメータについては、メソッドを実行しているオブジェクトのクラスで型付けしたオブジェクト一意名か、型付けしていない(UNIVERSALな)オブジェクト一意名を指定する必要があります。

属性が"OBJECT REFERENCE SELF"となっているパラメータについては、返却されるオブジェクトのクラスで型付けしたオブジェクト一意名か、型付けしていない(UNIVERSALな)オブジェクト一意名を指定する必要があります。

(*1)

コレクションクラスライブラリの各メソッドの手続き部の見出し(PROCEDURE DIVISION)のUSINGまたはRETURNINGのデータ項目を指します。

本書で使用する書体と記号

書体および記号	意味
斜体文字	データ部で宣言するデータ項目を表します。
[あいうえお]	[ ]で囲まれた文字列は省略できることを表します。

登録商標について

本書に記載されている登録商標を、以下に示します。

Microsoft, Windows, Windows NTは、米国 Microsoft Corporationの米国およびその他の国における登録商標です。

2002年7月

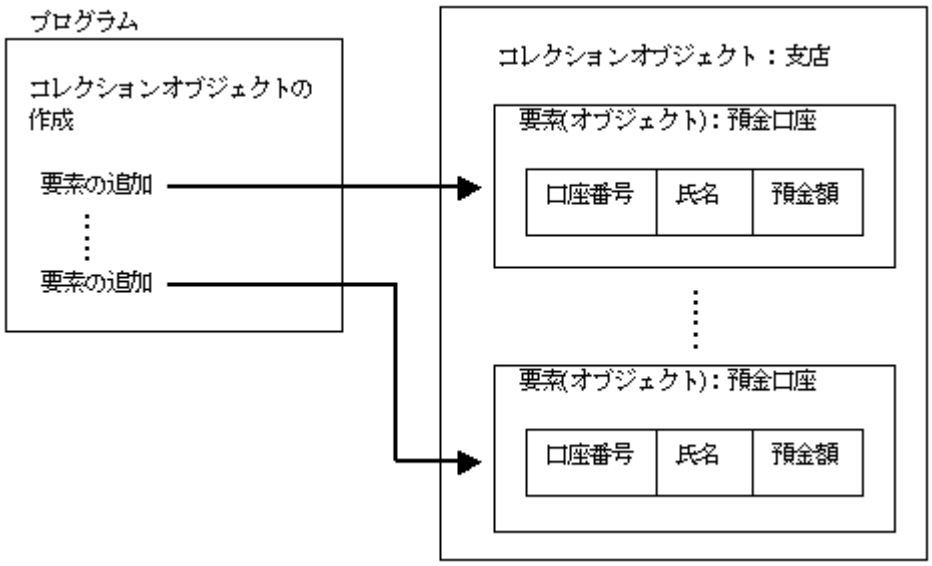
目次

第1章　コレクションクラスライブラリの概要

この章では、コレクションクラスライブラリについて説明します。

1.1　コレクションクラスライブラリとは

この章では、コレクションクラスライブラリについて説明します。
コレクションクラスライブラリとは、オブジェクトの集まり（コレクション）を扱うクラスの総称です。
例えば、預金口座のデータを支店の銀行で管理した場合を考えてみましょう。支店という名のコレクションを１つ作成します。支店コレクションに預金口座の情報を登録していきます。結果、支店コレクションは、その支店で取引のある口座の情報が登録されたオブジェクトの集まりになります。各口座の情報の変更などは、支店コレクションから情報を取り出して行います。つまり、各口座の情報を１つずつ管理するのではなく、複数の口座の情報を１つのコレクションとして管理します。



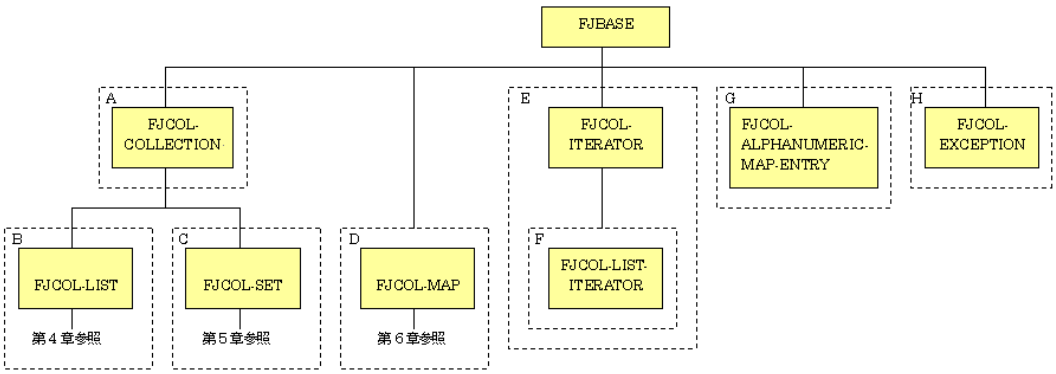
【図1.1 コレクションの概念図】

通常、上述したようなオブジェクトの集まりを管理するためには、アプリケーションにオブジェクト管理専用のプログラムを記述する必要があります。 しかし、オブジェクトの集まりを管理するコレクションクラスライブラリを利用すれば、オブジェクト管理専用のプログラムを記述する必要がなくなり、高い生産性を実現できます。

1.2　クラスの階層関係

コレクションクラスライブラリは、リストクラス、セットクラス、マップクラスなどから構成されています。
リストクラスは、要素の集まりを順序付けて扱うクラスです。
セットクラスは、要素の集まりを順序付けずに扱うクラスです。
マップクラスはキーと値からなるエントリの集まりを扱うクラスです。
その他のクラスは、どれもリストクラス／セットクラス／マップクラスの補助的な役割をはたします。
例えば、イテレータクラスは、リストクラス／セットクラス／マップクラスに登録されている要素やエントリを順に操作するために使用し、コレクション例外クラスは、処理中にエラーが発生した場合に例外情報を取り出すために使用します。
それぞれのクラスの機能については、後述の各章を参照してください。

a. コレクションクラス　… FJCOL-COLLECTION
b. リストクラス　… FJCOL-LIST、FJCOL-LINKED-LIST
c. セットクラス　… FJCOL-SET、FJCOL-IDENTITY-SET
d. マップクラス　… FJCOL-MAP、FJCOL-ALPHANUMERIC-MAP
e. イテレータクラス　… FJCOL-ITERATOR
f. リストイテレータクラス　… FJCOL-LIST-ITERATOR
g. マップエントリクラス　… FJCOL-ALPHANUMERIC-MAP-ENTRY
h. コレクション例外クラス　… FJCOL-EXCEPTION



【図1.2 コレクションクラスライブラリのクラス階層図】

第2章　開発方法

この章では、コレクションクラスライブラリを使用したアプリケーションの開発方法について説明します。

2.1　プログラムの作成

第3章以降で説明するコレクションクラスライブラリの各機能を使用して、プログラムを作成／編集します。
プログラム作成の際には、リポジトリ段落に使用するクラス名を指定してください。リポジトリ段落については、"COBOL文法書"の『リポジトリ段落』を参照してください。

2.2 プログラムの翻訳

プログラムを翻訳します。
翻訳の際には、コレクションクラスライブラリのリポジトリファイルが必要です。翻訳オプションにて、リポジトリファイルが格納されているフォルダを "リポジトリファイルの入力先フォルダ名" として指定してください。リポジトリファイルおよび翻訳オプションの指定方法については、"NetCOBOL 使用手引書"の『リポジトリファイル』を参照してください。

[例]

コレクションクラスライブラリを C:\Program Files\NetCOBOL にインストールし、翻訳オプションREPINでリポジトリファイルの入力先フォルダを指定する場合

```
REPIN("C:\Program Files\NetCOBOL\Rep")
```

また、コレクションクラスライブラリでは、例外エラーコードの判定のために使用する記号定数の宣言を持った登録集ファイルを提供しています。
この登録集ファイルを使用する場合は、翻訳オプションで、登録集ファイルが格納されているフォルダを"登録集ファイルのフォルダ"として指定する必要があります。

[例]

コレクションクラスライブラリを C:\Program Files\NetCOBOL にインストールし、翻訳オプションLIBで登録集ファイルのフォルダを指定する場合

```
LIB("C:\Program Files\NetCOBOL")
```

マルチスレッドプログラムの翻訳については、マルチスレッドプログラムの作成を参照してください。
Unicodeプログラムの翻訳については、Unicodeプログラムの作成を参照してください。

2.3　オブジェクトファイルのリンク

オブジェクトファイルをリンクし、実行可能ファイルを作成します。
オブジェクトファイルをリンクする際は、インポートライブラリ"F3BICCOL.LIB"をリンクする必要があります。ただし、プログラムの構造が動的プログラム構造である場合は、インポートライブラリ"F3BICCOL.LIB"は不要です。プログラムの構造については、"NetCOBOL 使用手引書"の『プログラム構造』を参照してください。
インポートライブラリ"F3BICCOL.LIB"は、コレクションクラスライブラリを C:\Program Files\NetCOBOL にインストールした場合、次のフォルダに格納されています。

[例]

C:\Program Files\NetCOBOL

2.4　アプリケーションの実行

アプリケーションを実行します。
プログラムの構造が動的プログラム構造である場合、アプリケーションの実行の際にエントリ情報ファイルが必要です。コレクションクラスライブラリで必要となるエントリ情報については、"FJCOENT.INF"ファイルで情報を公開しています。エントリ情報ファイル内に"FJCOENT.INF"ファイルの情報を記述してください。エントリ情報については、"NetCOBOL 使用手引書" の『クラスとメソッドのエントリ情報』を参照してください。
エントリ情報ファイル"FJCOENT.INF"は、コレクションクラスライブラリを C:\Program Files\NetCOBOL にインストールした場合、次のフォルダに格納されています。

[例]

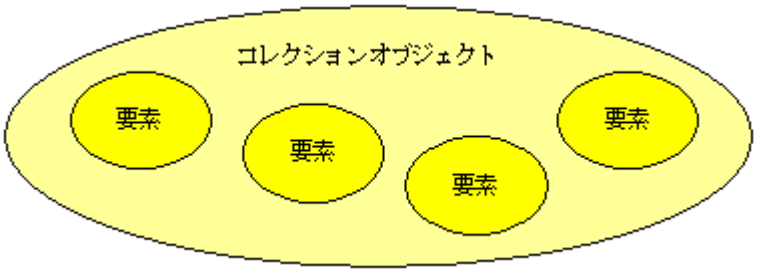
C:\Program Files\NetCOBOL

第3章　コレクションクラス

この章では、コレクションクラスについて説明します。

3.1　コレクションクラスとは

コレクションクラスとは、コレクションクラスを継承しているクラスに対して共通なメソッドを提供するクラスです。
コレクションクラスは、要素(オブジェクト)の集まりを扱います。要素の集まりには順序の概念がなく、同じ要素を複数扱うこともできます。



【図3.1 コレクションオブジェクトの概念図】

コレクションクラスは、FJCOL-COLLECTIONクラスです。

3.2　FJCOL-COLLECTIONクラス

説明

コレクションを操作します。

解説

FJCOL-COLLECTIONクラスはFJBASEクラスを継承しています。

― FJCOL-COLLECTIONクラスは抽象クラス(インタフェース定義だけのクラス)であり、NEWメソッド(ファクトリメソッド)をサポートしていません。

オブジェクトメソッド

メソッド名	機能概要
CLEAR-COLLECTION	現在のコレクションを空にします
CLONE-COLLECTION	現在のコレクションの複製を作成します
CONTAINS-ELEMENT	現在のコレクションに指定した要素が含まれるか検査します
CONTAINS-ALL-ELEMENT	現在のコレクションに指定したコレクションの要素がすべて含まれるか検査します
CREATE-ITERATOR	現在のコレクションに対してイテレータオブジェクトを作成します
GET-SIZE	現在のコレクションの要素数を返します
IS-EMPTY	現在のコレクションが空か検査します

使用例

```
PROGRAM-ID. PROGRAM1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LINKED-LIST
    CLASS FJCOL-COLLECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LINKED-LIST-OBJECT  USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  COL-OBJECT          USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT             USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LINKED-LIST-OBJECT
    INVOKE LINKED-LIST-OBJECT "ADD-FIRST-ELEMENT"
                              USING ELEMENT
                :
    SET COL-OBJECT TO LINKED-LIST-OBJECT                    ┐…[1]
    CALL "PROGRAM2" USING COL-OBJECT                        ┘
                :
```

```
PROGRAM-ID. PROGRAM2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-COLLECTION
    CLASS FJCOL-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ELEMENT          USAGE OBJECT REFERENCE.
01  SIZE-COUNT       PIC S9(9) COMP-5.
                :
LINKAGE SECTION.
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
PROCEDURE DIVISION USING COL-OBJECT.                          …[2]
    INVOKE COL-OBJECT "GET-SIZE" RETURNING SIZE-COUNT
    IF SIZE-COUNT NOT = 0 THEN
      INVOKE COL-OBJECT "CREATE-ITERATOR" RETURNING ITERATOR-OBJECT
      PERFORM SIZE-COUNT TIMES
        INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"
                               RETURNING ELEMENT
                :
      END-PERFORM
    END-IF
                :
```

[例の説明]
プログラムの引数としてFJCOL-COLLECTIONクラスのオブジェクトを受け取ります([2])。この引数のオブジェクト参照値は、FJCOL-COLLECTIONクラスを継承しているクラスのオブジェクト参照値ならどのクラスのオブジェクトでも構いません。例の場合、PROGRAM1でFJCOL-LINKED-LISTクラスのオブジェクトを設定しています([1])。PROGRAM2ではコレクションクラスを継承するクラスのオブジェクトに共通の処理を記述します。

3.2.1　CLEAR-COLLECTIONメソッド

説明

現在のコレクションを空にします。

記述形式

INVOKE *COLLECTION* **"CLEAR-COLLECTION"**

解説

コレクションを空の状態にします。空の状態とは要素数が0の状態をいいます。

使用例

```
PROGRAM-ID. PROGRAM1.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
```

```
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT          USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT
                :
    SET COL-OBJECT TO LIST-OBJECT
    CALL "PROGRAM2" USING COL-OBJECT
                :
```

```
PROGRAM-ID. PROGRAM2.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
                :
LINKAGE SECTION.
01  COL-OBJECT      USAGE OBJECT REFERENCE FJCOL-COLLECTION.
PROCEDURE DIVISION USING COL-OBJECT.
                :
    INVOKE COL-OBJECT "CLEAR-COLLECTION"                    …[1]
                :
```

[例の説明]

[1]

コレクションを空にします。

3.2.2　CLONE-COLLECTIONメソッド

説明

現在のコレクションの複製を作成します。

記述形式

INVOKE *COLLECTION* **"CLONE-COLLECTION"**
RETURNING *COLLECTION-A*

復帰値

COLLECTION-A **[属性: OBJECT REFERENCE CLASS OF SELF]**

複製のコレクションオブジェクトが返却されます。

解説

現在のコレクションに登録されている要素をすべて登録した、新しいコレクションを作成します。当メソッドは、コレクションに登録されている要素自身の複製は作成しません。現在のコレクションが空である場合、複製として空のコレクションを作成します。

使用例

```
PROGRAM-ID. PROGRAM1.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
```

```
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT          USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT
                :
    SET COL-OBJECT TO LIST-OBJECT
    CALL "PROGRAM2" USING COL-OBJECT
                :
```

```
PROGRAM-ID. PROGRAM2.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  COL-NEW-OBJECT  USAGE OBJECT REFERENCE FJCOL-COLLECTION.
                :
LINKAGE SECTION.
01  COL-OBJECT      USAGE OBJECT REFERENCE FJCOL-COLLECTION.
PROCEDURE DIVISION USING COL-OBJECT.
                :
    INVOKE COL-OBJECT "CLONE-COLLECTION"                    …[1]
                      RETURNING COL-NEW-OBJECT
                :
```

[例の説明]

[1]

現在のコレクションの複製を作成します。

3.2.3　CONTAINS-ELEMENTメソッド

説明

現在のコレクションに指定した要素が含まれるか検査します。

記述形式

```
INVOKE COLLECTION "CONTAINS-ELEMENT"
                  USING [BY CONTENT] ELEMENT
                  RETURNING FG
```

引数

ELEMENT **[属性: OBJECT REFERENCE]**

要素(オブジェクト)を指定します。

ただし、BY CONTENTを指定した場合、要素のクラスで型付けしたオブジェクト一意名を指定することもできます。要素のクラスで型付けしたオブジェクト一意名とは、要素のクラスをelement-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  ELEMENT  OBJECT REFERENCE element-class.
```

復帰値

FG **[属性: PIC 1(1) DISPLAY]**

要素がコレクションに含まれる場合は、B"1"が返却されます。
要素がコレクションに含まれない場合は、B"0"が返却されます。

解説

引数で指定した要素がコレクションに含まれるか検査します。
指定された要素のオブジェクト参照値が、現在のコレクションに登録されている要素のオブジェクト参照値のいずれかと一致する場合、コレクションに含まれるとみなします。

使用例

```
PROGRAM-ID. PROGRAM1.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT          USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT
              :
    SET COL-OBJECT TO LIST-OBJECT
    CALL "PROGRAM2" USING COL-OBJECT ELEMENT
              :
```

```
PROGRAM-ID. PROGRAM2.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  CONTAINS-FG     PIC 1(1) DISPLAY.
              :
LINKAGE SECTION.
01  COL-OBJECT      USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  CHK-ELEMENT     USAGE OBJECT REFERENCE.
PROCEDURE DIVISION USING COL-OBJECT CHK-ELEMENT.
              :
    INVOKE COL-OBJECT "CONTAINS-ELEMENT"                    …[1]
           USING CHK-ELEMENT RETURNING CONTAINS-FG
    IF CONTAINS-FG = B"1" THEN                          ┐
              :                                         │…[2]
    END-IF                                              ┘
              :
```

[例の説明]

[1]
現在のコレクションに指定した要素が含まれるか検査します。

[2]
要素が含まれていた場合の処理を記述します。

3.2.4　CONTAINS-ALL-ELEMENTメソッド

説明

現在のコレクションに指定したコレクションの要素がすべて含まれるか検査します。

記述形式

```
INVOKE COLLECTION "CONTAINS-ALL-ELEMENT"
                  USING [BY CONTENT] COLLECTION-A
                  RETURNING FG
```

引数

COLLECTION-A [属性: OBJECT REFERENCE]

コレクションのオブジェクトを指定します。
ただし、BY CONTENTを指定した場合、コレクションのクラスで型付けしたオブジェクト一意名を指定することもできます。コレクションのクラスで型付けしたオブジェクト一意名とは、コレクションのクラスをFJCOL-COLLECTIONとした場合は、次のようなオブジェクト一意名をいいます。

```
01  COLLECTION-A  OBJECT REFERENCE FJCOL-COLLECTION.
```

復帰値

FG [属性: PIC 1(1) DISPLAY]

指定したコレクションの要素がすべて現在のコレクションに含まれる場合は、B"1"が返却されます。
指定されたコレクションの要素の1つでも現在のコレクションに含まれない場合は、B"0"が返却されます。

解説

指定したコレクションが現在のコレクションに含まれるか検査します。
指定されたコレクションの要素がすべて現在のコレクションに含まれる場合、コレクションが含まれるとみなします。

注意

- 指定したコレクションが空である場合、復帰値にはB"1"が返されます。
- 指定したコレクションのオブジェクト参照値がNULLである場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-PARAMETER)が発生します。

使用例

```
PROGRAM-ID. PROGRAM1.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT          USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT
```

```
            :
    SET COL-OBJECT TO LIST-OBJECT
    CALL "PROGRAM2" USING COL-OBJECT
            :
```

```
PROGRAM-ID. PROGRAM2.
            :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  COL-OBJECT-X    USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  CONTAINS-FG     PIC 1(1) DISPLAY.
            :
LINKAGE SECTION.
01  COL-OBJECT      USAGE OBJECT REFERENCE FJCOL-COLLECTION.
PROCEDURE DIVISION USING COL-OBJECT.
            :
    INVOKE COL-OBJECT "CONTAINS-ALL-ELEMENT"                  …[1]
           USING BY CONTENT COL-OBJECT-X
           RETURNING CONTAINS-FG
    IF CONTAINS-FG = B"1" THEN                                ┐
            :                                                 │…[2]
    END-IF                                                    ┘
            :
```

[例の説明]

[1]

現在のコレクションに指定したコレクションの要素がすべて含まれるか検査します。

[2]

コレクションの要素がすべて含まれていた場合の処理を記述します。

3.2.5　CREATE-ITERATORメソッド

説明

現在のコレクションに対してイテレータオブジェクトを作成します。

記述形式

```
INVOKE COLLECTION "CREATE-ITERATOR"
                  RETURNING ITERATOR
```

復帰値

ITERATOR **[属性: OBJECT REFERENCE FJCOL-ITERATOR]**

イテレータクラスのオブジェクトが返却されます。

解説

現在のコレクションの要素を順に操作できるイテレータオブジェクトが作成されます。
イテレータオブジェクトは、イテレータクラスのオブジェクトです。

使用例

```
PROGRAM-ID. PROGRAM1.
```

```
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT          USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT
                :
    SET COL-OBJECT TO LIST-OBJECT
    CALL "PROGRAM2" USING COL-OBJECT
                :
```

```
PROGRAM-ID. PROGRAM2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-COLLECTION
    CLASS FJCOL-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-ITERATOR.
                :
LINKAGE SECTION.
01  COL-OBJECT      USAGE OBJECT REFERENCE FJCOL-COLLECTION.
PROCEDURE DIVISION USING COL-OBJECT.
                :
    INVOKE COL-OBJECT "CREATE-ITERATOR"           …[1]
           RETURNING ITERATOR-OBJECT
                :
```

[例の説明]

[1]

イテレータオブジェクトを作成します。

3.2.6　GET-SIZEメソッド

説明

現在のコレクションの要素数を返します。

記述形式

```
INVOKE COLLECTION "GET-SIZE"
                  RETURNING COUNT
```

復帰値

COUNT **[属性: PIC S9(9) COMP-5]**

要素数が返却されます。

解説

現在のコレクションの要素数が返されます。

使用例

```
PROGRAM-ID. PROGRAM1.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT          USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT
                :
    SET COL-OBJECT TO LIST-OBJECT
    CALL "PROGRAM2" USING COL-OBJECT
                :
```

```
PROGRAM-ID. PROGRAM2.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ELEMENT-COUNT   PIC S9(9) COMP-5.
                :
LINKAGE SECTION.
01  COL-OBJECT      USAGE OBJECT REFERENCE FJCOL-COLLECTION.
PROCEDURE DIVISION USING COL-OBJECT.
                :
    INVOKE COL-OBJECT "GET-SIZE" RETURNING ELEMENT-COUNT      …[1]
                :
```

[例の説明]

[1]

コレクションの要素数を獲得します。

3.2.7 IS-EMPTYメソッド

説明

現在のコレクションが空か検査します。

記述形式

```
INVOKE COLLECTION "IS-EMPTY"
                  RETURNING FG
```

復帰値

FG **[属性: PIC 1(1) DISPLAY]**

コレクションが空である場合は、B"1"が返却されます。

コレクションが空でない場合は、B"0"が返却されます。

解説

現在のコレクションが空であるか検査します。空とは要素数が0の状態をいいます。

使用例

```
PROGRAM-ID. PROGRAM1.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  COL-OBJECT       USAGE OBJECT REFERENCE FJCOL-COLLECTION.
01  ELEMENT          USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT
              :
    SET COL-OBJECT TO LIST-OBJECT
    CALL "PROGRAM2" USING COL-OBJECT
              :
```

```
PROGRAM-ID. PROGRAM2.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  EMPTY-FG        PIC 1(1) DISPLAY.
              :
LINKAGE SECTION.
01  COL-OBJECT      USAGE OBJECT REFERENCE FJCOL-COLLECTION.
PROCEDURE DIVISION USING COL-OBJECT.
              :
    INVOKE COL-OBJECT "IS-EMPTY" RETURNING EMPTY-FG          …[1]
    IF EMPTY-FG = B"1" THEN                                  ┐
              :                                              │…[2]
    END-IF                                                   ┘
              :
```

[例の説明]

[1]

コレクションが空か検査します。

[2]

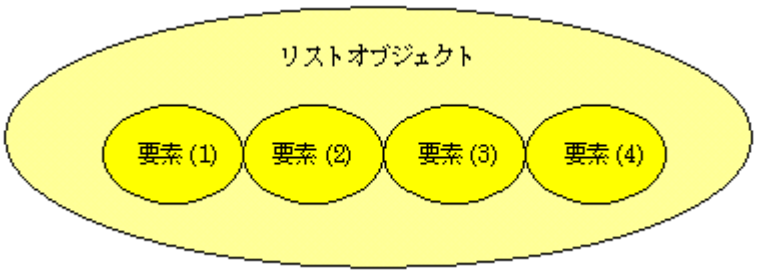
コレクションが空であった場合の処理を記述します。

第4章　リストクラス

この章では、リストクラスについて説明します。

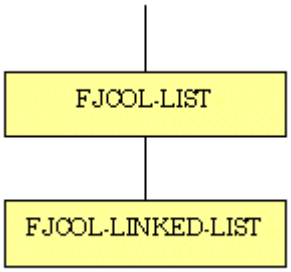
4.1　リストクラスとは

リストクラスは、要素(オブジェクト)の集まりを順序付けて扱います。リストクラスは、同じ要素を複数扱うことができます。



【図4.1-1 リストオブジェクトの概念図】

リストクラスには、FJCOL-LISTクラスとFJCOL-LINKED-LISTクラスがあります。



【図4.1-2 リストクラスのクラス階層図】

4.2　FJCOL-LISTクラス

説明

リストを操作します。

解説

FJCOL-LISTクラスはFJCOL-COLLECTIONクラスを継承しています。
リストクラスはすべてこのクラスを継承します。現在は、当クラスから継承しているクラスはFJCOL-LINKED-LISTクラスしか提供していません。しかし、将来当クラスを継承した別のリストクラスが提供された場合、当クラスを使用してリストクラスに共通の処理を記述することができます。

― FJCOL-LISTクラスは抽象クラス(インタフェース定義だけのクラス)であり、NEWメソッド(ファクトリメソッド)をサポートしていません。

オブジェクトメソッド

メソッド名	機能概要
CREATE-LIST-ITERATOR	現在のリストに対してリストイテレータオブジェクトを作成します
EQUALS-LIST-COLLECTION	現在のリストと指定したリストが同等か検査します

使用例

```
PROGRAM-ID. PROGRAM1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LINKED-LIST
    CLASS FJCOL-LIST.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LINKED-LIST-OBJECT  USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  LIST-OBJECT         USAGE OBJECT REFERENCE FJCOL-LIST.
01  ELEMENT             USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LINKED-LIST-OBJECT
    INVOKE LINKED-LIST-OBJECT "ADD-FIRST-ELEMENT"
                              USING ELEMENT
                :
    SET LIST-OBJECT TO LINKED-LIST-OBJECT                  ┐…[1]
    CALL "PROGRAM2" USING LIST-OBJECT                      ┘
                :
```

```
PROGRAM-ID. PROGRAM2.
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LIST
    CLASS FJCOL-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ELEMENT          USAGE OBJECT REFERENCE.
01  SIZE-COUNT       PIC S9(9) COMP-5.
              :
LINKAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LIST.
PROCEDURE DIVISION USING LIST-OBJECT.                          …[2]
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING SIZE-COUNT
    IF SIZE-COUNT NOT = 0 THEN
      INVOKE LIST-OBJECT "CREATE-ITERATOR"
                         RETURNING ITERATOR-OBJECT
      PERFORM SIZE-COUNT TIMES
        INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"
                               RETURNING ELEMENT
              :
      END-PERFORM
    END-IF
              :
```

[例の説明]
プログラムの引数としてFJCOL-LISTクラスのオブジェクトを受け取ります([2])。この引数のオブジェクト参照値は、FJCOL-LISTクラスを継承しているリストクラスのオブジェクト参照値ならどのクラスのオブジェクトでも構いません。例の場合、PROGRAM1でFJCOL-LINKED-LISTクラスのオブジェクトを設定しています([1])。PROGRAM2ではリストクラスのオブジェクトに共通の処理を記述します。

4.2.1　CREATE-LIST-ITERATORメソッド

説明

現在のリストに対してリストイテレータオブジェクトを作成します。

記述形式

```
INVOKE LIST "CREATE-LIST-ITERATOR"
            USING INDEX
            RETURNING LIST-ITERATOR
```

引数

INDEX [属性: S9(9) COMP-5]
　指標の値を指定します。
　数字定数で指定することもできます。

復帰値

LIST-ITERATOR [属性: OBJECT REFERENCE FJCOL-LIST-ITERATOR]
　リストイテレータオブジェクトが返却されます。

解説

現在のリストの要素を双方向に操作できるリストイテレータオブジェクトを作成します。
リストイテレータオブジェクトは、リストイテレータクラスのオブジェクトです。
作成された直後のリストイテレータオブジェクトの指標の値を引数で指定します。

― 指定する指標の値は、1から要素数+1までの範囲でなければなりません。範囲外の値が指定された場合は、コレクション例外クラスの例外オブジェクト(例外種別：FJCOL-SCE-INDEX)が発生します。

使用例

```
PROGRAM-ID. PROGRAM1.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  L-LIST-OBJECT   USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING L-LIST-OBJECT    …[1]
    INVOKE L-LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1    …[2]
    INVOKE L-LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2    …[2]
                :
    SET LIST-OBJECT  TO L-LIST-OBJECT
    CALL "PROGRAM2" USING LIST-OBJECT
                :
```

```
PROGRAM-ID. PROGRAM2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LIST
    CLASS FJCOL-LIST-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  INDEX-VALUE     PIC S9(9) COMP-5 VALUE 1.
                :
LINKAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LIST.
PROCEDURE DIVISION USING LIST-OBJECT.
                :
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                 …[3]
           USING INDEX-VALUE RETURNING ITERATOR-OBJECT
                :
```

[例の説明]

[1]

リストオブジェクトを作成します。

[2]

要素を2つ追加します。

[3]

リストイテレータオブジェクトを指標の値=1で作成します。
このリストイテレータオブジェクトで操作できる要素は、PROGRAM1の[2]で追加した2つの要素です。(リストイテレータオブジェクトに対して追加処理を行うなどして、操作する要素を増やすことは可能です。)

4.2.2　EQUALS-LIST-COLLECTIONメソッド

説明

現在のリストと指定したリストが同等か検査します。

記述形式

```
INVOKE LIST "EQUALS-LIST-COLLECTION"
            USING [BY CONTENT] LIST-A
            RETURNING FG
```

引数

***LIST-A* [属性: OBJECT REFERENCE]**

リストクラスのオブジェクトを指定します。
ただし、BY CONTENTを指定した場合、リストクラスで型付けしたオブジェクト一意名を指定することもできます。リストクラスで型付けしたオブジェクト一意名とは、リストクラスをFJCOL-LISTとした場合は、次のようなオブジェクト一意名をいいます。

```
01  LIST-A  OBJECT REFERENCE FJCOL-LIST.
```

復帰値

***FG* [属性: PIC 1(1) DISPLAY]**

現在のリストと指定したリストが同等である場合、B"1"が返却されます。
現在のリストと指定したリストが同等でない場合、B"0"が返却されます。

解説

現在のリストと指定したリストが同等か検査します。リストが同等とは次のような状態を言います。

- 現在のリストと指定したリストの要素数が同じである。かつ
- それぞれのリストの先頭から要素を取り出し、その要素のオブジェクト参照値を比較した場合、すべて同一の値である。(登録されている要素の数、オブジェクト参照値が同じ場合でも、登録されている順番が異なる場合は、同等とはみなしません。)

- 現在のリストと指定したリストが共に空である場合は、B"1"が返却されます。
- 指定したリストのオブジェクト参照値がNULLである場合は、B"0"が返却されます。

使用例

```
PROGRAM-ID. PROGRAM1.
```

```
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  L-LIST-OBJECT1  USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  L-LIST-OBJECT2  USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  LIST-OBJECT1    USAGE OBJECT REFERENCE FJCOL-LIST.
01  LIST-OBJECT2    USAGE OBJECT REFERENCE FJCOL-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING L-LIST-OBJECT1
    INVOKE L-LIST-OBJECT1 "ADD-FIRST-ELEMENT" USING ELEMENT1
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING L-LIST-OBJECT2
    INVOKE L-LIST-OBJECT2 "ADD-FIRST-ELEMENT" USING ELEMENT2
              :
    SET LIST-OBJECT1 TO L-LIST-OBJECT1
    SET LIST-OBJECT2 TO L-LIST-OBJECT2
    CALL "PROGRAM2" USING LIST-OBJECT1 LIST-OBJECT2
              :
```

```
PROGRAM-ID. PROGRAM2.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  COMPARE-FG      PIC 1(1) DISPLAY.
              :
LINKAGE SECTION.
01  LIST-OBJECT1    USAGE OBJECT REFERENCE FJCOL-LIST.
01  LIST-OBJECT2    USAGE OBJECT REFERENCE FJCOL-LIST.
PROCEDURE DIVISION USING LIST-OBJECT1 LIST-OBJECT2.
              :
    INVOKE LIST-OBJECT1 "EQUALS-LIST-COLLECTION"              …[1]
           USING BY CONTENT LIST-OBJECT2 RETURNING COMPARE-FG
    IF COMPARE-FG = B"1" THEN                                 ┐
              :                                               │…[2]
    END-IF                                                    ┘
              :
```

[例の説明]

[1]

リストが同等かどうか検査します。

[2]

リストが同等だったときの処理を記述します。

4.3　FJCOL-LINKED-LISTクラス

説明

順序付けされたリストを作成し、操作します。

解説

FJCOL-LINKED-LISTクラスは、FJCOL-LISTクラスを継承しています。
FJCOL-LINKED-LISTクラスは、リストの先頭、末尾に対して操作を行うことができます。

ファクトリメソッド

メソッド名	機能概要
NEW	リストクラスのオブジェクトを作成します

オブジェクトメソッド

メソッド名	機能概要
ADD-FIRST-ELEMENT	現在のリストの先頭に要素を追加します
ADD-LAST-ELEMENT	現在のリストの末尾に要素を追加します
GET-FIRST-ELEMENT	現在のリストの先頭の要素を返します
GET-LAST-ELEMENT	現在のリストの末尾の要素を返します
REMOVE-FIRST-ELEMENT	現在のリストの先頭の要素を削除します
REMOVE-LAST-ELEMENT	現在のリストの末尾の要素を削除します

使用例

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LINKED-LIST.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LINKED-LIST-OBJECT  USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ELEMENT1            USAGE OBJECT REFERENCE.
01  ELEMENT2            USAGE OBJECT REFERENCE.
01  ELEMENT3            USAGE OBJECT REFERENCE.
01  F-ELEMENT           USAGE OBJECT REFERENCE.
01  L-ELEMENT           USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LINKED-LIST-OBJECT
    INVOKE LINKED-LIST-OBJECT "ADD-FIRST-ELEMENT"            …[1]
                                  USING ELEMENT1
    INVOKE LINKED-LIST-OBJECT "ADD-FIRST-ELEMENT"            …[2]
                                  USING ELEMENT2
    INVOKE LINKED-LIST-OBJECT "ADD-FIRST-ELEMENT"            …[3]
                                  USING ELEMENT3
```

```
          :
INVOKE LINKED-LIST-OBJECT "GET-FIRST-ELEMENT"          …[4]
                          RETURNING F-ELEMENT
INVOKE LINKED-LIST-OBJECT "GET-LAST-ELEMENT"           …[5]
                          RETURNING L-ELEMENT
          :
```

[例の説明]

[1]

リストに要素(ELEMENT1)を登録します。

[2]

リストに要素(ELEMENT2)を登録します。

[3]

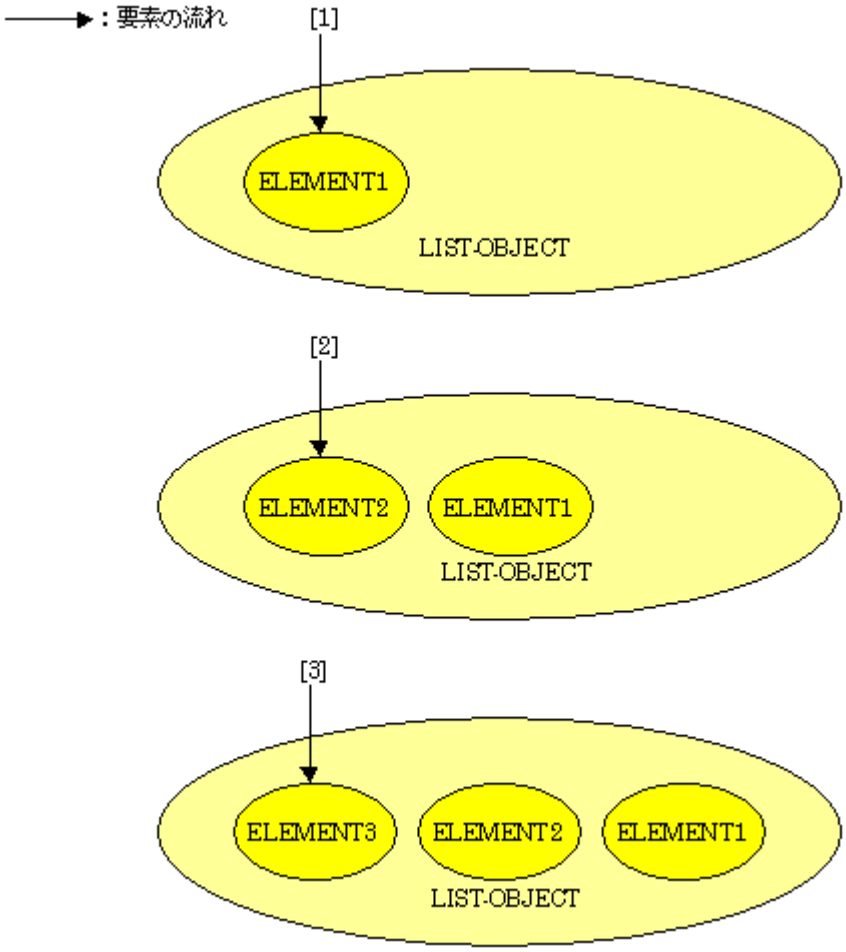
リストに要素(ELEMENT3)を登録します。

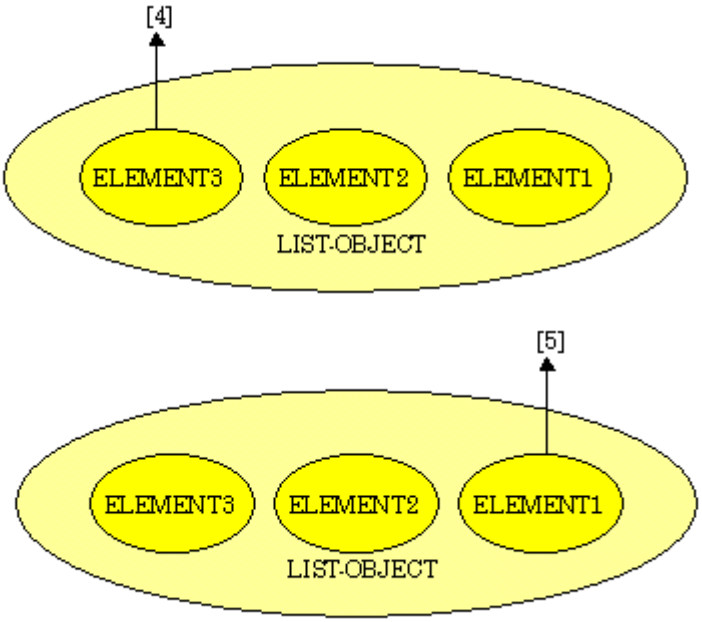
[4]

リストに登録されている先頭の要素を獲得します。
ここでは、[3]で登録した要素(ELEMENT3)が獲得されます。

[5]

リストに登録されている末尾の要素を獲得します。
ここでは、[1]で登録した要素(ELEMENT1)が獲得されます。





【図4.3 使用例の概念図 】

4.3.1　　NEWメソッド

説明

FJCOL-LINKED-LISTクラスのリストオブジェクトを作成します。

記述形式

```
INVOKE FJCOL-LINKED-LIST  "NEW"
             RETURNING LINKED-LIST
```

復帰値

LINKED-LIST [属性: OBJECT REFERENCE SELF]

FJCOL-LINKED-LISTクラスのオブジェクトが返却されます。

解説

FJCOL-LINKED-LISTクラスのオブジェクトが作成されます。

使用例

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LINKED-LIST.                                  …[1]
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.     …[2]
              :
PROCEDURE DIVISION.
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[3]
              :
```

[例の説明]

[1]

リポジトリ段落にFJCOL-LINKED-LISTクラスを宣言します。

[2]

リストオブジェクトのオブジェクト参照値を保持するためのデータ項目を宣言します。

[3]

リストオブジェクトを作成します。

4.3.2　ADD-FIRST-ELEMENTメソッド

説明

現在のリストの先頭に指定した要素を追加します。

記述形式

```
INVOKE LINKED-LIST "ADD-FIRST-ELEMENT"
            USING [BY CONTENT] ELEMENT
```

引数

ELEMENT **[属性: OBJECT REFERENCE]**

要素(オブジェクト)を指定します。

ただし、BY CONTENTを指定した場合、要素のクラスで型付けしたオブジェクト一意名を指定することもできます。要素のクラスで型付けしたオブジェクト一意名とは、要素のクラスをelement-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  ELEMENT  OBJECT REFERENCE element-class.
```

解説

指定した要素をリストの先頭に追加します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT1     …[2]
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT2     …[3]
              :
```

[例の説明]

[1]

リストオブジェクトを作成します。

[2]

先頭にオブジェクトを追加します。

[3]

先頭にオブジェクトを追加します。

この一連の操作により、先頭に[3]で追加した要素(ELEMENT2)、2番目に[2]で追加した要素(ELEMENT1)のリストになります。

4.3.3　ADD-LAST-ELEMENTメソッド

説明

現在のリストの末尾に指定した要素を追加します。

記述形式

```
INVOKE LINKED-LIST "ADD-LAST-ELEMENT"
            USING [BY CONTENT] ELEMENT
```

引数

ELEMENT [属性: OBJECT REFERENCE]

要素(オブジェクト)を指定します。
ただし、BY CONTENTを指定した場合、要素のクラスで型付けしたオブジェクト一意名を指定することもできます。要素のクラスで型付けしたオブジェクト一意名とは、要素のクラスをelement-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  ELEMENT  OBJECT REFERENCE element-class.
```

解説

指定した要素をリストの末尾に追加します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1      …[2]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2      …[3]
              :
```

[例の説明]

[1]
リストオブジェクトを作成します。

[2]
末尾にオブジェクトを追加します。

[3]
末尾にオブジェクトを追加します。
この一連の操作により、先頭に[2]で追加した要素(ELEMENT1)、2番目に[3]で追加した要素(ELEMENT2)のリストになります。

4.3.4　GET-FIRST-ELEMENTメソッド

説明

現在のリストの先頭の要素を返します。

記述形式

```
INVOKE LINKED-LIST "GET-FIRST-ELEMENT"
                   RETURNING ELEMENT
```

復帰値

ELEMENT [属性: OBJECT REFERENCE]

要素のオブジェクトが返却されます。

解説

現在のリストの先頭の要素が返されます。

― 現在のリストが空である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-NOELEMENT)が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1   ┐ …[2]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2   ┘
    INVOKE LIST-OBJECT "GET-FIRST-ELEMENT"                    …[3]
                       RETURNING ELEMENT-X
              :
```

[例の説明]

[1]

リストオブジェクトを作成します。

[2]

要素を２つ追加します。

[3]

先頭の要素を獲得します。

データ項目ELEMENT-Xには、ELEMENT1のオブジェクト参照値と同じ値が返されます。

4.3.5　GET-LAST-ELEMENTメソッド

説明

現在のリストの末尾の要素を返します。

記述形式

```
INVOKE LINKED-LIST "GET-LAST-ELEMENT"
                   RETURNING ELEMENT
```

復帰値

ELEMENT [属性: OBJECT REFERENCE]

要素のオブジェクトが返却されます。

解説

現在のリストの末尾の要素が返されます。

― 現在のリストが空である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-NOELEMENT)が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1    ┐ …[2]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2    ┘
    INVOKE LIST-OBJECT "GET-LAST-ELEMENT" RETURNING ELEMENT-X …[3]
              :
```

[例の説明]

[1]

リストオブジェクトを作成します。

[2]

要素を２つ追加します。

[3]

末尾の要素を獲得します。

データ項目ELEMENT-Xには、ELEMENT2のオブジェクト参照値と同じ値が返されます。

4.3.6　REMOVE-FIRST-ELEMENTメソッド

説明

現在のリストの先頭の要素を削除します。

記述形式

```
INVOKE LINKED-LIST "REMOVE-FIRST-ELEMENT"
                   RETURNING ELEMENT
```

復帰値

ELEMENT [属性: OBJECT REFERENCE]
　要素のオブジェクトが返却されます。

解説

リストの先頭の要素を削除し、削除した要素(オブジェクト)を返します。

― 現在のリストが空である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-NOELEMENT)が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1      …[2]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2      …[3]
    INVOKE LIST-OBJECT "REMOVE-FIRST-ELEMENT"                 …[4]
                       RETURNING ELEMENT-X
                :
```

[例の説明]

[1]
　リストオブジェクトを作成します。

[2]
　末尾に要素を追加します。

[3]
　末尾に要素を追加します。

[4]
　先頭の要素を削除します。
　リストから削除される要素は、[2]で追加した要素(ELEMENT1)であり、その要素のオブジェクト参照値が返されます。(ELEMENT-X = ELEMENT1)

4.3.7　REMOVE-LAST-ELEMENTメソッド

説明

現在のリストの末尾の要素を削除します。

記述形式

```
INVOKE LINKED-LIST "REMOVE-LAST-ELEMENT"
                   RETURNING ELEMENT
```

復帰値

ELEMENT [属性: OBJECT REFERENCE]

要素のオブジェクトが返却されます。

解説

リストの末尾の要素を削除し、削除した要素(オブジェクト)を返します。

― 現在のリストが空である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-NOELEMENT)が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1      …[2]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2      …[3]
    INVOKE LIST-OBJECT "REMOVE-LAST-ELEMENT"                  …[4]
                       RETURNING ELEMENT-X
                :
```

[例の説明]

[1]

リストオブジェクトを作成します。

[2]

末尾に要素を追加します。

[3]

末尾に要素を追加します。

[4]

末尾の要素を削除します。

リストから削除される要素は、[3]で追加した要素(ELEMENT2)であり、その要素のオブジェクト参照値が返されます。(ELEMENT-X = ELEMENT2)

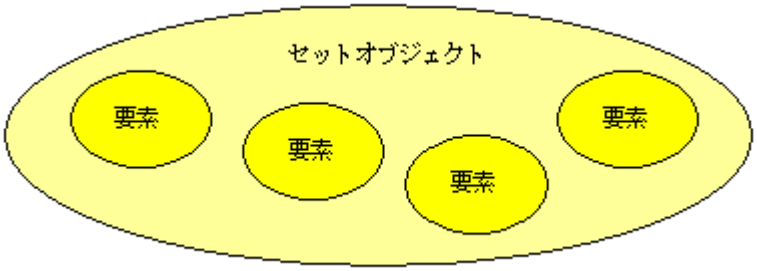
第5章　セットクラス

この章では、セットクラスについて説明します。

5.1　セットクラスとは

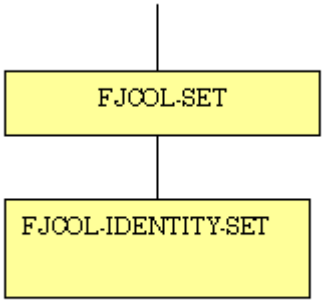
セットクラスは、要素(オブジェクト)の集まりを操作するクラスです。セットは重複した要素を持ちません。

要素(オブジェクト)が一致するかどうかの判断は、そのセットクラスを作成したクラスに依存します。例えばFJCOL-IDENTITY-SETクラスであれば、要素(オブジェクト)が一致するかどうかはオブジェクト参照値が一致するかどうかで判断します。



【図5.1-1 セットオブジェクトの概念図】

セットクラスには、FJCOL-SETクラスとFJCOL-IDENTITY-SETクラスがあります。



【図5.1-2 セットクラスのクラス階層図】

5.2　FJCOL-SETクラス

説明

セットを操作します。

解説

FJCOL-SETクラスは、FJCOL-COLLECTIONクラスを継承しています。
セットクラスはすべてこのクラスを継承します。現在は、当クラスから継承しているクラスはFJCOL-IDENTITY-SETクラスしか提供していません。しかし、将来当クラスを継承した別のセットクラスが提供された場合、当クラスを使用してセットクラスに共通の処理を記述することができます。

- FJCOL-SETクラスは抽象クラス(インタフェース定義だけのクラス)であり、NEWメソッド(ファクトリメソッド)をサポートしていません。
- 要素(オブジェクト)が一致するかどうかの判断は、そのセットクラスを作成したクラスに依存します。

オブジェクトメソッド

メソッド名	機能概要
ADD-ELEMENT	現在のセットに要素を追加します
ADD-ALL-ELEMENT	現在のセットに指定したコレクションが持つ要素を追加します
EQUALS-SET-COLLECTION	現在のセットと指定したセットが同等か検査します
REMOVE-ELEMENT	現在のセットから要素を削除します
REMOVE-ALL-ELEMENT	現在のセットから指定したコレクションが持つ要素を削除します

使用例

```
PROGRAM-ID. PROGRAM1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-IDENTITY-SET
    CLASS FJCOL-SET.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  IDENTITY-SET-OBJECT   USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  SET-OBJECT          USAGE OBJECT REFERENCE FJCOL-SET.
01  VALUE-OBJECT        USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING IDENTITY-SET-OBJECT
    INVOKE IDENTITY-SET-OBJECT "ADD-ELEMENT" USING VALUE-OBJECT
                :
    SET SET-OBJECT TO IDENTITY-SET-OBJECT                    ┐…[1]
```

```
    CALL "PROGRAM2" USING SET-OBJECT                              ┘
              :
```

```
PROGRAM-ID. PROGRAM2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-SET
    CLASS FJCOL-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  VALUE-OBJECT     USAGE OBJECT REFERENCE.
01  SIZE-COUNT       PIC S9(9) COMP-5.
              :
LINKAGE SECTION.
01  SET-OBJECT       USAGE OBJECT REFERENCE FJCOL-SET.
PROCEDURE DIVISION USING SET-OBJECT.                              …[2]
    INVOKE SET-OBJECT "GET-SIZE" RETURNING SIZE-COUNT
    IF SIZE-COUNT NOT = 0 THEN
      INVOKE SET-OBJECT "CREATE-ITERATOR"
                        RETURNING ITERATOR-OBJECT
      PERFORM SIZE-COUNT TIMES
        INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"
                               RETURNING VALUE-OBJECT
              :
      END-PERFORM
    END-IF
              :
```

[例の説明]
プログラムの引数としてFJCOL-SETクラスのオブジェクトを受け取ります([2])。この引数のオブジェクト参照値は、FJCOL-SETクラスを継承しているセットクラスのオブジェクト参照値ならどのクラスのオブジェクトでも構いません。例の場合、PROGRAM1でFJCOL-IDENTITY-SETクラスのオブジェクトを設定しています([1])。PROGRAM2ではセットクラスのオブジェクトに共通の処理を記述します。

5.2.1　ADD-ELEMENTメソッド

説明

現在のセットに、指定したオブジェクトを要素として追加します。

記述形式

```
INVOKE SET "ADD-ELEMENT"
           USING [BY CONTENT] ELEMENT
           RETURNING FG
```

引数

ELEMENT [属性: OBJECT REFERENCE]
追加する要素(オブジェクト)を指定します。
ただし、BY CONTENTを指定した場合、要素のクラスで型付けしたオブジェクト一意

名を指定することもできます。要素のクラスで型付けしたオブジェクト一意名とは、要素のクラスをelement-classとした場合は、次のようなオブジェクト一意名をいいます。

01　*ELEMENT*　OBJECT REFERENCE element-class.

復帰値

***FG* [属性: PIC 1(1) DISPLAY]**

現在のセットに指定の要素が存在していなかった場合は、B"1"が返却されます。
現在のセットに指定の要素が存在していた場合は、B"0"が返却されます。

解説

現在のセットに指定したオブジェクトが存在しない場合、要素として追加します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  IDENTITY-SET-OBJECT  USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  SET-OBJECT          USAGE OBJECT REFERENCE FJCOL-SET.
01  ELEMENT1            USAGE OBJECT REFERENCE.
01  ELEMENT2            USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING IDENTITY-SET-OBJECT     …[1]
    INVOKE IDENTITY-SET-OBJECT "ADD-ELEMENT" USING ELEMENT1           …[2]
    INVOKE IDENTITY-SET-OBJECT "ADD-ELEMENT" USING ELEMENT2           …[3]
                :
```

[例の説明]

[1]
セットオブジェクトを作成します。

[2]
要素をセットオブジェクトに追加します。

[3]
要素をセットオブジェクトに追加します。

5.2.2　ADD-ALL-ELEMENTメソッド

説明

現在のセットに、指定したコレクションが持つ要素を追加します。

記述形式

```
INVOKE SET "ADD-ALL-ELEMENT"
           USING [BY CONTENT] COLLECTION
           RETURNING FG
```

引数

***COLLECTION* [属性: OBJECT REFERENCE]**

現在のセットに追加したい要素を持つコレクションを指定します。
ただし、BY CONTENTを指定した場合、クラスで型付けしたオブジェクト一意名を指定することもできます。クラスで型付けしたオブジェクト一意名とは、次のような

オブジェクト一意名をいいます。

```
01  IDENTITY-SET-A  OBJECT REFERENCE FJCOL-IDENTITY-SET.
```

復帰値

FG [属性: PIC 1(1) DISPLAY]

現在のセットが持つ要素に変更がある場合は、B"1"が返却されます。
現在のセットが持つ要素に変更がない場合は、B"0"が返却されます。

解説

指定したコレクションオブジェクトが持つすべての要素の中に、現在のセット中に存在しない要素がある場合、その存在しない要素のすべてを現在のセットに追加します。

- 指定したコレクションのオブジェクト参照値がNULLである場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-PARAMETER)が発生します。
- コレクションオブジェクト以外を引数*COLLECTION*に指定した場合の動作は保証されません。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SET-OBJECTA USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  SET-OBJECTB USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  ELEMENT1    USAGE OBJECT REFERENCE.
01  ELEMENT2    USAGE OBJECT REFERENCE.
01  ELEMENT3    USAGE OBJECT REFERENCE.
               :
PROCEDURE DIVISION.
               :
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECTA      …[1]
    INVOKE SET-OBJECTA "ADD-ELEMENT" USING ELEMENT1          ┐ …[2]
    INVOKE SET-OBJECTA "ADD-ELEMENT" USING ELEMENT2          ┘
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECTB      …[3]
    INVOKE SET-OBJECTB "ADD-ELEMENT" USING ELEMENT2          ┐ …[4]
    INVOKE SET-OBJECTB "ADD-ELEMENT" USING ELEMENT3          ┘
               :
    INVOKE SET-OBJECTA "ADD-ALL-ELEMENT"                       …[5]
                       USING BY CONTENT SET-OBJECTB
               :
```

[例の説明]

[1]

マップオブジェクト(SET-OBJECTA) を作成します。

[2]

SET-OBJECTA に要素を２つ(ELEMENT1,ELEMENT2)追加します。

[3]

マップオブジェクト(SET-OBJECTB) を作成します。

[4]

SET-OBJECTB に要素を２つ(ELEMENT2,ELEMENT3)追加します。

[5]

SET-OBJECTA にSET-OBJECTB のエントリをすべて追加します。ELEMENT2は重複しているため、SET-OBJECTA の要素数は３になります。

5.2.3　EQUALS-SET-COLLECTIONメソッド

説明

現在のセットと指定したセットが同等か検査します。

記述形式

```
INVOKE SET "EQUALS-SET-COLLECTION"
           USING [BY CONTENT] SET-A
           RETURNING FG
```

引数

SET-A [属性: OBJECT REFERENCE]

セットクラスのオブジェクトを指定します。
ただし、BY CONTENTを指定した場合、セットクラスで型付けしたオブジェクト一意名を指定することもできます。セットクラスで型付けしたオブジェクト一意名とは、セットクラスをFJCOL-SETとした場合は、次のようなオブジェクト一意名をいいます。

```
01  SET-A  OBJECT REFERENCE FJCOL-SET.
```

復帰値

FG [属性: PIC 1(1) DISPLAY]

現在のセットと指定したセットが同等である場合、B"1"が返却されます。
現在のセットと指定したセットが同等でない場合、B"0"が返却されます。

解説

現在のリストと指定したリストが同等か検査します。リストが同等とは次のような状態を言います。

- 現在のセットと指定したセットの要素数が同じである。かつ、
- 現在のセットが持つすべての要素と指定したセットが持つすべての要素がすべて同一である。

- 現在のセットと指定したセットが共に空である場合は、B"1"が返却されます。
- 指定したセットのオブジェクト参照値がNULLである場合は、B"0"が返却されます。
- 要素(オブジェクト)が一致するかどうかの判断は、そのセットクラスを作成したクラスに依存します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SET-OBJECT1 USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  SET-OBJECT2 USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  COMPARE-FG        PIC 1(1) DISPLAY.
                :
PROCEDURE DIVISION.
                :
```

```
INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECT1
          :
INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECT2
          :
INVOKE SET-OBJECT1 "EQUALS-SET-COLLECTION"                ...[1]
                   USING BY CONTENT SET-OBJECT2
                   RETURNING COMPARE-FG
IF COMPARE-FG = B"1" THEN                                 ┐
          :                                               |...[2]
END-IF                                                    ┘
          :
```

[例の説明]

[1]

セットが同等か検査します。

[2]

セットが同等だったときの処理を記述します。

5.2.4　REMOVE-ELEMENTメソッド

説明

現在のセットから指定の要素を削除します。

記述形式

```
INVOKE SET "REMOVE-ELEMENT"
           USING [BY CONTENT] ELEMENT
           RETURNING FG
```

引数

ELEMENT **[属性: OBJECT REFERENCE]**

削除する要素(オブジェクト)を指定します。
ただし、BY CONTENTを指定した場合、要素のクラスで型付けしたオブジェクト一意名を指定することもできます。要素のクラスで型付けしたオブジェクト一意名とは、要素のクラスをelement-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  ELEMENT  OBJECT REFERENCE element-class.
```

復帰値

FG **[属性: PIC 1(1) DISPLAY]**

現在のセットが指定の要素を持っていたのなら、B"1"が返却されます。
現在のセットが指定の要素を持っていなかったのなら、B"0"が返却されます。

解説

現在のセットから指定の要素を削除します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SET-OBJECT  USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  ELEMENT1    USAGE OBJECT REFERENCE.
01  ELEMENT-X   USAGE OBJECT REFERENCE.
```

```
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECT    …[1]
    INVOKE SET-OBJECT "ADD-ELEMENT" USING ELEMENT1          …[2]
                :
    INVOKE SET-OBJECT "REMOVE-ELEMENT" USING ELEMENT-X      …[3]
                :
```

[例の説明]

[1]
セットオブジェクトを作成します。

[2]
要素を追加します。

[3]
ELEMENT-Xで示される要素がセットに含まれる場合、その要素を削除します。

5.2.5　REMOVE-ALL-ELEMENTメソッド

説明

現在のセットから、指定したコレクションが持つ要素をすべて削除します。

記述形式

```
INVOKE SET "REMOVE-ALL-ELEMENT"
            USING [BY CONTENT] COLLECTION
            RETURNING FG
```

引数

COLLECTION **[属性: OBJECT REFERENCE]**

現在のセットから削除したい要素を持つコレクションを指定します。
ただし、BY CONTENTを指定した場合、クラスで型付けしたオブジェクト一意名を指定することもできます。クラスで型付けしたオブジェクト一意名とは、次のようなオブジェクト一意名をいいます。

```
01  IDENTITY-SET-A  OBJECT REFERENCE FJCOL-IDENTITY-SET.
```

復帰値

FG **[属性: PIC 1(1) DISPLAY]**

現在のセットが持つ要素に変更がある場合は、B"1"が返却されます。
現在のセットが持つ要素に変更がない場合は、B"0"が返却されます。

解説

現在のセットから指定したコレクションが持つ要素をすべて削除します。

- 指定したコレクションのオブジェクト参照値がNULLである場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-PARAMETER)が発生します。
- コレクションオブジェクト以外を引数*COLLECTION*に指定した場合の動作は保証されません。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SET-OBJECTA USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  SET-OBJECTB USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  ELEMENT1    USAGE OBJECT REFERENCE.
01  ELEMENT2    USAGE OBJECT REFERENCE.
01  ELEMENT3    USAGE OBJECT REFERENCE.
             :
PROCEDURE DIVISION.
             :
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECTA      …[1]
    INVOKE SET-OBJECTA "ADD-ELEMENT" USING ELEMENT1          ┐ …[2]
    INVOKE SET-OBJECTA "ADD-ELEMENT" USING ELEMENT2          ┘
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECTB      …[3]
    INVOKE SET-OBJECTB "ADD-ELEMENT" USING ELEMENT2          ┐ …[4]
    INVOKE SET-OBJECTB "ADD-ELEMENT" USING ELEMENT3          ┘
             :
    INVOKE SET-OBJECTA "REMOVE-ALL-ELEMENT"                    …[5]
                       USING BY CONTENT SET-OBJECTB
             :
```

[例の説明]

[1]

マップオブジェクト(SET-OBJECTA) を作成します。

[2]

SET-OBJECTA に要素を２つ(ELEMENT1,ELEMENT2)追加します。

[3]

マップオブジェクト(SET-OBJECTB) を作成します。

[4]

SET-OBJECTB に要素を２つ(ELEMENT2,ELEMENT3)追加します。

[5]

SET-OBJECTA からSET-OBJECTB に含まれる要素をすべて削除します。SET-OBJECTA からはELEMENT2が削除され、ELEMENT1だけが残ります。

5.3　FJCOL-IDENTITY-SETクラス

説明

FJCOL-SETクラスを継承したセットクラスです。

解説

FJCOL-IDENTITY-SETクラスは、FJCOL-SETクラスを継承しています。
当クラスでは、要素(オブジェクト)が一致するかどうかをオブジェクト参照値が一致するかどうかで判断します。

ファクトリメソッド

メソッド名	機能概要
NEW	セットクラスのオブジェクトを作成します

使用例

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-IDENTITY-SET.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SET-OBJECT    USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  ELEMENT1      USAGE OBJECT REFERENCE.
01  ELEMENT2      USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECT
    INVOKE SET-OBJECT "ADD-ELEMENT" USING ELEMENT1          …[1]
    INVOKE SET-OBJECT "ADD-ELEMENT" USING ELEMENT2          …[2]
                :
    INVOKE SET-OBJECT "REMOVE-ELEMENT" USING ELEMENT1       …[3]
                :
```

[例の説明]

[1]
セットに要素(ELEMENT1)を登録します。

[2]
セットに要素(ELEMENT2)を登録します。

[3]
セットから指定した要素(ELEMENT1)を削除します。

5.3.1　NEWメソッド

説明

FJCOL-IDENTITY-SETクラスのセットオブジェクトを作成します。

記述形式

```
INVOKE FJCOL-IDENTITY-SET "NEW"
           RETURNING IDENTITY-SET
```

復帰値

IDENTITY-SET [属性: OBJECT REFERENCE SELF]

FJCOL-IDENTITY-SETクラスのオブジェクトが返却されます。

解説

FJCOL-IDENTITY-SETクラスのオブジェクトを作成します。

使用例

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-IDENTITY-SET.                                  …[1]
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SET-OBJECT USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.     …[2]
PROCEDURE DIVISION.
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECT      …[3]
              :
```

[例の説明]

[1]

リポジトリ段落にFJCOL-IDENTITY-SETクラスを宣言します。

[2]

セットオブジェクトのオブジェクト参照値を保持するためのデータ項目を宣言します。

[3]

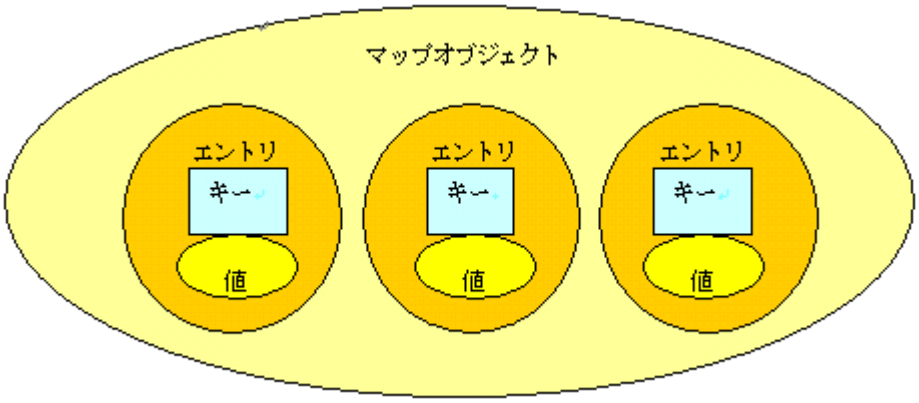
セットオブジェクトを作成します。

第6章　マップクラス

この章では、マップクラスについて説明します。

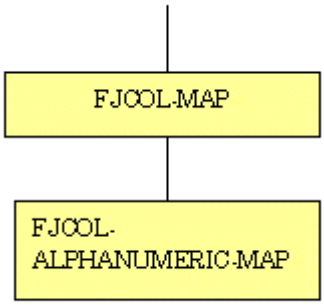
6.1　マップクラスとは

マップクラスは、キーの要素と値の要素の組を操作するクラスです。
キーの要素と値の要素の組のことをエントリと言います。マップクラスは、キーを指定することにより操作対象とするエントリを決定することができます。



【図6.1-1 マップオブジェクトの概念図】

マップクラスには、FJCOL-MAPクラスとFJCOL-ALPHANUMERIC-MAPクラスがあります。



【図6.1-2 マップクラスのクラス階層図】

6.2　FJCOL-MAPクラス

説明

マップを操作します。

解説

FJCOL-MAPクラスはFJBASEクラスを継承しています。
マップクラスはすべてこのクラスを継承します。現在は、当クラスから継承しているクラスはFJCOL-ALPHANUMERIC-MAPクラスしか提供していません。しかし、将来当クラスを継承した別のマップクラスが提供された場合、当クラスを使用してマップクラスに共通の処理を記述することができます。

― FJCOL-MAPクラスは抽象クラス(インタフェース定義だけのクラス)であり、NEWメソッド(ファクトリメソッド)をサポートしていません。

オブジェクトメソッド

メソッド名	機能概要
CLEAR-COLLECTION	現在のマップを空にします
CLONE-COLLECTION	現在のマップの複製を作成します
CREATE-ITERATOR	現在のマップに対してイテレータオブジェクトを作成します
GET-SIZE	現在のマップのエントリ数を返します
IS-EMPTY	現在のマップが空か検査します

使用例

```
PROGRAM-ID. PROGRAM1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-ALPHANUMERIC-MAP
    CLASS FJCOL-MAP.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ALPHANUM-MAP-OBJECT USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECT          USAGE OBJECT REFERENCE FJCOL-MAP.
01  KEY-DATA            PIC X(80).
01  VALUE-OBJECT        USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING ALPHANUM-MAP-OBJECT
    INVOKE ALPHANUM-MAP-OBJECT "PUT-ENTRY" USING KEY-DATA VALUE-OBJECT
                :
    SET MAP-OBJECT TO ALPHANUM-MAP-OBJECT                ┐…[1]
    CALL "PROGRAM2" USING MAP-OBJECT                     ┘
                :
```

```
PROGRAM-ID. PROGRAM2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-MAP
    CLASS FJCOL-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ENTRY-OBJECT     USAGE OBJECT REFERENCE.
01  KEY-DATA         PIC X(80).
01  VALUE-OBJECT     USAGE OBJECT REFERENCE.
01  SIZE-COUNT       PIC S9(9) COMP-5.
              :
LINKAGE SECTION.
01  MAP-OBJECT       USAGE OBJECT REFERENCE FJCOL-MAP.
PROCEDURE DIVISION USING MAP-OBJECT.                          …[2]
    INVOKE MAP-OBJECT "GET-SIZE" RETURNING SIZE-COUNT
    IF SIZE-COUNT NOT = 0 THEN
      INVOKE MAP-OBJECT "CREATE-ITERATOR"
                        RETURNING ITERATOR-OBJECT
      PERFORM SIZE-COUNT TIMES
        INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"
                               RETURNING ENTRY-OBJECT
        INVOKE ENTRY-OBJECT "GET-KEY" USING KEY-DATA
        INVOKE ENTRY-OBJECT "GET-VALUE" RETURNING VALUE-OBJECT
              :
      END-PERFORM
    END-IF
              :
```

[例の説明]
プログラムの引数としてFJCOL-MAPクラスのオブジェクトを受け取ります([2])。この引数のオブジェクト参照値は、FJCOL-MAPクラスを継承しているマップクラスのオブジェクト参照値ならどのクラスのオブジェクトでも構いません。例の場合、PROGRAM1でFJCOL-ALPHANUMERIC-MAPクラスのオブジェクトを設定しています([1])。PROGRAM2ではマップクラスのオブジェクトに共通の処理を記述します。

6.2.1　CLEAR-COLLECTIONメソッド

説明

現在のマップを空にします。

記述形式

```
INVOKE MAP  "CLEAR-COLLECTION"
```

解説

マップを空の状態にします。空の状態とはエントリ数が0の状態をいいます。

使用例

```
PROGRAM-ID. PROGRAM1.
```

```
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  A-MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
01  KEY-DATA        PIC X(10).
01  VALUE-OBJECT    USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING A-MAP-OBJECT
    INVOKE A-MAP-OBJECT "PUT-ENTRY" USING KEY-DATA VALUE-OBJECT
              :
    SET MAP-OBJECT TO A-MAP-OBJECT
    CALL "PROGRAM2" USING MAP-OBJECT
              :
```

```
PROGRAM-ID. PROGRAM2.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
              :
LINKAGE SECTION.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
PROCEDURE DIVISION USING MAP-OBJECT.
              :
    INVOKE MAP-OBJECT "CLEAR-COLLECTION"                    …[1]
              :
```

[例の説明]

[1]

マップを空にします。 (エントリ数=0)

6.2.2　CLONE-COLLECTIONメソッド

説明

現在のマップの複製を作成します。

記述形式

INVOKE *MAP* **"CLONE-COLLECTION"**
RETURNING *MAP-A*

復帰値

MAP-A **[属性: OBJECT REFERENCE CLASS OF SELF]**

複製のマップオブジェクトが返却されます。

解説

現在のマップに登録されているエントリをすべて登録した、新しいマップを作成します。
当メソッドは、マップに登録されているキーや値(オブジェクト)の複製は作成しません。
現在のマップが空である場合、複製として空のマップを作成します。

使用例

```
PROGRAM-ID. PROGRAM1.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  A-MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
01  KEY-DATA        PIC X(10).
01  VALUE-OBJECT    USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING A-MAP-OBJECT
    INVOKE A-MAP-OBJECT "PUT-ENTRY" USING KEY-DATA VALUE-OBJECT
              :
    SET MAP-OBJECT TO A-MAP-OBJECT
    CALL "PROGRAM2" USING MAP-OBJECT
              :
```

```
PROGRAM-ID. PROGRAM2.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-NEW-OBJECT  USAGE OBJECT REFERENCE FJCOL-MAP.
              :
LINKAGE SECTION.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
PROCEDURE DIVISION USING MAP-OBJECT.
              :
    INVOKE MAP-OBJECT "CLONE-COLLECTION"                    …[1]
                      RETURNING MAP-NEW-OBJECT
              :
```

[例の説明]

[1]

現在のマップの複製を作成します。

6.2.3　CREATE-ITERATORメソッド

説明

現在のマップに対してイテレータオブジェクトを作成します。

記述形式

```
INVOKE MAP "CREATE-ITERATOR"
           RETURNING ITERATOR
```

復帰値

ITERATOR [属性: OBJECT REFERENCE FJCOL-ITERATOR]

イテレータオブジェクトが返却されます。

解説

現在のマップのエントリを順に操作できるイテレータオブジェクトを作成します。
イテレータオブジェクトは、イテレータクラスのオブジェクトです。

使用例

```
PROGRAM-ID. PROGRAM1.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  A-MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
01  KEY-DATA        PIC X(10).
01  VALUE-OBJECT    USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING A-MAP-OBJECT
    INVOKE A-MAP-OBJECT "PUT-ENTRY" USING KEY-DATA VALUE-OBJECT
                :
    SET MAP-OBJECT TO A-MAP-OBJECT
    CALL "PROGRAM2" USING MAP-OBJECT
                :
```

```
PROGRAM-ID. PROGRAM2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-MAP
    CLASS FJCOL-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-ITERATOR.
                :
LINKAGE SECTION.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
PROCEDURE DIVISION USING MAP-OBJECT.
                :
    INVOKE MAP-OBJECT "CREATE-ITERATOR"                          …[1]
                      RETURNING ITERATOR-OBJECT
                :
```

[例の説明]

[1]

イテレータオブジェクトを作成します。

6.2.4 GET-SIZEメソッド

説明

現在のマップのエントリ数を返します。

記述形式

```
INVOKE MAP  "GET-SIZE"
            RETURNING COUNT
```

復帰値

COUNT [属性: PIC S9(9) COMP-5]
　エントリ数が返却されます。

解説

現在のマップのエントリ数が返されます。

使用例

```
PROGRAM-ID. PROGRAM1.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  A-MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
01  KEY-DATA        PIC X(10).
01  VALUE-OBJECT    USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING A-MAP-OBJECT
    INVOKE A-MAP-OBJECT "PUT-ENTRY" USING KEY-DATA VALUE-OBJECT
                :
    SET MAP-OBJECT TO A-MAP-OBJECT
    CALL "PROGRAM2" USING MAP-OBJECT
                :
```

```
PROGRAM-ID. PROGRAM2.
                :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ENTRY-COUNT   PIC S9(9) COMP-5.
                :
LINKAGE SECTION.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
PROCEDURE DIVISION USING MAP-OBJECT.
                :
    INVOKE MAP-OBJECT "GET-SIZE" RETURNING ENTRY-COUNT        …[1]
                :
```

[例の説明]
[1]
　マップのエントリ数を獲得します。

6.2.5　IS-EMPTYメソッド

説明

現在のマップが空か検査します。

記述形式

```
INVOKE MAP  "IS-EMPTY"
            RETURNING FG
```

復帰値

FG [属性: PIC 1(1) DISPLAY]

現在のマップが空である場合は、B"1"が返却されます。
現在のマップが空でない場合は、B"0"が返却されます。

解説

現在のマップが空であるか検査します。空とはエントリ数が0の状態をいいます。

使用例

```
PROGRAM-ID. PROGRAM1.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  A-MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
01  KEY-DATA        PIC X(10).
01  VALUE-OBJECT    USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING A-MAP-OBJECT
    INVOKE A-MAP-OBJECT "PUT-ENTRY" USING KEY-DATA VALUE-OBJECT
              :
    SET MAP-OBJECT TO A-MAP-OBJECT
    CALL "PROGRAM2" USING MAP-OBJECT
              :
```

```
PROGRAM-ID. PROGRAM2.
              :
DATA DIVISION.
WORKING-STORAGE SECTION.
01  EMPTY-FG        PIC 1(1) DISPLAY.
              :
LINKAGE SECTION.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-MAP.
PROCEDURE DIVISION USING MAP-OBJECT.
              :
    INVOKE MAP-OBJECT "IS-EMPTY" RETURNING EMPTY-FG          …[1]
    IF EMPTY-FG = B"1" THEN                                  ┐
              :                                              │…[2]
    END-IF                                                   ┘
              :
```

[例の説明]

[1]

マップが空か検査します。

[2]

マップが空であった場合の処理を記述します。

6.3　FJCOL-ALPHANUMERIC-MAPクラス

説明

英数字項目のキーと値（オブジェクト）のエントリを持つコレクションを作成し、操作します。

解説

FJCOL-ALPHANUMERIC-MAPクラスは、FJCOL-MAPクラスを継承しています。
当クラスで操作するエントリのキーには、英数字項目の基本項目または集団項目を指定します。ただし、いずれの場合も先頭から256バイトがキーとして有効です。キーの比較には文字比較の規則が適用されます。つまり、比較対象のデータ項目長が異なる場合、短い方の比較対象の右側には、長い方の比較対象の項目長と同じになるまで空白があるものとみなされます。
各データ項目の基本項目の大きさについては、"COBOL 文法書"を参照してください。
当クラスで操作するエントリの値には、オブジェクトを指定します。
キーの指定方法を例として挙げます。
［例１］
― 以下のKEY1からKEY4は同一のキーが指定されたとみなされます。

```
01  KEY1 PIC X(10) VALUE "1234567890".
01  KEY2 PIC X(20) VALUE "1234567890".
01  KEY3 PIC X(20) VALUE "1234567890          ".
01  KEY4.
  02  KEY4-1 PIC X(5) VALUE "12345".
  02  KEY4-2 PIC X(5) VALUE "67890".
```

― KEY5、KEY6は上記KEY1からKEY4とは別のキーとみなされます。

```
01  KEY5 PIC X(20) VALUE "          1234567890".
01  KEY6 PIC X(20) VALUE "1234568790ABCDEFGHIJ".
```

［例２］
― 以下のKEY7、KEY8は先頭から256 バイトまでがキーとみなされます。

```
01  KEY7 PIC X(500) VALUE ALL "1".
01  KEY8.
  02  KEY8-1 PIC X(100) VALUE ALL "1".
  02  KEY8-2 PIC X(100) VALUE ALL "2".
  02  KEY8-3 PIC X(100) VALUE ALL "3".
```

［例３］
― 以下のKEY9のように集団項目を使用することで英数字以外のデータ項目をキーとすることが可能です。

```
01  KEY9.
  02  KEY9-1  PIC N(4) VALUE NC"営業二部".
  02  KEY9-2  PIC N(4) VALUE NC"営業三課".
  02  KEY9-3  PIC 9(8) VALUE 11111111.
```

［例４］
― 英数字項目以外の基本項目は指定できません。そのため以下の項目はキーとして不適合です。

```
01  KEY10 PIC 9(8) VALUE 12345678.
01  KEY11 PIC N(4) VALUE NC"営業二部".
```

ファクトリメソッド

メソッド名	機能概要
NEW	マップクラスのオブジェクトを作成します

オブジェクトメソッド

メソッド名	機能概要
CONTAINS-KEY	現在のマップに指定したキーが含まれるか検査します
CONTAINS-VALUE	現在のマップに指定した値（オブジェクト）が含まれるか検査します
EQUALS-MAP-COLLECTION	現在のマップと指定されたマップが同等か検査します
GET-VALUE	現在のマップから指定したキーに対する値（オブジェクト）を返します
PUT-ENTRY	現在のマップにエントリ（キーと値）を追加します
PUT-ALL-ENTRY	現在のマップに､指定したマップに含まれるすべてのエントリ(キーと値)を追加します
REMOVE-ENTRY	現在のマップから指定したキーに対するエントリ（キーと値）を削除します

使用例

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-ALPHANUMERIC-MAP.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  KEY1          PIC X(80).
01  VALUE1        USAGE OBJECT REFERENCE.
01  KEY2          PIC X(80).
01  VALUE2        USAGE OBJECT REFERENCE.
01  VALUE-X       USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY1 VALUE1            …[1]
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY2 VALUE2            …[2]
                :
    INVOKE MAP-OBJECT "GET-VALUE" USING KEY2 RETURNING VALUE-X …[3]
                :
```

[例の説明]

[1]

マップにエントリ(キー:KEY1、値:VALUE1)を登録します。

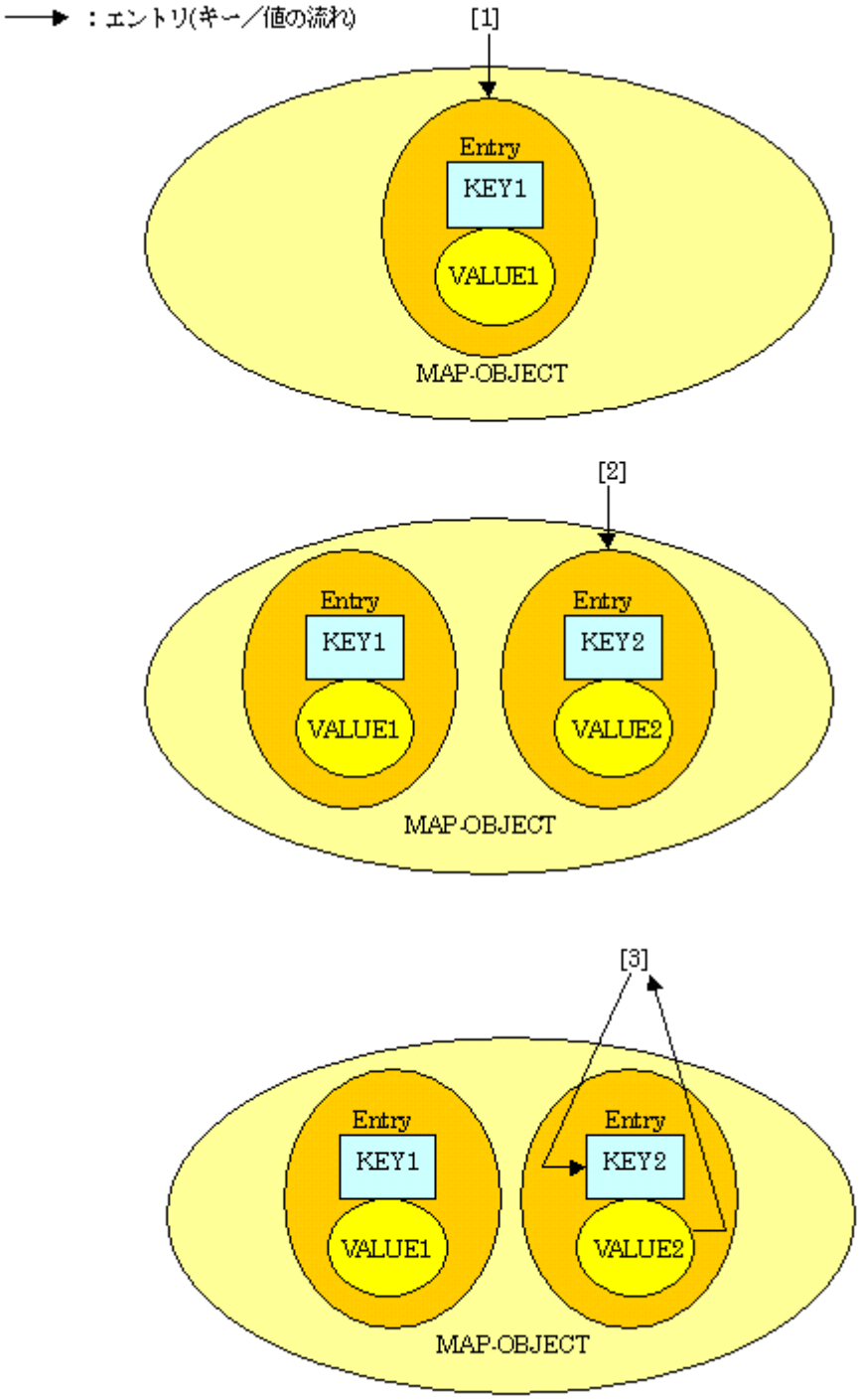
[2]

マップにエントリ(キー:KEY2、値:VALUE2)を登録します。

[3]

指定したキーに対する値を取り出します。

ここでは、キー"KEY2"が指定されているため、[2]で登録した値"VALUE2"が取り出されます。



【図6.3 使用例の概念図 】

6.3.1　NEWメソッド

説明

FJCOL-ALPHANUMERIC-MAPクラスのマップオブジェクトを作成します。

記述形式

```
INVOKE FJCOL-ALPHANUMERIC-MAP "NEW"
```

RETURNING *ALPHANUMERIC-MAP*

復帰値

ALPHANUMERIC-MAP [属性: OBJECT REFERENCE SELF]
FJCOL-ALPHANUMERIC-MAPクラスのオブジェクトが返却されます。

解説

FJCOL-ALPHANUMERIC-MAPクラスのオブジェクトを作成します。

使用例

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-ALPHANUMERIC-MAP.                              …[1]
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP. …[2]
PROCEDURE DIVISION.
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT  …[3]
             :
```

[例の説明]

[1]
リポジトリ段落にFJCOL-ALPHANUMERIC-MAPクラスを宣言します。

[2]
マップオブジェクトのオブジェクト参照値を保持するためのデータ項目を宣言します。

[3]
マップオブジェクトを作成します。

6.3.2　CONTAINS-KEYメソッド

説明

現在のマップに指定したキーが含まれるか検査します。

記述形式

INVOKE *ALPHANUMERIC-MAP* "CONTAINS-KEY"
　　USING *KEY*
　　RETURNING *FG*

引数

KEY [属性: 6.3 FJCOL-ALPHANUMERIC-MAPクラス 解説 参照]
キーを指定します。
英数字項目の基本項目または集団項目でなければなりません。

復帰値

FG [属性: PIC 1(1) DISPLAY]
キーがマップに含まれる場合は、B"1"が返却されます。
キーがマップに含まれない場合は、B"0"が返却されます。

解説

現在のマップに指定したキーが含まれるか検査します。
指定したキーが、現在のマップに登録されているエントリのキーのいずれかと一致する場

合、マップに含まれるとみなします。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  KEY-X         PIC X(256).
01  CONTAINS-FG   PIC 1(1) DISPLAY.
               :
PROCEDURE DIVISION.
               :
    INVOKE MAP-OBJECT "CONTAINS-KEY"                          …[1]
           USING KEY-X RETURNING CONTAINS-FG
    IF CONTAINS-FG = B"1" THEN                              ┐
               :                                            │…[2]
    END-IF                                                  ┘
               :
```

[例の説明]

[1]

指定したキーがマップに含まれるか検査します。

[2]

キーが含まれている場合の処理を記述します。

6.3.3　CONTAINS-VALUEメソッド

説明

現在のマップに指定した値（オブジェクト）が含まれるか検査します。

記述形式

INVOKE *ALPHANUMERIC-MAP* **"CONTAINS-VALUE"**
USING [BY CONTENT] *VALUE*
RETURNING *FG*

引数

VALUE **[属性: OBJECT REFERENCE]**

値(オブジェクト)を指定します。
ただし、BY CONTENTを指定した場合、値(オブジェクト)のクラスで型付けしたオブジェクト一意名を指定することもできます。値のクラスで型付けしたオブジェクト一意名とは、値のクラスをvalue-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  VALUE  OBJECT REFERENCE value-class.
```

復帰値

FG **[属性: PIC 1(1) DISPLAY]**

値(オブジェクト)がマップに含まれる場合は、B"1"が返却されます。
値(オブジェクト)がマップに含まれない場合は、B"0"が返却されます。

解説

現在のマップに指定した値(オブジェクト)が含まれるか検査します。
指定した値(オブジェクト)のオブジェクト参照値が、現在のマップに登録されているエントリの値(オブジェクト)のオブジェクト参照値のいずれかと一致する場合、マップに含ま

れるとみなします。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT    USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  VALUE-X       USAGE OBJECT REFERENCE.
01  CONTAINS-FG   PIC 1(1) DISPLAY.
              :
PROCEDURE DIVISION.
              :
    INVOKE MAP-OBJECT "CONTAINS-VALUE"                          …[1]
                      USING VALUE-X RETURNING CONTAINS-FG
    IF CONTAINS-FG = B"1" THEN                                  ┐
              :                                                 │…[2]
    END-IF                                                      ┘
              :
```

[例の説明]

[1]

指定した値(オブジェクト)がマップに含まれるか検査します。

[2]

値(オブジェクト)が含まれている場合の処理を記述します。

6.3.4　EQUALS-MAP-COLLECTIONメソッド

説明

現在のマップと指定されたマップが同等か検査します。

記述形式

```
INVOKE ALPHANUMERIC-MAP  "EQUALS-MAP-COLLECTION"
               USING [BY CONTENT] ALPHANUMERIC-MAP-A
               RETURNING FG
```

引数

ALPHANUMERIC-MAP-A **[属性: OBJECT REFERENCE]**

FJCOL-ALPHANUMERIC-MAPクラスのオブジェクトを指定します。
ただし、BY CONTENTを指定した場合、クラスで型付けしたオブジェクト一意名を指定することもできます。クラスで型付けしたオブジェクト一意名とは、次のようなオブジェクト一意名をいいます。

```
01  ALPHANUMERIC-MAP-A  OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
```

復帰値

FG **[属性: PIC 1(1) DISPLAY]**

現在のマップと指定したマップが同等である場合は、B"1"が返却されます。
現在のマップと指定したマップが同等でない場合は、B"0"が返却されます。

解説

現在のマップと指定したマップが同等か検査します。マップが同等とは次のような状態をいいます。

― 現在のマップと指定したマップのエントリ数が同じである。かつ
― 指定したマップのすべてのエントリのキーが現在のマップのエントリに含まれる。

かつ

― 指定したマップのそれぞれのエントリのキーに対する値（オブジェクト）のオブジェクト参照値と、そのキーと同一のキーを持つ現在のマップのエントリの値（オブジェクト）のオブジェクト参照値が同じである

― 現在のマップと指定したマップが共に空である場合は、B"1"が返却されます。
― 指定したマップのオブジェクト参照値がNULLである場合は、B"0"が返却されます。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT1 USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECT2 USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  COMPARE-FG        PIC 1(1) DISPLAY.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT1
              :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT2
              :
    INVOKE MAP-OBJECT1 "EQUALS-MAP-COLLECTION"                  …[1]
                       USING BY CONTENT MAP-OBJECT2
                       RETURNING COMPARE-FG
    IF COMPARE-FG = B"1" THEN                                  ┐
              :                                                │…[2]
    END-IF                                                     ┘
              :
```

[例の説明]

[1]

マップが同等か検査します。

[2]

マップが同等だったときの処理を記述します。

6.3.5　GET-VALUEメソッド

説明

現在のマップから指定したキーに対する値（オブジェクト）を返します。

記述形式

```
INVOKE ALPHANUMERIC-MAP  "GET-VALUE"
                  USING KEY
                  RETURNING VALUE
```

引数

KEY [**属性**: 6.3 FJCOL-ALPHANUMERIC-MAPクラス 解説 参照]

キーを指定します。

英数字項目の基本項目または集団項目でなければなりません。

復帰値

VALUE [属性: OBJECT REFERENCE]
値(オブジェクト)が返却されます。

解説

現在のマップのエントリに指定したキーが含まれる場合、そのエントリの値(オブジェクト)を返却します。指定したキーが含まれない場合、NULLが返却されます。

― 指定したキーが含まれる場合でもNULLが返却される場合があります。それは、キーに対するエントリの値(オブジェクト)がNULLとして登録されているときです。NULLが返された場合にどちらの事象であったのか知りたい場合は、CONTAINS-KEYメソッドを呼出し、キーが現在のマップに含まれるか検査する必要があります。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT  USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  KEY1        PIC X(256).
01  VALUE1      USAGE OBJECT REFERENCE.
01  KEY2        PIC X(256).
01  VALUE2      USAGE OBJECT REFERENCE.
01  VALUE-X     USAGE OBJECT REFERENCE.
               :
PROCEDURE DIVISION.
               :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT  …[1]
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY1 VALUE1         ┐…[2]
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY2 VALUE2         ┘
               :
    INVOKE MAP-OBJECT "GET-VALUE" USING KEY1                  …[3]
                                  RETURNING VALUE-X
               :
```

[例の説明]

[1]
マップオブジェクトを作成します。

[2]
エントリを２つ追加します。

[3]
キー"KEY1"を指定して、対応する値(オブジェクト)を獲得します。データ項目VALUE-Xには、VALUE1と同じオブジェクト参照値が返されます。

6.3.6　PUT-ENTRYメソッド

説明

現在のマップにエントリ（キーと値）を追加します。

記述形式

INVOKE *ALPHANUMERIC-MAP* **"PUT-ENTRY"**

```
USING KEY [BY CONTENT] VALUE
RETURNING REPLACE-VALUE
```

引数

KEY [属性: 6.3 FJCOL-ALPHANUMERIC-MAP 解説　参照]

キーを指定します。

英数字項目の基本項目または集団項目でなければなりません。

VALUE [属性: OBJECT REFERENCE]

値(オブジェクト)を指定します。

ただし、BY CONTENTを指定した場合、値(オブジェクト)のクラスで型付けしたオブジェクト一意名を指定することもできます。値のクラスで型付けしたオブジェクト一意名とは、値のクラスをvalue-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  VALUE  OBJECT REFERENCE value-class.
```

復帰値

REPLACE-VALUE [属性: OBJECT REFERENCE]

置き換える前の値(オブジェクト)が返却されます。

解説

指定したキーと値(オブジェクト)をエントリとして、現在のマップに追加します。

ただし、現在のマップの状態によって次のようになります。

- 指定したキーが現在のマップに含まれていない場合、指定したキーと値(オブジェクト)をエントリとして追加します。復帰値にはNULLが返却されます。
- 指定したキーが現在のマップに含まれている場合、指定されたキーに対するエントリの値(オブジェクト)を指定した値(オブジェクト)に置き換えます。復帰値には、置き換える前の値(オブジェクト)が返却されます。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT  USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  KEY1        PIC X(256).
01  VALUE1      USAGE OBJECT REFERENCE.
               :
PROCEDURE DIVISION.
               :
  INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT  …[1]
  INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY1 VALUE1           …[2]
               :
```

[例の説明]

[1]

マップオブジェクトを作成します。

[2]

キーをKEY1、値(オブジェクト)をVALUE1とするエントリを追加します。

6.3.7　PUT-ALL-ENTRYメソッド

説明

現在のマップに指定したマップに含まれるすべてのエントリ（キーと値）を追加します。

記述形式

```
INVOKE ALPHANUMERIC-MAP "PUT-ALL-ENTRY"
            USING [BY CONTENT] ALPHANUMERIC-MAP-A
```

引数

ALPHANUMERIC-MAP-A **[属性: OBJECT REFERENCE]**

FJCOL-ALPHANUMERIC-MAPクラスのオブジェクトを指定します。
ただし、BY CONTENTを指定した場合、クラスで型付けしたオブジェクト一意名を指定することもできます。クラスで型付けしたオブジェクト一意名とは、次のようなオブジェクト一意名をいいます。

```
01  ALPHANUMERIC-MAP-A  OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
```

解説

指定したマップに含まれるすべてのエントリを、現在のマップに追加します。
ただし、現在のマップの状態によって次のようになります。

- 指定したマップの各エントリ(エントリA)のキーが現在のマップのエントリに含まれていない場合、エントリAを現在のマップに追加します。
- 指定したマップの各エントリ(エントリA)のキーが現在のマップのエントリ(エントリB)に含まれている場合、エントリBの値(オブジェクト)をエントリAの値(オブジェクト)で置き換えます。

- 指定したマップのオブジェクト参照値がNULLである場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-PARAMETER)が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECTA USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  MAP-OBJECTB USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  KEY1        PIC X(256).
01  VALUE1      USAGE OBJECT REFERENCE.
01  KEY2        PIC X(256).
01  VALUE2      USAGE OBJECT REFERENCE.
01  KEY3        PIC X(256).
01  VALUE3      USAGE OBJECT REFERENCE.
               :
PROCEDURE DIVISION.
               :
   INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECTA …[1]
   INVOKE MAP-OBJECTA "PUT-ENTRY" USING KEY1 VALUE1          …[2]
   INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECTB …[3]
   INVOKE MAP-OBJECTB "PUT-ENTRY" USING KEY2 VALUE2        ┐ …[4]
   INVOKE MAP-OBJECTB "PUT-ENTRY" USING KEY3 VALUE3        ┘
               :
   INVOKE MAP-OBJECTA "PUT-ALL-ENTRY"                        …[5]
                      USING BY CONTENT MAP-OBJECTB
               :
```

[例の説明]

[1]
マップオブジェクト(MAP-OBJECTA) を作成します。
[2]
MAP-OBJECTA にエントリを１つ追加します。
[3]
マップオブジェクト(MAP-OBJECTB) を作成します。
[4]
MAP-OBJECTB にエントリを２つ追加します。
[5]
MAP-OBJECTA にMAP-OBJECTB のエントリをすべて追加します。この結果により、MAP-OBJECTA のエントリ数は３になります。

6.3.8　REMOVE-ENTRYメソッド

説明

現在のマップから指定したキーに対するエントリ（キーと値）を削除します。

記述形式

```
INVOKE ALPHANUMERIC-MAP  "REMOVE-ENTRY"
                 USING KEY
                 RETURNING VALUE
```

引数

KEY **[属性: 6.3 FJCOL-ALPHANUMERIC-MAPクラス 解説 参照]**
キーを指定します。
英数字項目の基本項目または集団項目でなければなりません。

復帰値

VALUE **[属性: OBJECT REFERENCE]**
削除したエントリの値(オブジェクト)が返却されます。

解説

指定したキーに対するエントリを削除します。
ただし、現在のマップの状態によって次のようになります。
― 指定したキーが現在のマップのエントリに含まれている場合、キーに対するエントリを削除し、そのエントリの値(オブジェクト)を返却します。
― 指定したキーが現在のマップのエントリに含まれていない場合、NULLを返却します。

－ 指定したキーが含まれる場合でもNULLが返却される場合があります。それは、キーに対するエントリの値(オブジェクト)がNULLとして登録されているときです。NULLが返された場合にどちらの事象であったのか知りたい場合は、事前にCONTAINS-KEYメソッドを呼出し、キーが現在のマップに含まれるか検査する必要があります。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT  USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  KEY1        PIC X(256).
01  VALUE1      USAGE OBJECT REFERENCE.
```

```
01  VALUE-X     USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT  …[1]
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY1 VALUE1           …[2]
                :
    INVOKE MAP-OBJECT "REMOVE-ENTRY" USING KEY1               …[3]
                                     RETURNING VALUE-X
                :
```

[例の説明]

[1]

マップオブジェクトを作成します。

[2]

キーをKEY1、値(オブジェクト)をVALUE1とするエントリを追加します。

[3]

KEY1で示されるキーがマップのエントリに含まれる場合、そのキーに対するエントリを削除します。例の場合、[2] で追加したエントリが削除され、VALUE-XにはVALUE1と同じオブジェクト参照値が返却されます。

第7章　イテレータクラス

この章では、イテレータクラスについて説明します。

7.1　イテレータクラスとは

イテレータクラスは、あるコレクションの特定の要素またはエントリを指すイテレータオブジェクトを操作するためのクラスです。
イテレータクラスのオブジェクト(イテレータオブジェクト)は、コレクションに含まれる要素またはエントリを順に操作するために使用します。１つのコレクションから複数のイテレータオブジェクトを作成することができます。

7.1.1　リストのイテレータオブジェクト

リストのイテレータオブジェクトは、コレクションクラスのCREATE-ITERATORメソッドにより作成されます。イテレータオブジェクトは、リストに登録されている要素を順に操作するために使用します。
次のプログラムを見てください。
プログラムの[1]から[3]でUSER-CLASSクラスのオブジェクトを要素としてリストに追加し、[4]でそのリストのイテレータオブジェクトを作成します。[5]のNEXT-ELEMENTメソッドの処理を繰り返すことで、要素を順に読み出します(ここでは、要素数が3であるため3回繰り返します)。つまり、[5]のNEXT-ELEMENTメソッドで返されるデータ項目ELEMENT-Xには[1]から[3]で追加した要素が返されます。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LINKED-LIST
    CLASS FJCOL-ITERATOR
    CLASS USER-CLASS.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ELEMENT1         USAGE OBJECT REFERENCE USER-CLASS.
01  ELEMENT2         USAGE OBJECT REFERENCE USER-CLASS.
01  ELEMENT3         USAGE OBJECT REFERENCE USER-CLASS.
01  ELEMENT-X        USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT

    INVOKE USER-CLASS "NEW" RETURNING ELEMENT1
    INVOKE ELEMENT1 "SET-DATA"
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT"                    …[1]
                       USING BY CONTENT ELEMENT1

    INVOKE USER-CLASS "NEW" RETURNING ELEMENT2
    INVOKE ELEMENT2 "SET-DATA"
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT"                    …[2]
                       USING BY CONTENT ELEMENT2
```

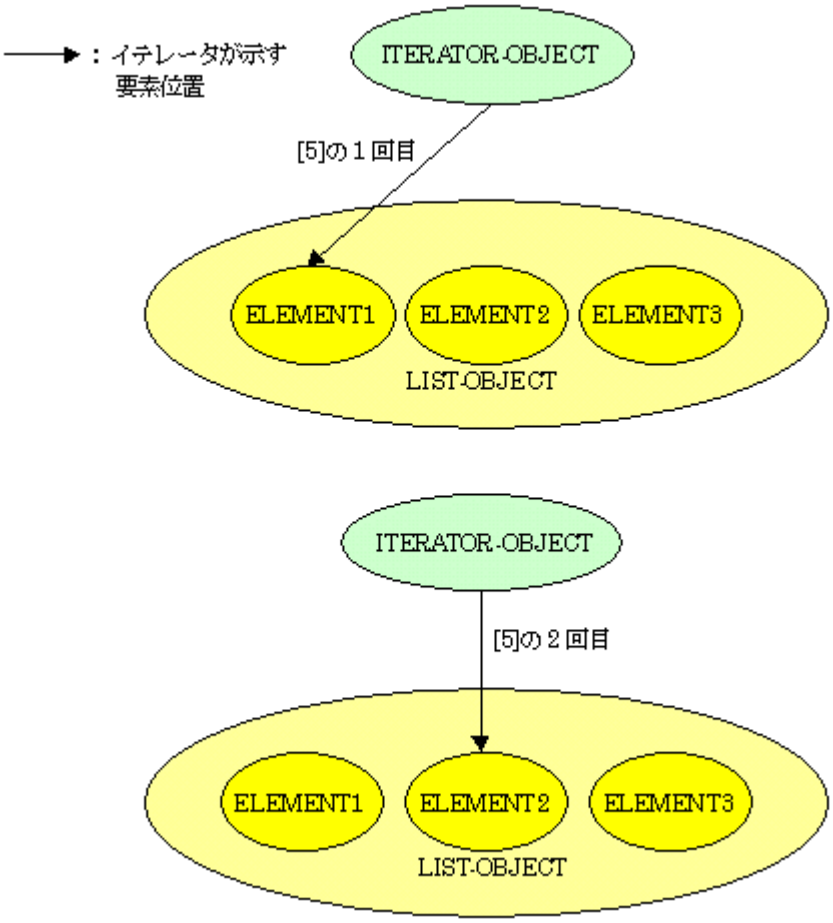
```
INVOKE USER-CLASS "NEW" RETURNING ELEMENT3
INVOKE ELEMENT3 "SET-DATA"
INVOKE LIST-OBJECT "ADD-LAST-ELEMENT"                    …[3]
                   USING BY CONTENT ELEMENT3

INVOKE LIST-OBJECT "CREATE-ITERATOR"                     …[4]
                   RETURNING ITERATOR-OBJECT
PERFORM TEST BEFORE UNTIL
             ITERATOR-OBJECT::"HAS-NEXT-ELEMENT" = B"0"
  INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                  …[5]
                         RETURNING ELEMENT-X
  INVOKE ELEMENT-X "GET-DATA"
END-PERFORM
           :
```

【例7.1.1　リストのイテレータオブジェクト使用例】

［例の補足］

USER-CLASSのSET-DATAメソッドは、要素(オブジェクト)のデータを設定するメソッドです。

USER-CLASSのGET-DATAメソッドは、要素(オブジェクト)のデータを獲得するメソッドです。



【図7.1.1 リストのイテレータオブジェクト使用例の概念図】

7.1.2　セットのイテレータオブジェクト

セットのイテレータオブジェクトは、コレクションクラスのCREATE-ITERATORメソッドにより作成されます。イテレータオブジェクトは、セットに登録されている要素を順に操作するために使用します。

― イテレータオブジェクトで操作するエントリの順番は、セットで内部的に管理している順番であり、セットに追加した順番とは限りません。

次のプログラムを見てください。
プログラムの[1]から[3]でUSER-CLASSクラスのオブジェクトを要素としてセットに追加し、[4]でそのセットのイテレータオブジェクトを作成します。[5]のNEXT-ELEMENTメソッドの処理を繰り返すことで、要素を順に読み出します(ここでは、要素数が3であるため3回繰り返します)。つまり、[5]のNEXT-ELEMENTメソッドで返されるデータ項目ELEMENT-Xには[1]から[3]で追加した要素が返されます。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-IDENTITY-SET
    CLASS FJCOL-ITERATOR
    CLASS USER-CLASS.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SET-OBJECT       USAGE OBJECT REFERENCE FJCOL-IDENTITY-SET.
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ELEMENT1         USAGE OBJECT REFERENCE USER-CLASS.
01  ELEMENT2         USAGE OBJECT REFERENCE USER-CLASS.
01  ELEMENT3         USAGE OBJECT REFERENCE USER-CLASS.
01  ELEMENT-X        USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
    INVOKE FJCOL-IDENTITY-SET "NEW" RETURNING SET-OBJECT

    INVOKE USER-CLASS "NEW" RETURNING ELEMENT1
    INVOKE ELEMENT1 "SET-DATA"
    INVOKE SET-OBJECT "ADD-ELEMENT"                          …[1]
                      USING BY CONTENT ELEMENT1

    INVOKE USER-CLASS "NEW" RETURNING ELEMENT2
    INVOKE ELEMENT2 "SET-DATA"
    INVOKE SET-OBJECT "ADD-ELEMENT"                          …[2]
                      USING BY CONTENT ELEMENT2

    INVOKE USER-CLASS "NEW" RETURNING ELEMENT3
    INVOKE ELEMENT3 "SET-DATA"
```

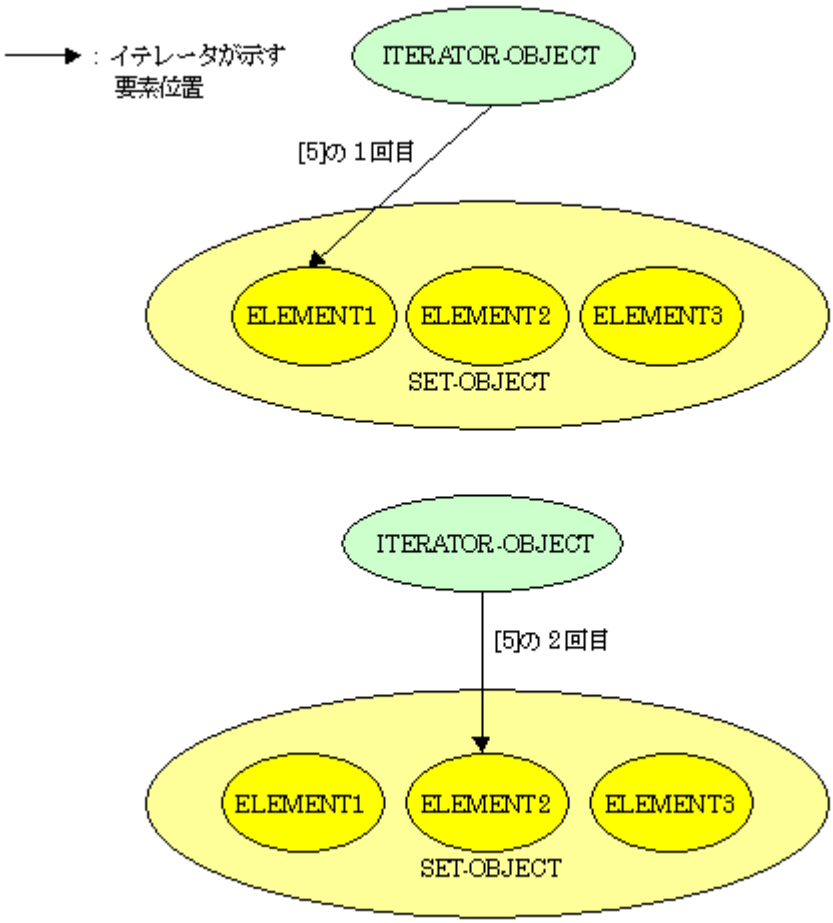
```
INVOKE SET-OBJECT "ADD-ELEMENT"                              …[3]
                  USING BY CONTENT ELEMENT3

INVOKE SET-OBJECT "CREATE-ITERATOR"                          …[4]
                  RETURNING ITERATOR-OBJECT
PERFORM TEST BEFORE UNTIL
            ITERATOR-OBJECT::"HAS-NEXT-ELEMENT" = B"0"
  INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                      …[5]
                         RETURNING ELEMENT-X
  INVOKE ELEMENT-X "GET-DATA"
END-PERFORM
          :
```

【例7.1.2　セットのイテレータオブジェクト使用例】

［例の補足］

USER-CLASSのSET-DATAメソッドは、要素(オブジェクト)のデータを設定するメソッドです。

USER-CLASSのGET-DATAメソッドは、要素(オブジェクト)のデータを獲得するメソッドです。



【図7.1.2 セットのイテレータオブジェクト使用例の概念図】

7.1.3　マップのイテレータオブジェクト

マップのイテレータオブジェクトは、マップクラスのCREATE-ITERATORメソッドにより作成されます。イテレータオブジェクトは、マップに登録されているエントリを順に操作するために使用します。エントリとはキーと値の組のことをいいます。
リストのイテレータオブジェクトで操作する単位は要素ですが、マップのイテレータオブジェクトで操作する単位はエントリであることに注意してください。そのため、NEXT-ELEMENTメソッドおよびREMOVE-ELEMENTメソッドの返却値であるエントリ（正確には、キーと値を持っているマップエントリオブジェクト）に登録されているキーや値の内容を確認したい場合は、エントリに対してマップエントリクラスのメソッドを実行する必要があります。

― イテレータオブジェクトで操作するエントリの順番は、マップで内部的に管理している順番であり、マップに追加した順番とは限りません。

次のプログラムを見てください。
プログラムの[1]から[3]で、キーが10桁の文字項目、値がUSER-CLASSクラスのオブジェクトのエントリをマップに追加し、[4]でそのマップのイテレータオブジェクトを作成します。[5]のNEXT-ELEMENTメソッドの処理を繰り返すことで、エントリを順に読み出します(ここでは、エントリ数が3であるため3回繰り返します)。つまり、[5]のNEXT-ELEMENTメソッドで返されるデータ項目ENTRY-OBJECTには、[1]から[3]で追加したエントリ(マップエントリオブジェクト)が返されます。ENTRY-OBJECTで返されるエントリのキーや値の内容を確認したい場合は、[6]のようにマップエントリクラスのメソッドを実行します。[6]のGET-KEYメソッドで得られるキーが、[1]から[3]でエントリを追加したときのキーであり、GET-VALUEメソッドで得られる値が[1]から[3]でエントリを追加したときの値(オブジェクト)です。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-ALPHANUMERIC-MAP
    CLASS FJCOL-ITERATOR
    CLASS USER-CLASS.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT       USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  KEY1             PIC X(10).
01  VALUE1           USAGE OBJECT REFERENCE USER-CLASS.
01  KEY2             PIC X(10).
01  VALUE2           USAGE OBJECT REFERENCE USER-CLASS.
01  KEY3             PIC X(10).
01  VALUE3           USAGE OBJECT REFERENCE USER-CLASS.
01  ENTRY-OBJECT     USAGE OBJECT REFERENCE.
01  KEY-X            PIC X(10).
01  VALUE-X          USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT

    INVOKE USER-CLASS "NEW" RETURNING VALUE1
```

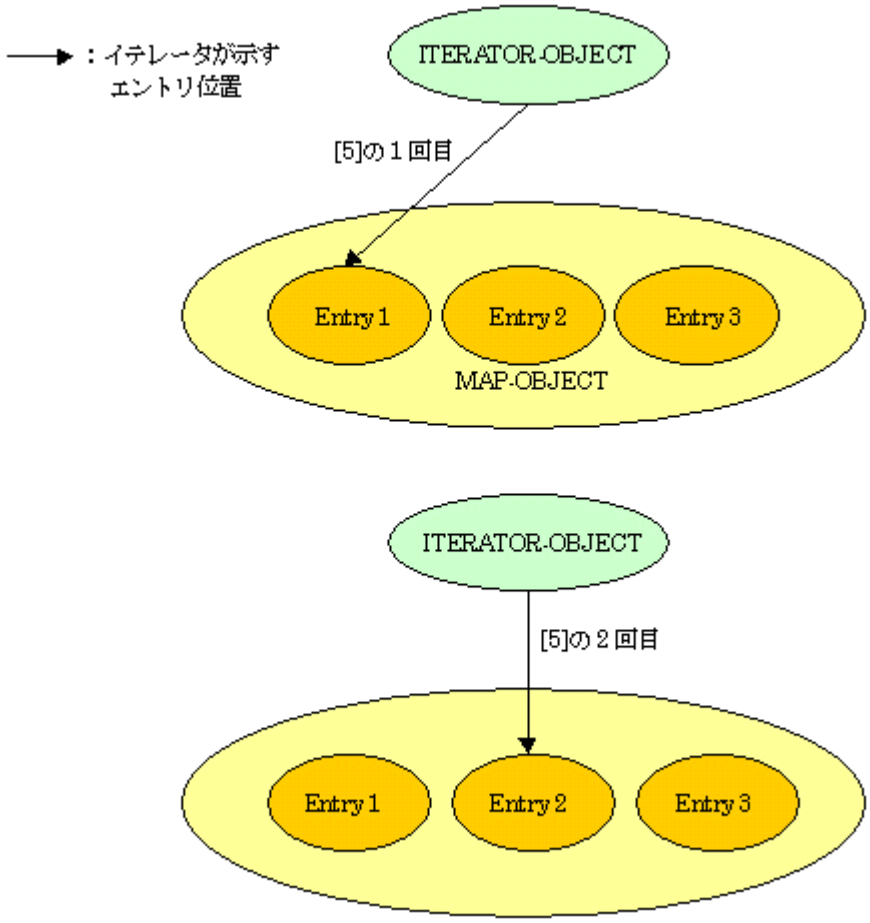
```
INVOKE VALUE1 "SET-DATA" RETURNING KEY1
INVOKE MAP-OBJECT "PUT-ENTRY"                              …[1]
                  USING KEY1 BY CONTENT VALUE1

INVOKE USER-CLASS "NEW" RETURNING VALUE2
INVOKE VALUE2 "SET-DATA" RETURNING KEY2
INVOKE MAP-OBJECT "PUT-ENTRY"                              …[2]
                  USING KEY2 BY CONTENT VALUE2

INVOKE USER-CLASS "NEW" RETURNING VALUE3
INVOKE VALUE3 "SET-DATA" RETURNING KEY3
INVOKE MAP-OBJECT "PUT-ENTRY"                              …[3]
                  USING KEY3 BY CONTENT VALUE3

INVOKE MAP-OBJECT "CREATE-ITERATOR"                        …[4]
                  RETURNING ITERATOR-OBJECT
PERFORM TEST BEFORE UNTIL
        ITERATOR-OBJECT::"HAS-NEXT-ELEMENT" = B"0"
  INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                    …[5]
                         RETURNING ENTRY-OBJECT
  INVOKE ENTRY-OBJECT "GET-KEY" USING KEY-X              ┬…[6]
  INVOKE ENTRY-OBJECT "GET-VALUE"                        ┘
                        RETURNING VALUE-X
  INVOKE VALUE-X "GET-DATA"
           :
END-PERFORM
           :
```

【例7.1.3　マップのイテレータオブジェクト使用例】

［補足］

USER-CLASSのSET-DATAメソッドは、値(オブジェクト)のデータを設定し、キーを返却するメソッドです。

USER-CLASSのGET-DATAメソッドは、値(オブジェクト)のデータを獲得するメソッドです。



【図7.1.3 マップのイテレータオブジェクト使用例の概念図】

7.1.4　イテレータオブジェクトの有効範囲

イテレータオブジェクトは、オブジェクトを作成する時点でのコレクションの要素やエントリとして登録されているオブジェクトを順に操作するものです。そのため、イテレータオブジェクト作成後に、コレクションに対して追加、削除処理を行った場合、イテレータオブジェクトは無効になります。これは、イテレータオブジェクトが管理できる要素やエントリと、コレクションが保持している要素やエントリに違いが生じてしまったからです。ここで言う無効とは、イテレータオブジェクトに対して、追加、削除、要素の返却処理を行うメソッドが実行できない状態を言います。イテレータオブジェクトが無効の状態で追加、削除、要素の返却処理を行うメソッドを実行した場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR)が発生します。
イテレータオブジェクトを使用して、コレクションの要素やエントリを追加、削除することができます。イテレータオブジェクトから追加や削除処理を行っても、そのイテレータオブジェクトは有効なままです。
また、コレクションからイテレータオブジェクトを作成後にコレクションを削除する場合、そのコレクションに属するイテレータオブジェクトもすべて削除してください。
1つのコレクションから複数のイテレータオブジェクトを作成した場合、それぞれのイテレータオブジェクトに有効範囲があります。前述したように、イテレータオブジェクトに対して追加、削除処理を行った場合は、そのイテレータオブジェクトは有効です。ただし、別のイテレータオブジェクトで追加、削除処理を行った場合は、他の同じコレクションから作成されたすべてのイ

テレータオブジェクトは無効になります。
例えば、次のプログラムの[1]のREMOVE-ELEMENTメソッドの処理を行うと、ITERATOR-OBJECT1のイテレータオブジェクトは以降も有効ですが、ITERATOR-OBJECT2のイテレータオブジェクトは無効になります。
また、イテレータオブジェクトの作成元であるリストに対してプログラムの[2]のADD-LAST-ELEMENTメソッドを実行した場合、ITERATOR-OBJECT1のイテレータオブジェクトも無効になります。

```
  DATA DIVISION.
  WORKING-STORAGE SECTION.
  01  LIST-OBJECT      USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
  01  ITERATOR-OBJECT1 USAGE OBJECT REFERENCE FJCOL-ITERATOR.
  01  ITERATOR-OBJECT2 USAGE OBJECT REFERENCE FJCOL-ITERATOR.
                :
  PROCEDURE DIVISION.
      INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT
      INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT
                :
 ↑    INVOKE LIST-OBJECT "CREATE-ITERATOR" RETURNING ITERATOR-OBJECT1
 |↑   INVOKE LIST-OBJECT "CREATE-ITERATOR" RETURNING ITERATOR-OBJECT2
 ||             :
 |(b) INVOKE ITERATOR-OBJECT1 "NEXT-ELEMENT" RETURNING OBJECT-1
(a)|  INVOKE ITERATOR-OBJECT2 "NEXT-ELEMENT" RETURNING OBJECT-2
 |↓             :
 |    INVOKE ITERATOR-OBJECT1 "REMOVE-ELEMENT"                 …[1]
 |                            RETURNING OBJECT-X
 ↓              :
      INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT     …[2]
                :
```

【例7.1.4　イテレータオブジェクトの有効範囲例】

[例の説明]

(a)‥イテレータオブジェクト１の有効範囲
(b)‥イテレータオブジェクト２の有効範囲

7.2　FJCOL-ITERATORクラス

説明

CREATE-ITERATORメソッドで作成されたイテレータオブジェクトを操作します。

解説

FJCOL-ITERATORクラスは、FJBASEを継承しています。

オブジェクトメソッド

メソッド名	機能概要
HAS-NEXT-ELEMENT	イテレータオブジェクトが示す要素があるか検査します
NEXT-ELEMENT	イテレータオブジェクトが示す要素を返します
REMOVE-ELEMENT	イテレータオブジェクトが直前に返した要素を削除します

7.2.1　HAS-NEXT-ELEMENTメソッド

説明

イテレータオブジェクトが示す要素（またはエントリ）があるか検査します。

記述形式

```
INVOKE ITERATOR "HAS-NEXT-ELEMENT"
                RETURNING FG
```

復帰値

FG [属性: PIC 1(1) DISPLAY]

イテレータオブジェクトが示す要素(またはエントリ)がある場合は、B"1"が返却されます。
イテレータオブジェクトが示す要素(またはエントリ)がない場合は、B"0"が返却されます。

解説

イテレータオブジェクトが示す要素(またはエントリ)があるか検査します。
当メソッドを実行することによって、当メソッドの実行直後にNEXT-ELEMENTメソッドを実行した場合に返却する要素(またはエントリ)があるか事前に検査することができます。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITER-OBJECT USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  NEXT-FG     PIC 1(1) DISPLAY.
01  OBJECT-X    USAGE OBJECT REFERENCE.
               :
PROCEDURE DIVISION.
               :
    INVOKE ITER-OBJECT "HAS-NEXT-ELEMENT" RETURNING NEXT-FG   …[1]
    IF NEXT-FG = B"1" THEN
      INVOKE ITER-OBJECT "NEXT-ELEMENT" RETURNING OBJECT-X    …[2]
```

```
                :
    END-IF
                :
```

[例の説明]

[1]

HAS-NEXT-ELEMENTメソッドを実行して、次の要素があるか検査します。

[2]

次の要素がある場合、NEXT-ELEMENTメソッドを実行します。

7.2.2　NEXT-ELEMENTメソッド

説明

イテレータオブジェクトが示す要素（またはエントリ）を返します。

記述形式

```
INVOKE ITERATOR "NEXT-ELEMENT"
                RETURNING ELEMENT
```

復帰値

ELEMENT **[属性: OBJECT REFERENCE]**

イテレータオブジェクトが示す要素またはエントリが返却されます。

解説

イテレータオブジェクトが示す要素(またはエントリ)が返却されます。 ただし、イテレータオブジェクトが示す要素(またはエントリ)がない場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-NOELEMENT)が発生します。
当メソッドの処理終了時には、イテレータオブジェクトが示す要素(またはエントリ)は返した要素の次の要素(またはエントリ)に位置づけられます。

- 当メソッド実行時に、イテレータオブジェクトが無効である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR）が発生します。
- リストから作られたイテレータオブジェクトで例外オブジェクトが発生した場合、コレクション例外クラスのGET-CLASS-NAMEメソッドで返されるクラス名は、"FJCOL-LIST-ITERATOR" になります。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT    USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITER-OBJECT    USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  OBJECT-X       USAGE OBJECT REFERENCE.
01  LIST-SIZE      PIC S9(9) COMP-5.
01  CNT            PIC S9(9) COMP-5.
                :
PROCEDURE DIVISION.
                :
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING LIST-SIZE          …[1]
```

```
INVOKE LIST-OBJECT "CREATE-ITERATOR" RETURNING ITER-OBJECT…[2]
          :
PERFORM TEST BEFORE VARYING CNT FROM 1 BY 1
        UNTIL CNT > LIST-SIZE
  INVOKE ITER-OBJECT "NEXT-ELEMENT" RETURNING OBJECT-X    …[3]
          :
END-PERFORM
          :
```

[例の説明]

[1]
コレクションの要素数を獲得します。

[2]
イテレータオブジェクトを作成します。

[3]
NEXT-ELEMENTメソッドを繰り返し、要素を順に獲得します。

7.2.3　REMOVE-ELEMENTメソッド

説明

イテレータオブジェクトが直前に返した要素（またはエントリ）を削除します。

記述形式

```
INVOKE ITERATOR "REMOVE-ELEMENT"
                RETURNING ELEMENT
```

復帰値

ELEMENT [属性: OBJECT REFERENCE]
削除した要素またはエントリが返却されます。

解説

イテレータオブジェクトが直前に返した要素(またはエントリ)を削除し、削除した要素(またはエントリ)を返却します。
削除対象となる『直前に返した要素(またはエントリ)』とは、直前に実行されたNEXT-ELEMENTメソッドによって返された要素(またはエントリ)です。
イテレータオブジェクトを作成後、NEXT-ELEMENTメソッド実行前に当メソッドが実行された場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-STATE)が発生します。


注意:

- 処理対象となるイテレータオブジェクトが無効である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR）が発生します。
- リストから作られたイテレータオブジェクトで例外オブジェクトが発生した場合、コレクション例外クラスのGET-CLASS-NAMEメソッドで返されるクラス名は、"FJCOL-LIST-ITERATOR" になります。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT    USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
```

```
01  ITER-OBJECT    USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  OBJECT-X       USAGE OBJECT REFERENCE.
01  OBJECT-R       USAGE OBJECT REFERENCE.
01  LIST-SIZE      PIC S9(9) COMP-5.
01  CNT            PIC S9(9) COMP-5.
              :
PROCEDURE DIVISION.
              :
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING LIST-SIZE
    INVOKE LIST-OBJECT "CREATE-ITERATOR" RETURNING ITER-OBJECT…[1]
    PERFORM TEST BEFORE VARYING CNT FROM 1 BY 1
            UNTIL CNT > LIST-SIZE
      INVOKE ITER-OBJECT "NEXT-ELEMENT" RETURNING OBJECT-X    …[2]
      IF  … THEN                                              ┐
        INVOKE ITER-OBJECT "REMOVE-ELEMENT"                   │…[3]
                           RETURNING OBJECT-R                 ┘
      END-IF
    END-PERFORM
              :
```

[例の説明]

[1]

イテレータオブジェクトを作成します。

[2]

NEXT-ELEMENTメソッドを実行し、要素を順に獲得します。

[3]

条件に一致した場合、REMOVE-ELEMENTメソッドにより要素を削除します。削除される要素は、[2]のNEXT-ELEMENTメソッドで直前に獲得した要素です。

第8章　リストイテレータクラス

この章では、リストイテレータクラスについて説明します。

8.1　リストイテレータクラスとは

リストイテレータクラスは、リストの特定の要素を示すリストイテレータオブジェクトを操作するためのクラスです。リストイテレータクラスはイテレータクラスを継承しています。そのため、リストイテレータクラスを理解するためには、まずイテレータクラスを理解しておく必要があります。

イテレータオブジェクトとリストイテレータオブジェクトの違いは、リストイテレータオブジェクトは、リスト中の特定の要素を識別する指標を持つことです。指標を用いて、リストに含まれる要素を双方向に操作することができます。

リストイテレータオブジェクトが持つ指標の値は、NEXT-ELEMENTメソッドおよびPREVIOUS-ELEMENTメソッドによって変更されます。NEXT-ELEMENTメソッドを実行すると指標の値が＋１され、PREVIOUS-ELEMENTメソッドを実行すると－１されます。

リストイテレータオブジェクト作成時の指標の値は、リストイテレータオブジェクトを作成するCREATE-LIST-ITERATORメソッドの実行時に指定された指標の値であり、１から要素数＋１までの範囲です。

― REMOVE-ELEMENTメソッドによって指標の値が変更される場合もあります。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LINKED-LIST
    CLASS FJCOL-LIST-ITERATOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT          USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  LIST-ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  ELEMENT1             USAGE OBJECT REFERENCE.
01  ELEMENT2             USAGE OBJECT REFERENCE.
01  ELEMENT3             USAGE OBJECT REFERENCE.
01  SIZE-COUNT           PIC S9(9) COMP-5.
01  INDEX-VALUE          PIC S9(9) COMP-5 VALUE 1.
01  ELEMENT-X            USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-LINKED-LIST "NEW"  RETURNING LIST-OBJECT       …[1]
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT1     ┐
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT2     │…[2]
    INVOKE LIST-OBJECT "ADD-FIRST-ELEMENT" USING ELEMENT3     ┘
                :
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING SIZE-COUNT          …[3]
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                   …[4]
           USING INDEX-VALUE RETURNING LIST-ITERATOR-OBJECT
    PERFORM TEST BEFORE                                       ┐
     UNTIL LIST-ITERATOR-OBJECT::"HAS-NEXT-ELEMENT" = B"0"    │…[5]
      INVOKE LIST-ITERATOR-OBJECT "NEXT-ELEMENT"              │
```

```
                        RETURNING ELEMENT-X
            :
        END-PERFORM
        PERFORM TEST BEFORE
         UNTIL LIST-ITERATOR-OBJECT::"HAS-PREVIOUS-ELEMENT" = B"0" …[6]
          INVOKE LIST-ITERATOR-OBJECT "PREVIOUS-ELEMENT"
                        RETURNING ELEMENT-X
            :
        END-PERFORM
            :
```

【例8.1　リストイテレータクラスを使用した例】

[例の説明]

[1]

リストオブジェクトを生成します。

[2]

リストオブジェクトに要素のオブジェクトを登録します。

[3]

リストオブジェクトに登録されている現在の要素数を獲得します。

[4]

リストイテレータオブジェクトを作成します。リストイテレータオブジェクトの指標は１として作成されます(引数であるINDEX-VALUE の値が1と指定されているため)。ここで指定できる指標の値は1から[3]で獲得した要素数(SIZE-COUNT)+1の値の範囲内です。

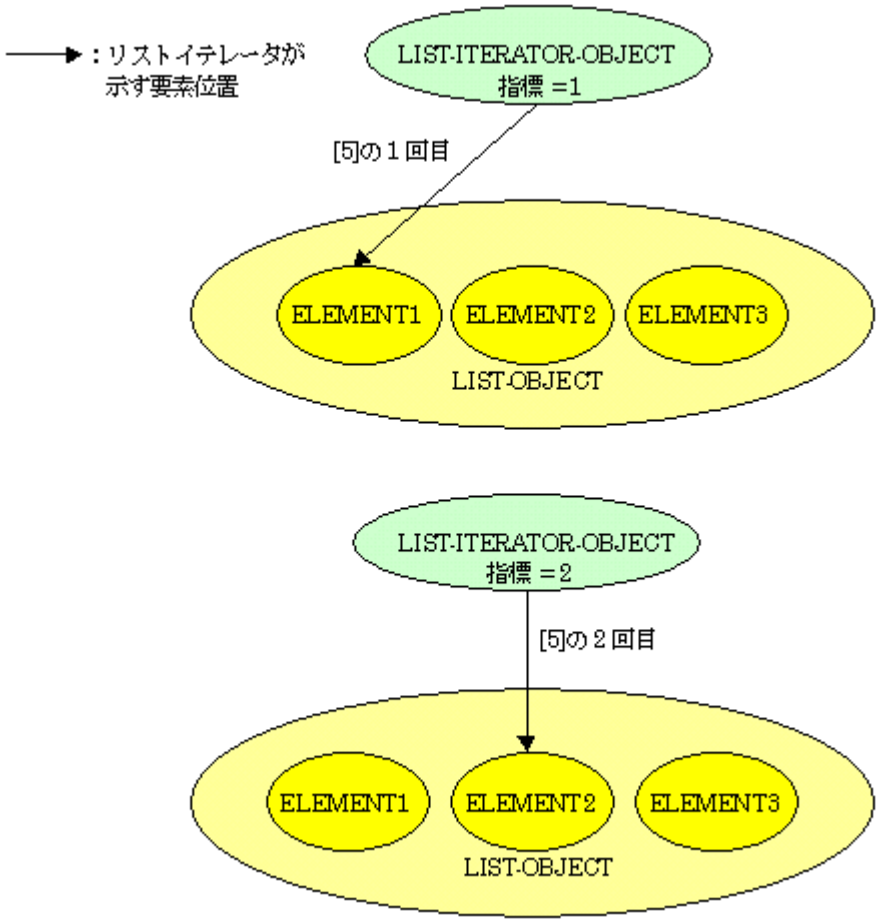
[5]

リストイテレータオブジェクトを使用して、登録されている要素を後方向に順に呼び出します。NEXT-ELEMENTメソッドを実行することにより指標の値が1ずつ増加します。

[6]

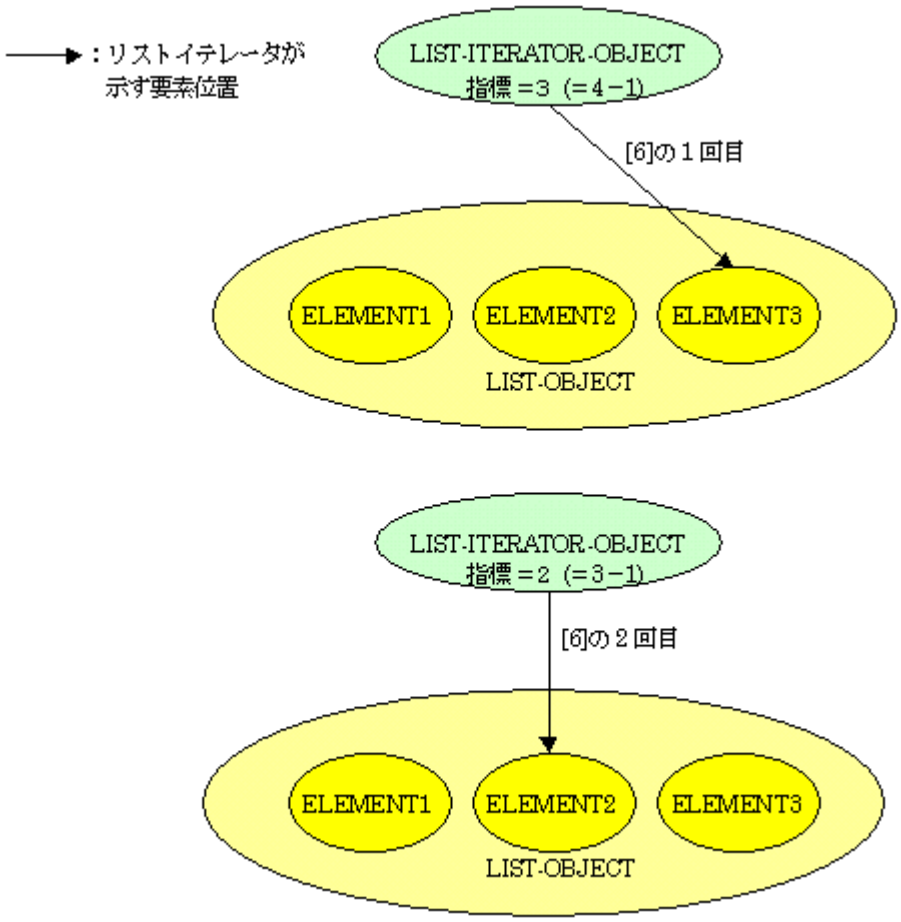
リストイテレータオブジェクトを使用して、登録されている要素を前方向に順に呼び出します。PREVIOUS-ELEMENTメソッドを実行することにより指標の値が1ずつ減少します。ここでは、[5]で最後まで要素を読み込んだリストイテレータオブジェクトが、今度は前方向に最初まで要素を読み込んでいくことになります。

[NEXT-ELEMENTメソッドを実行した場合の指標の値]



【図8.1-1 リストイテレータクラスを使用した例の概念図 -1 】

[PREVIOUS-ELEMENTメソッドを実行した場合の指標の値]



【図8.1-2 リストイテレータクラスを使用した例の概念図 -2 】

8.2　FJCOL-LIST-ITERATORクラス

説明

リストクラスのCREATE-LIST-ITERATORメソッドで作成されたリストイテレータオブジェクトを操作します。

解説

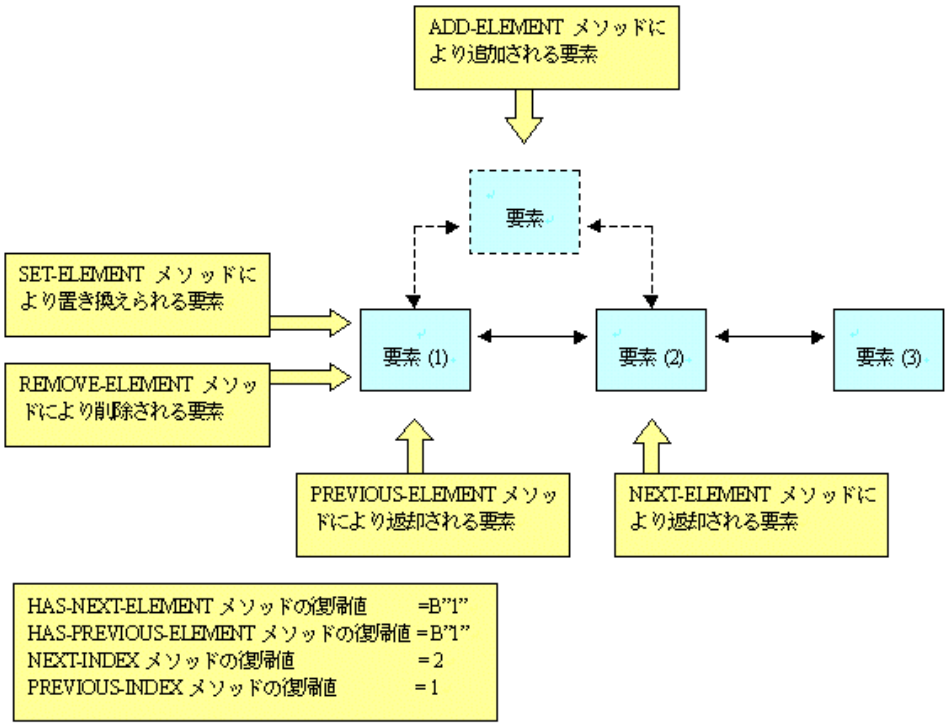
FJCOL-LIST-ITERATORクラスは、FJCOL-ITERATORクラスを継承しています。

オブジェクトメソッド

メソッド名	機能概要
ADD-ELEMENT	指標の値が示す位置に要素を追加します
HAS-NEXT-ELEMENT	直後にNEXT-ELEMENTメソッドを実行した場合に返却する要素があるか検査します
HAS-PREVIOUS-ELEMENT	直後にPREVIOUS-ELEMENTメソッドを実行した場合に返却する要素があるか検査します
NEXT-ELEMENT	指標の値が示す要素を返し、指標の値を＋１します
NEXT-INDEX	指標の値を返します
PREVIOUS-ELEMENT	指標の値を－１してから、指標の値が示す要素を返します
PREVIOUS-INDEX	指標の値より－１した値を返します
REMOVE-ELEMENT	直前に操作した要素を削除します
SET-ELEMENT	直前に操作した要素を変更します

各オブジェクトメソッドの関連

要素数=3のリストに対してNEXT-ELEMENTメソッドにより先頭の要素を返した状態(指標の値=2)から、各オブジェクトメソッドを次に実行させた場合の状態について以下に示します。



【図8.2 各オブジェクトメソッドの関連図】

8.2.1　ADD-ELEMENTメソッド

説明

指標の値が示す位置に要素を追加します。

記述形式

```
INVOKE LIST-ITERATOR "ADD-ELEMENT"
                USING [BY CONTENT] ELEMENT
```

引数

ELEMENT [属性: OBJECT REFERENCE]

要素(オブジェクト)を指定します。
ただし、BY CONTENTを指定した場合、要素のクラスで型付けしたオブジェクト一意名を指定することもできます。要素のクラスで型付けしたオブジェクト一意名とは、要素のクラスをelement-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  ELEMENT  OBJECT REFERENCE element-class.
```

解説

リストイテレータオブジェクトの指標が示す位置に指定した要素を追加します。

― リストイテレータオブジェクトが無効である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR）が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
01  ELEMENT3        USAGE OBJECT REFERENCE.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
01  INDEX-VALUE     PIC S9(9) COMP-5 VALUE 2.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1   ┐
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2   │…[2]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT3   ┘
              :
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                 …[3]
                       USING INDEX-VALUE
                       RETURNING ITERATOR-OBJECT
    INVOKE ITERATOR-OBJECT "ADD-ELEMENT" USING ELEMENT-X      …[4]
```

```
                :
```

[例の説明]

[1]

リストオブジェクトを作成します。

[2]

要素を3つ追加します。

[3]

リストイテレータオブジェクトを作成します。その時の指標の値は引数の値に従い、2となります。

[4]

ADD-ELEMENTメソッドを実行し、要素を追加します。追加する位置は、現在の指標の値=2の位置です。
つまり、要素の順番は次のようになります。

	[1番目の要素]	[2番目の要素]	[3番目の要素]	[4番目の要素]
実行前：	ELEMENT1	ELEMENT2	ELEMENT3	
実行後：	ELEMENT1	ELEMENT-X	ELEMENT2	ELEMENT3

8.2.2　HAS-NEXT-ELEMENTメソッド

説明

直後にNEXT-ELEMENTメソッドを実行した場合に返却する要素があるか検査します。

記述形式

```
INVOKE LIST-ITERATOR "HAS-NEXT-ELEMENT"
                RETURNING FG
```

復帰値

FG [属性: PIC 1(1) DISPLAY]

直後にNEXT-ELEMENTメソッドを実行した場合に返却する要素がある場合は、B"1"が返却されます。
直後にNEXT-ELEMENTメソッドを実行した場合に返却する要素がない場合は、B"0"が返却されます。

解説

直後にNEXT-ELEMENTメソッドを実行した場合に返却する要素があるか検査します。
当メソッドの操作によって、指標の値は変更されません。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  NEXT-FG         PIC 1(1) DISPLAY.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE ITERATOR-OBJECT "HAS-NEXT-ELEMENT"                 …[1]
                           RETURNING NEXT-FG
```

```
IF NEXT-FG = B"1" THEN
  INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                    …[2]
                         RETURNING ELEMENT-X
            :
END-IF
            :
```

[例の説明]

[1]

HAS-NEXT-ELEMENTメソッドを実行して、次の要素があるか検査します。

[2]

次の要素がある場合、NEXT-ELEMENTメソッドを実行します。

8.2.3　HAS-PREVIOUS-ELEMENTメソッド

説明

直後にPREVIOUS-ELEMENTメソッドを実行した場合に返却する要素があるか検査します。

記述形式

INVOKE *LIST-ITERATOR* **"HAS-PREVIOUS-ELEMENT"**
　　RETURNING *FG*

復帰値

FG **[属性: PIC 1(1) DISPLAY]**

直後にPREVIOUS-ELEMENTメソッドを実行した場合に返却する要素がある場合は、B"1"が返却されます。
直後にPREVIOUS-ELEMENTメソッドを実行した場合に返却する要素がない場合は、B"0"が返却されます。

解説

直後にPREVIOUS-ELEMENTメソッドを実行した場合に返却する要素があるか検査します。
当メソッドの操作によって、指標の値は変更されません。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  PREVIOUS-FG     PIC 1(1) DISPLAY.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
            :
PROCEDURE DIVISION.
            :
    INVOKE ITERATOR-OBJECT "HAS-PREVIOUS-ELEMENT"          …[1]
                           RETURNING PREVIOUS-FG
    IF PREVIOUS-FG = B"1" THEN
      INVOKE ITERATOR-OBJECT "PREVIOUS-ELEMENT"            …[2]
                             RETURNING ELEMENT-X
            :
    END-IF
            :
```

[例の説明]

[1]

HAS-PREVIOUS-ELEMENTメソッドを実行して、前の要素があるか検査します。

[2]

前の要素がある場合、PREVIOUS-ELEMENTメソッドを実行します。

8.2.4　NEXT-ELEMENTメソッド

説明

指標の値が示す要素を返し、指標の値を＋１します。

記述形式

```
INVOKE LIST-ITERATOR "NEXT-ELEMENT"
                RETURNING ELEMENT
```

復帰値

ELEMENT [属性: OBJECT REFERENCE]

要素(オブジェクト)が返却されます。

解説

当メソッドは次のように処理を行います。

1) リストイテレータオブジェクトの指標の値が要素数+1の値である場合、指標の値は有効な要素を示していないため、コレクション例外クラスの例外オブジェクト(例外種別：FJCOL-SCE-NOELEMENT)が発生します。
2) 指標の値が示す要素を返却します。
3) 指標の値を+1します。

― リストイテレータオブジェクトが無効である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR）が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
01  LIST-SIZE       PIC S9(9) COMP-5.
01  CNT             PIC S9(9) COMP-5.
                :
PROCEDURE DIVISION.
                :
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING LIST-SIZE          …[1]
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                  …[2]
           USING 1 RETURNING ITERATOR-OBJECT
    PERFORM TEST BEFORE VARYING CNT FROM 1 BY 1
            UNTIL CNT > LIST-SIZE
      INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                    …[3]
                             RETURNING ELEMENT-X
                :
```

```
END-PERFORM
          :
```

[例の説明]

[1]
リストオブジェクトの要素数を獲得します。

[2]
指標の値を1として、リストイテレータオブジェクトを作成します。

[3]
要素数の値だけ、NEXT-ELEMENTメソッドを繰り返し、先頭から順番に要素を獲得します。

8.2.5　NEXT-INDEXメソッド

説明

指標の値を返します。

記述形式

```
INVOKE LIST-ITERATOR "NEXT-INDEX"
             RETURNING INDEX
```

復帰値

INDEX **[属性: PIC S9(9) COMP-5]**
指標の値が返却されます。

解説

当メソッド実行直後にNEXT-ELEMENTメソッドを実行した場合に返される要素に対する指標の値が返されます。
当メソッドの操作によって、指標の値は変更されません。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
01  LIST-SIZE       PIC S9(9) COMP-5.
01  INDEX-VALUE     PIC S9(9) COMP-5.
          :
PROCEDURE DIVISION.
          :
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING LIST-SIZE          …[1]
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                  …[2]
                       USING 1 RETURNING ITERATOR-OBJECT
    PERFORM LIST-SIZE TIMES
      INVOKE ITERATOR-OBJECT "NEXT-INDEX"                      …[3]
                             RETURNING INDEX-VALUE
      IF INDEX-VALUE >= 10 THEN                                …[4]
        EXIT PERFORM
      END-IF
      INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                    …[5]
```

```
                                RETURNING ELEMENT-X
                :
    END-PERFORM
                :
```

[例の説明]

[1]
リストオブジェクトの要素数を獲得します。

[2]
指標の値を1として、リストイテレータオブジェクトを作成します。

[3]
NEXT-INDEXメソッドを実行し、[5]のNEXT-ELEMENTメソッド実行時に獲得される要素の指標の値を得ます。

[4]
[3]で得た指標の値が10であった場合、繰り返し処理を抜けます。つまり、指標の値が10以上の要素については[5]の処理を行わないことになります。

[5]
リストイテレータオブジェクトに対してNEXT-ELEMENTメソッドを実行し、要素を獲得します。

8.2.6　PREVIOUS-ELEMENTメソッド

説明

指標の値を－１してから、指標の値が示す要素を返します。

記述形式

```
INVOKE LIST-ITERATOR "PREVIOUS-ELEMENT"
                RETURNING ELEMENT
```

復帰値

ELEMENT [属性: OBJECT REFERENCE]
要素(オブジェクト)が返却されます。

解説

当メソッドは次のような処理を行います。

1) リストイテレータオブジェクトの指標の値が1である場合、指標の値-1は有効な要素を示していないため、コレクション例外クラスの例外オブジェクト(例外種別：FJCOL-SCE-NOELEMENT)を発生します。
2) 指標の値を-1します。
3) 指標の値が示す要素を返却します。

― リストイテレータオブジェクトが無効である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR）が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
```

```
01  ELEMENT-X       USAGE OBJECT REFERENCE.
01  INDEX-VALUE     PIC S9(9) COMP-5.
01  LIST-SIZE       PIC S9(9) COMP-5.
01  CNT             PIC S9(9) COMP-5.
                :
PROCEDURE DIVISION.
                :
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING LIST-SIZE         …[1]
    COMPUTE INDEX-VALUE = LIST-SIZE + 1
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                 …[2]
           USING INDEX-VALUE RETURNING ITERATOR-OBJECT
    PERFORM TEST BEFORE VARYING CNT FROM INDEX-VALUE BY -1
            UNTIL CNT NOT > 1
      INVOKE ITERATOR-OBJECT "PREVIOUS-ELEMENT"               …[3]
                             RETURNING ELEMENT-X
                :
    END-PERFORM
                :
```

[例の説明]

[1]

リストオブジェクトの要素数を獲得します。

[2]

指標の値を要素数+1として、リストイテレータオブジェクトを作成します。

[3]

要素数+1から1ずつ減算しながら1より大きい間（つまり要素数分）、PREVIOUS-ELEMENTメソッドを繰り返し、末尾から順番に要素を獲得します。

8.2.7　PREVIOUS-INDEXメソッド

説明

指標の値より－１した値を返します。

記述形式

```
INVOKE LIST-ITERATOR  "PREVIOUS-INDEX"
                RETURNING INDEX
```

復帰値

INDEX **[属性: PIC S9(9) COMP-5]**

指標の値が返却されます。

解説

当メソッド実行直後にPREVIOUS-ELEMENTメソッドを実行した場合に返される要素に対する指標の値が返されます。

当メソッドの操作によって、指標の値は変更されません。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
```

```
01  ELEMENT-X       USAGE OBJECT REFERENCE.
01  LIST-SIZE       PIC S9(9) COMP-5.
01  INDEX-VALUE     PIC S9(9) COMP-5.
                :
PROCEDURE DIVISION.
                :
    INVOKE LIST-OBJECT "GET-SIZE" RETURNING LIST-SIZE          …[1]
    COMPUTE INDEX-VALUE = LIST-SIZE + 1
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                  …[2]
           USING INDEX-VALUE RETURNING ITERATOR-OBJECT
    PERFORM LIST-SIZE TIMES
      INVOKE ITERATOR-OBJECT "PREVIOUS-INDEX"                  …[3]
                             RETURNING INDEX-VALUE
      IF INDEX-VALUE <= 10 THEN                                …[4]
        EXIT PERFORM
      END-IF
      INVOKE ITERATOR-OBJECT "PREVIOUS-ELEMENT"                …[5]
                             RETURNING ELEMENT-X
                :
    END-PERFORM
                :
```

[例の説明]

[1]
リストオブジェクトの要素数を獲得します。

[2]
指標の値を要素数+1として、リストイテレータオブジェクトを作成します。

[3]
PREVIOUS-INDEXメソッドを実行し、[5]のPREVIOUS-ELEMENEメソッド実行時に獲得される要素の指標の値を得ます。

[4]
[3]で得た指標の値が10であった場合、繰り返し処理を抜けます。つまり、指標の値が10以下の要素については[5]の処理を行わないことになります。

[5]
リストイテレータオブジェクトに対してPREVIOUS-ELEMENTメソッドを実行し、要素を獲得します。

8.2.8　REMOVE-ELEMENTメソッド

説明

直前に操作した要素を削除します。

記述形式

```
INVOKE LIST-ITERATOR  "REMOVE-ELEMENT"
                RETURNING ELEMENT
```

復帰値

ELEMENT **[属性: OBJECT REFERENCE]**
削除した要素(オブジェクト)が返却されます。

解説

直前に操作した要素を削除し、削除した要素を返却します。

『直前に操作した要素』とは、当メソッドの前に実行されたメソッドにより以下のように定義が異なります。

a) 直前に実行されたメソッドがNEXT-ELEMENTメソッドである場合、直前に操作した要素とはNEXT-ELEMENTメソッドで返された要素を示します。
b) 直前に実行されたメソッドがPREVIOUS-ELEMENTメソッドである場合、直前に操作した要素とはPREVIOUS-ELEMENTメソッドで返された要素を示します。
c) 直前に実行されたメソッドがSET-ELEMENTメソッドである場合、直前に操作した要素とはSET-ELEMENTメソッドで置き換えた要素を示します。
d) 直前に実行したメソッドがADD-ELEMENTメソッドまたはREMOVE-ELEMENTメソッドである場合、削除対象となる要素は存在しません。そのため、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-STATE)が発生します。
e) 直前に実行したメソッドが次のいずれかである場合、その前に実行したメソッドを直前に実行したメソッドとし、a.からd.の規則が適用されます。
   - HAS-NEXT-ELEMENT
   - HAS-PREVIOUS-ELEMENT
   - NEXT-INDEX
   - PREVIOUS-INDEX

当メソッドの操作によって、指標の値が変更されるのは次の場合です。それ以外の場合は、指標の値は変更されません。

― a.の条件にあてはまる場合、指標の値は-1されます。

― リストイテレータオブジェクトが無効である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR）が発生します。
― リストイテレータオブジェクトを作成後に最初に実行されるメソッドが当メソッドである場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-STATE)が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  ELEMENT1        USAGE OBJECT REFERENCE.
01  ELEMENT2        USAGE OBJECT REFERENCE.
01  ELEMENT3        USAGE OBJECT REFERENCE.
01  ELEMENT4        USAGE OBJECT REFERENCE.
01  ELEMENT5        USAGE OBJECT REFERENCE.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
01  LIST-SIZE       PIC S9(9) COMP-5.
01  INDEX-VALUE     PIC S9(9) COMP-5.
01  CNT             PIC S9(9) COMP-5.
              :
PROCEDURE DIVISION.
              :
    INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT      …[1]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT1    ┐
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT2    │…[2]
    INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT3    │
```

```
INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT4     │
INVOKE LIST-OBJECT "ADD-LAST-ELEMENT" USING ELEMENT5     ┘
          :
INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                  …[3]
          USING 1  RETURNING ITERATOR-OBJECT
PERFORM UNTIL ITERATOR-OBJECT::"HAS-NEXT-ELEMENT" = B"0"
  INVOKE ITERATOR-OBJECT "NEXT-INDEX"                      …[4]
                          RETURNING INDEX-VALUE
  INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                    …[5]
                         RETURNING ELEMENT-X
  IF INDEX-VALUE = 3 THEN                                  ┐
    INVOKE ITERATOR-OBJECT "REMOVE-ELEMENT"                │…[6]
    EXIT PERFORM                                           │
  END-IF                                                   ┘
          :
END-PERFORM
INVOKE ITERATOR-OBJECT "NEXT-INDEX"                        …[7]
                        RETURNING INDEX-VALUE
INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                      …[8]
                        RETURNING ELEMENT-X
          :
```

[例の説明]

[1]

リストオブジェクトを作成します。

[2]

リストオブジェクトに要素を追加します。

[3]

リストイテレータオブジェクトを作成します。

[4]

NEXT-INDEXメソッドを実行し、[5]で獲得する要素の指標値を得ます。

[5]

NEXT-ELEMENTメソッドを実行し、要素を獲得します。

[6]

[4]で得た指標値が3の場合、[5]で獲得した要素を削除します。この削除処理によって、[2]で追加したELEMENT3の要素を削除します(下記の削除後の状態)。

指標値：	1	2	3	4	5
削除前：	ELEMENT1	ELEMENT2	ELEMENT3	ELEMENT4	ELEMENT5
削除後：	ELEMENT1	ELEMENT2	ELEMENT4	ELEMENT5	

[7]

NEXT-INDEXメソッドを実行し、[8]で獲得する要素の指標値を得ます。
[6]のREMOVE-ELEMENTメソッドの処理により指標の値も-1されるため、[8]で返される要素の指標の値は3になります。

[8]

NEXT-ELEMENTメソッドを実行し、要素を獲得します。
返却される要素はELEMENT4になります。

8.2.9 SET-ELEMENTメソッド

説明

直前に操作した要素を変更します。

記述形式

```
INVOKE LIST-ITERATOR "SET-ELEMENT"
                USING [BY CONTENT] NEW-ELEMENT
                RETURNING OLD-ELEMENT
```

引数

NEW-ELEMENT [属性: OBJECT REFERENCE]

置き換える要素(オブジェクト)を指定します。
ただし、BY CONTENTを指定した場合、要素のクラスで型付けしたオブジェクト一意名を指定することもできます。要素のクラスで型付けしたオブジェクト一意名とは、要素のクラスをelement-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  NEW-ELEMENT  OBJECT REFERENCE element-class.
```

復帰値

OLD-ELEMENT [属性: OBJECT REFERENCE]

置き換えられる前の要素(オブジェクト)が返却されます。

解説

直前に操作した要素を指定した要素に置き換え、置き換える前の要素を復帰値に返却します。
『直前に操作した要素』とは、当メソッドの前に実行されたメソッドにより定義が異なります。

a) 直前に実行されたメソッドがNEXT-ELEMENTメソッドまたはPREVIOUS-ELEMENTメソッドである場合、直前に操作した要素とはNEXT-ELEMENTまたはPREVIOUS-ELEMENTメソッドで返された要素を示します。
b) 直前に実行されたメソッドがSET-ELEMENTメソッドである場合、直前に操作した要素とはSET-ELEMENTメソッドで置き換えた要素を示します。
c) 直前に実行したメソッドがADD-ELEMENTメソッドまたはREMOVE-ELEMENTメソッドである場合、置き換え対象となる要素は存在しません。そのため、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-STATE)が発生します。
d) 直前に実行したメソッドが次のいずれかである場合、その前に実行したメソッドを直前に実行したメソッドとし、a.からc.の規則が適用されます。
   - HAS-NEXT-ELEMENT
   - HAS-PREVIOUS-ELEMENT
   - NEXT-INDEX
   - PREVIOUS-INDEX

当メソッドの操作によって、指標の値は変更されません。

— リストイテレータオブジェクトが無効である場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-INVALID-ITERATOR）が発生します。
— リストイテレータオブジェクトを作成後に最初に実行されるメソッドが当メソッドである場合、コレクション例外クラスの例外オブジェクト（例外種別：FJCOL-SCE-ILLEGAL-STATE)が発生します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT     USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-LIST-ITERATOR.
01  ELEMENT-X       USAGE OBJECT REFERENCE.
01  NEW-ELEMENT     USAGE OBJECT REFERENCE.
01  OLD-ELEMENT     USAGE OBJECT REFERENCE.
01  INDEX-VALUE     PIC S9(9) COMP-5 VALUE 1.
              :
PROCEDURE DIVISION.
              :
    INVOKE LIST-OBJECT "CREATE-LIST-ITERATOR"                 …[1]
           USING INDEX-VALUE RETURNING ITERATOR-OBJECT
    PERFORM TEST BEFORE UNTIL
            ITERATOR-OBJECT::"HAS-NEXT-ELEMENT" = B"0"
      INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                   …[2]
                             RETURNING ELEMENT-X
              :
      IF  … THEN                                          ┐
        INVOKE ITERATOR-OBJECT "SET-ELEMENT"              │…[3]
                               USING NEW-ELEMENT          │
                               RETURNING OLD-ELEMENT      │
      END-IF                                              ┘
    END-PERFORM
              :
```

[例の説明]

[1]

リストイテレータオブジェクトを作成します。

[2]

NEXT-ELEMENTメソッドを実行し、要素を獲得します。

[3]

条件に一致した場合、SET-ELEMENTメソッドにより要素を置き換えます。置き換えられる要素は、[2]のNEXT-ELEMENTメソッドで直前に獲得した要素です。

第9章　マップエントリクラス

この章では、マップエントリクラスについて説明します。

9.1　マップエントリクラスとは

マップエントリクラスは、マップに対してイテレータを作成した際にNEXT-ELEMENTメソッドおよびREMOVE-ELEMENTメソッドの処理で返却されるエントリ(キーと値の組)のオブジェクト(マップエントリオブジェクト)を操作するためのクラスです。
マップエントリオブジェクトを操作することで、エントリに保持しているキーや値を獲得、変更することができます。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-ALPHANUMERIC-MAP
    CLASS FJCOL-ITERATOR
    CLASS VALUE-CLASS.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT       USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  KEY1             PIC X(256).
01  VALUE1           USAGE OBJECT REFERENCE VALUE-CLASS.
                :
01  ITERATOR-OBJECT  USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  SIZE-COUNT       PIC S9(9) COMP-5.
01  CNT              PIC S9(9) COMP-5.
01  ENTRY-OBJECT     USAGE OBJECT REFERENCE.
01  KEY-X            PIC X(256).
01  VALUE-X          USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT    …[1]
    INVOKE VALUE-CLASS "NEW" RETURNING VALUE1
    INVOKE VALUE1 "SET-DATA" RETURNING KEY1
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY1 BY CONTENT VALUE1  …[2]
                :
    INVOKE MAP-OBJECT "GET-SIZE" RETURNING SIZE-COUNT           …[3]
    INVOKE MAP-OBJECT "CREATE-ITERATOR"                         …[4]
                      RETURNING ITERATOR-OBJECT
    PERFORM TEST BEFORE VARYING CNT FROM 1 BY 1
            UNTIL CNT < SIZE-COUNT
      INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                     …[5]
                             RETURNING ENTRY-OBJECT
      INVOKE ENTRY-OBJECT "GET-KEY" USING KEY-X                 …[6]
      INVOKE ENTRY-OBJECT "GET-VALUE" RETURNING VALUE-X         …[7]
      INVOKE VALUE-X "GET-DATA"
                :
    END-PERFORM
                :
```

【例9.1　マップエントリオブジェクトの使用例】

[例の説明]

[1]

マップオブジェクトを作成します。

[2]

マップオブジェクトにエントリ(キーと値)を登録します。

[3]

マップオブジェクトに登録されている現在のエントリ数を獲得します。

[4]

マップオブジェクトのエントリを操作するイテレータオブジェクトを作成します。

[5]

イテレータオブジェクトを使用して、登録されているエントリを順に呼び出します。
返されるのは、マップエントリオブジェクトです。

[6]

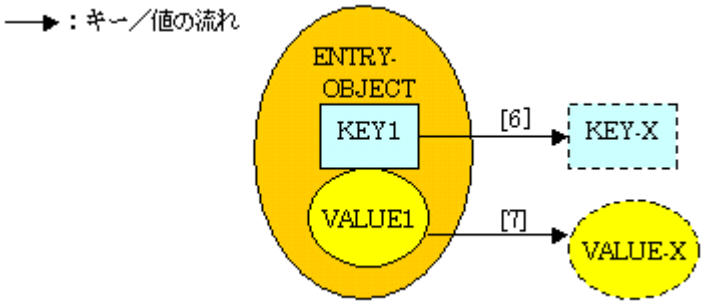
[5]で獲得したエントリのキーの情報を得ます。

[7]

[5]で獲得したエントリの値の情報を得ます。

[補足]

VALUE-CLASSのSET-DATAメソッドは、値(オブジェクト)のデータを設定し、キーを返却するメソッドです。
VALUE-CLASSのGET-DATAメソッドは、値(オブジェクト)のデータを獲得するメソッドです。



【図9.1 マップエントリオブジェクトの使用例の概念図】

9.2　FJCOL-ALPHANUMERIC-MAP-ENTRYクラス

説明

イテレータオブジェクトの操作により返されたエントリのオブジェクト(マップエントリオブジェクト）を操作します。

解説

FJCOL-ALPHANUMERIC-MAP-ENTRYクラスは、FJBASEクラスを継承しています。

オブジェクトメソッド

メソッド名	機能概要
EQUALS-ENTRY	現在のエントリと指定されたエントリが同等か検査します
GET-KEY	現在のエントリに対するキーを返します
GET-VALUE	現在のエントリに対する値(オブジェクト)を返します
SET-VALUE	現在のエントリに対する値(オブジェクト)を指定された値(オブジェクト)に置き換えます

9.2.1　EQUALS-ENTRYメソッド

説明

現在のエントリと指定されたエントリが同等か検査します。

記述形式

```
INVOKE MAP-ENTRY "EQUALS-ENTRY"
                 USING [BY CONTENT] MAP-ENTRY-A
                 RETURNING FG
```

引数

MAP-ENTRY-A [属性: OBJECT REFERENCE]

マップエントリクラスのオブジェクトを指定します。
ただし、BY CONTENTを指定した場合、FJCOL-ALPHANUMERIC-MAP-ENTRYクラスで型付けしたオブジェクト一意名を指定することもできます。クラスで型付けしたオブジェクト一意名とは、次のようなオブジェクト一意名をいいます。

```
01  MAP-ENTRY-A  OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP-ENTRY.
```

復帰値

FG [属性: PIC 1(1) DISPLAY]

指定したエントリが現在のエントリと同等である場合は、B"1"が返却されます。
指定したエントリが現在のエントリと同等でない場合は、B"0"が返却されます。

解説

指定したエントリが現在のエントリと同等であるか検査します。ここでいうエントリとは、マップで作成されたイテレータオブジェクトからNEXT-ELEMENTメソッドおよびREMOVE-ELEMENTメソッドで返されたエントリ(マップエントリオブジェクト)をいいます。
エントリが同等とは次のような状態を言います。

- 現在のエントリのキーと指定したエントリのキーが同値である。かつ
- 現在のエントリの値(オブジェクト)と指定したエントリの値(オブジェクト)のオブジェクト参照値が同じである

― 指定したエントリのオブジェクト参照値がNULLであった場合は、B"0"が返却されます。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITER-OBJECT1      USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ITER-OBJECT2      USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ENTRY-OBJECT1     USAGE OBJECT REFERENCE.
01  ENTRY-OBJECT2     USAGE OBJECT REFERENCE.
01  EQUAL-FG          PIC 1(1) DISPLAY.
                :
PROCEDURE DIVISION.
                :
    INVOKE ITER-OBJECT1 "NEXT-ELEMENT" RETURNING ENTRY-OBJECT1…[1]
                :
    INVOKE ITER-OBJECT2 "NEXT-ELEMENT" RETURNING ENTRY-OBJECT2…[2]
                :
    INVOKE ENTRY-OBJECT1 "EQUALS-ENTRY"                        …[3]
                USING ENTRY-OBJECT2 RETURNING EQUAL-FG
    IF EQUAL-FG = B"1" THEN                                    ┐
                :                                              │…[4]
    END-IF                                                     ┘
                :
```

[例の説明]

[1]
イテレータオブジェクト(ITER-OBJECT1)が示すエントリを獲得します。

[2]
イテレータオブジェクト(ITER-OBJECT2)が示すエントリを獲得します。

[3]
[1]、[2]で獲得したエントリが同等か検査します。

[4]
エントリが同等だったときの処理を記述します。

9.2.2　GET-KEYメソッド

説明

現在のエントリに対するキーを返します。

記述形式

```
INVOKE MAP-ENTRY  "GET-KEY"
                  USING KEY
```

引数

KEY **[属性:** 6.3 FJCOL-ALPHANUMERIC-MAPクラス 解説 参照**]**
キーを指定します。
英数字項目の基本項目または集団項目でなければなりません。

解説

現在のエントリに対するキーが返されます。
登録時に指定したキーのデータ型に関係なく、ここで指定したキーのデータ型に合わせて値が返されます。ただし、返される値はキーとみなしている256バイト以内の値です。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITER-OBJECT   USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ENTRY-OBJECT  USAGE OBJECT REFERENCE.
01  KEY-X         PIC X(256).
               :
PROCEDURE DIVISION.
               :
    INVOKE ITER-OBJECT "NEXT-ELEMENT" RETURNING ENTRY-OBJECT  …[1]
    INVOKE ENTRY-OBJECT "GET-KEY" USING KEY-X                 …[2]
               :
```

[例の説明]

[1]
エントリを獲得します。

[2]
[1]で獲得したエントリのキーを得ます。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MAP-OBJECT      USAGE OBJECT REFERENCE FJCOL-ALPHANUMERIC-MAP.
01  ITERATOR-OBJECT USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ENTRY-OBJECT    USAGE OBJECT REFERENCE.
01  VALUE1          USAGE OBJECT REFERENCE.
01  VALUE2          USAGE OBJECT REFERENCE.
01  KEY1.
  02  KEY1-1        PIC X(5) VALUE "12345".
  02  KEY1-2        PIC X(5) VALUE "67890".
01  KEY2            PIC X(500) VALUE ALL "1".
01  KEY-X.
  02  KEY-X-1       PIC X(7).
  02  KEY-X-2       PIC X(3).
01  KEY-Y           PIC X(500).
               :
PROCEDURE DIVISION.
               :
    INVOKE FJCOL-ALPHANUMERIC-MAP "NEW" RETURNING MAP-OBJECT
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY1 VALUE1           …[1]
    INVOKE MAP-OBJECT "PUT-ENTRY" USING KEY2 VALUE2           …[2]
    INVOKE MAP-OBJECT "CREATE-ITERATOR"
                      RETURNING ITERATOR-OBJECT
    INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"
```

```
                    RETURNING ENTRY-OBJECT
  INVOKE ENTRY-OBJECT "GET-KEY" USING KEY-X                    …[3]
  INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"
                    RETURNING ENTRY-OBJECT
  INVOKE ENTRY-OBJECT "GET-KEY" USING KEY-Y                    …[4]
           :
```

[例の説明]

[1]

エントリを登録します。

[2]

エントリを登録します。

[3]

操作するエントリが[1]で登録したエントリである場合、データ項目KEY-Xに返される値は次のようになります。

KEY-X-1 = "1234567"

KEY-X-2 = "890"

[4]

操作するエントリが[2]で登録したエントリである場合、データ項目KEY-Yに返される値は次のようになります。(指定されたデータ項目長が256バイトより長いときは以降は空白詰めされます。)

KEY-Y = "11111111111…111　　…　"

256文字"1"　　244文字空白

9.2.3　GET-VALUEメソッド

説明

現在のエントリに対する値（オブジェクト）を返します。

記述形式

INVOKE *MAP-ENTRY* "GET-VALUE"
RETURNING *VALUE*

復帰値

***VALUE* [属性: OBJECT REFERENCE]**

値(オブジェクト)か返却されます。

解説

現在のエントリに対する値(オブジェクト)が返却されます。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT   USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ENTRY-OBJECT      USAGE OBJECT REFERENCE.
01  VALUE-X           USAGE OBJECT REFERENCE.
              :
PROCEDURE DIVISION.
              :
   INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                       …[1]
                          RETURNING ENTRY-OBJECT
```

```
    INVOKE ENTRY-OBJECT "GET-VALUE" RETURNING VALUE-X          …[2]
                :
```

[例の説明]

[1]

エントリを獲得します。

[2]

[1]で獲得したエントリの値(オブジェクト)を得ます。

9.2.4　SET-VALUEメソッド

説明

現在のエントリに対する値（オブジェクト）を指定された値（オブジェクト）に置き換えます。

記述形式

```
INVOKE MAP-ENTRY "SET-VALUE"
            USING [BY CONTENT] NEW-VALUE
            RETURNING OLD-VALUE
```

引数

NEW-VALUE **[属性: OBJECT REFERENCE]**

置き換える値(オブジェクト)を指定します。

ただし、BY CONTENTを指定した場合、値(オブジェクト)のクラスで型付けしたオブジェクト一意名を指定することもできます。値のクラスで型付けしたオブジェクト一意名とは、値のクラスをvalue-classとした場合は、次のようなオブジェクト一意名をいいます。

```
01  NEW-VALUE  OBJECT REFERENCE value-class.
```

復帰値

OLD-VALUE **[属性: OBJECT REFERENCE]**

置き換えられる前の値(オブジェクト)が返却されます。

解説

現在のエントリの値(オブジェクト)を指定した値(オブジェクト)に置き換え、置き換える前の値(オブジェクト)を返却します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ITERATOR-OBJECT   USAGE OBJECT REFERENCE FJCOL-ITERATOR.
01  ENTRY-OBJECT      USAGE OBJECT REFERENCE.
01  NEW-VALUE         USAGE OBJECT REFERENCE.
01  OLD-VALUE         USAGE OBJECT REFERENCE.
                :
PROCEDURE DIVISION.
                :
    INVOKE ITERATOR-OBJECT "NEXT-ELEMENT"                      …[1]
                           RETURNING ENTRY-OBJECT
    INVOKE ENTRY-OBJECT "SET-VALUE" USING NEW-VALUE            …[2]
                                    RETURNING OLD-VALUE
```

:

[例の説明]

[1]

エントリを獲得します。

[2]

[1]で獲得したエントリの値(オブジェクト)を置き換えます。

第10章　コレクション例外クラス

この章では、コレクション例外クラスについて説明します。

10.1　コレクション例外クラスとは

コレクションクラスライブラリのクラスは、処理中に何らかのエラーを検出した場合、例外オブジェクトを発生させます。コレクション例外クラスは、その例外オブジェクトのクラスです。
コレクション例外クラスのメソッドを実行することにより，例外が発生したクラス名、メソッド名、例外種別および例外メッセージを取り出すことができます。
例外オブジェクトおよび例外オブジェクトが発生した場合の例外処理の詳細については、"NetCOBOL 使用手引書 " を参照してください。
この章では、コレクション例外クラスのメソッドを中心に説明します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJCOL-LINKED-LIST
    CLASS FJCOL-EXCEPTION.                                    …[1]
DATA DIVISION.
WORKING-STORAGE SECTION.
01  LIST-OBJECT USAGE OBJECT REFERENCE FJCOL-LINKED-LIST.
01  GET-OBJECT  USAGE OBJECT REFERENCE.
*  Exception Information
01  CLASS-NAME               PIC X(80).
01  CLASS-NAME-LENGTH        PIC S9(9) COMP-5.
01  METHOD-NAME              PIC X(80).
01  METHOD-NAME-LENGTH       PIC S9(9) COMP-5.
01  EXCEPTION-CODE           PIC S9(9) COMP-5.
01  EXCEPTION-MESSAGE        PIC X(256).
01  EXCEPTION-MESSAGE-LENGTH PIC S9(9) COMP-5.
              :
PROCEDURE DIVISION.
DECLARATIVES.
ERR SECTION.
    USE AFTER EXCEPTION FJCOL-EXCEPTION.                      …[2]
    INVOKE EXCEPTION-OBJECT "GET-CLASS-NAME"                  …[3]
                        USING     CLASS-NAME
                        RETURNING CLASS-NAME-LENGTH.
    DISPLAY CLASS-NAME ( 1 : CLASS-NAME-LENGTH ).
    INVOKE EXCEPTION-OBJECT "GET-METHOD-NAME"
                        USING     METHOD-NAME
                        RETURNING METHOD-NAME-LENGTH.
    DISPLAY METHOD-NAME ( 1 : METHOD-NAME-LENGTH ).
    INVOKE EXCEPTION-OBJECT "GET-CODE"
                        RETURNING EXCEPTION-CODE.
    DISPLAY EXCEPTION-CODE.
    INVOKE EXCEPTION-OBJECT "GET-MESSAGE"
                        USING     EXCEPTION-MESSAGE
                        RETURNING EXCEPTION-MESSAGE-LENGTH.
    DISPLAY EXCEPTION-MESSAGE ( 1 : EXCEPTION-MESSAGE-LENGTH ).
END DECLARATIVES.
              :
```

```
INVOKE FJCOL-LINKED-LIST "NEW" RETURNING LIST-OBJECT  …[4]
INVOKE LIST-OBJECT "GET-FIRST-ELEMENT"                …[5]
                   RETURNING GET-OBJECT
          :
```

【例10.1　コレクション例外クラスの使用例】

[例の説明]

[1]

コレクション例外クラス"FJCOL-EXCEPTION"をリポジトリ段落に記述します。

[2]

例外オブジェクトに対するUSE文を記述します。

[3]

コレクション例外クラスのメソッドを実行します。

[例の流れ]

新しく作成したリストオブジェクト([4]) に対してすぐにGET-FIRST-ELEMENTメソッドを実行します([5])。要素が登録されていない状態でGET-FIRST-ELEMENTメソッドを実行すると、例外オブジェクトが発生します。プログラム中にコレクション例外クラスFJCOL-EXCEPTIONに対するUSE文([2])が記述されているため、制御はINVOKE文([3])に移ります。

仮に、[2]のUSE文が記述されていない場合、COBOLランタイムシステムがエラーを検出します。

10.2　FJCOL-EXCEPTIONクラス

説明

コレクションクラスライブラリの例外オブジェクトのクラスです。
当クラスのメソッドは、手続き部の宣言節部分に記述する例外手続き中で使用することができます。

解説

FJCOL-EXCEPTIONクラスは、FJBASEクラスを継承しています。

オブジェクトメソッド

メソッド名	機能概要
GET-CLASS-NAME	例外オブジェクトが発生したクラスライブラリのクラス名を返します
GET-METHOD-NAME	例外オブジェクトが発生したクラスライブラリのメソッド名を返します
GET-CODE	例外種別を返します
GET-MESSAGE	例外メッセージを返します

10.2.1　GET-CLASS-NAMEメソッド

説明

例外オブジェクトが発生したクラスライブラリのクラス名を返します。

記述形式

```
INVOKE EXCEPTION-OBJECT "GET-CLASS-NAME"
                USING CLASS-NAME
                RETURNING CLASS-NAME-LENGTH
```

引数

CLASS-NAME [属性: PIC X ANY LENGTH]
　例外が発生したクラス名を格納するデータ項目を指定します。

復帰値

CLASS-NAME-LENGTH [属性: PIC S9(9) COMP-5]
　例外が発生したクラス名の長さ(バイト数)が返却されます。

解説

例外オブジェクトが発生したクラスライブラリのクラス名およびクラス名の長さを返却します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  CLASS-NAME         PIC X(100).
01  CLASS-NAME-LENGTH  PIC S9(9) COMP-5.
                :
PROCEDURE DIVISION.
```

```
DECLARATIVES.
ERR SECTION.
    USE AFTER EXCEPTION FJCOL-EXCEPTION
    INVOKE EXCEPTION-OBJECT "GET-CLASS-NAME"                    …[1]
           USING CLASS-NAME RETURNING CLASS-NAME-LENGTH
    DISPLAY "Exception! CLASS="
                        CLASS-NAME(1:CLASS-NAME-LENGTH).
END DECLARATIVES.
                :
```

[例の説明]

[1]

GET-CLASS-NAMEメソッドを実行し、例外が発生したクラス名を獲得します。

10.2.2　GET-METHOD-NAMEメソッド

説明

例外オブジェクトが発生したクラスライブラリのメソッド名を返します。

記述形式

```
INVOKE EXCEPTION-OBJECT "GET-METHOD-NAME"
               USING METHOD-NAME
               RETURNING METHOD-NAME-LENGTH
```

引数

METHOD-NAME **[属性: PIC X ANY LENGTH]**

例外が発生したメソッド名を格納するデータ項目を指定します。

復帰値

METHOD-NAME-LENGTH **[属性: PIC S9(9) COMP-5]**

例外が発生したメソッド名の長さ(バイト数)が返却されます。

解説

例外オブジェクトが発生したクラスライブラリのメソッド名およびメソッド名の長さを返却します。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  METHOD-NAME         PIC X(100).
01  METHOD-NAME-LENGTH  PIC S9(9) COMP-5.
                :
PROCEDURE DIVISION.
DECLARATIVES.
ERR SECTION.
    USE AFTER EXCEPTION FJCOL-EXCEPTION
    INVOKE EXCEPTION-OBJECT "GET-METHOD-NAME"                   …[1]
           USING METHOD-NAME RETURNING METHOD-NAME-LENGTH
    DISPLAY "Exception! METHOD="
                        METHOD-NAME(1:METHOD-NAME-LENGTH).
```

```
END DECLARATIVES.
              :
```

[例の説明]

[1]

GET-METHOD-NAMEメソッドを実行し、例外が発生したメソッド名を獲得します。

10.2.3　GET-CODEメソッド

説明

例外種別を返します。

記述形式

```
INVOKE EXCEPTION-OBJECT "GET-CODE"
                RETURNING CODE
```

復帰値

CODE **[属性: PIC S9(9) COMP-5]**

例外種別が返却されます。

解説

例外オブジェクトが発生したときの例外条件を通知する例外種別が返却されます。

- 例外種別を宣言した登録集ファイル"FJCOSYMB.cbl"を提供しています。登録集ファイルを使用する場合は、以下のようにしてください。

  a- 環境部構成節特殊名段落のSYMBOLIC CONSTANT 句にCOPY文を記述し、登録集ファイルを指定してください。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    SYMBOLIC CONSTANT
    COPY FJCOSYMB.
    .
```

  b- 登録集ファイルが格納されているフォルダを "登録集ファイルのフォルダ" として指定してください。登録集ファイルのフォルダの指定方法については、"NetCOBOL 使用手引書"を参照してください。

使用例

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
              :
SPECIAL-NAMES.
    SYMBOLIC CONSTANT
    COPY FJCOSYMB.                                    …[1]
    .
DATA DIVISION.
WORKING-STORAGE SECTION.
```

```
01  EXCEPTION-CODE      PIC S9(9) COMP-5.
                :
PROCEDURE DIVISION.
DECLARATIVES.
ERR SECTION.
    USE AFTER EXCEPTION FJCOL-EXCEPTION
    INVOKE EXCEPTION-OBJECT "GET-CODE"                          …[2]
                    RETURNING EXCEPTION-CODE
    IF EXCEPTION-CODE = FJCOL-SCE-NOELEMENT THEN                …[3]
      DISPLAY "NOELEMENT Exception!"
    ELSE
      DISPLAY "Other Exception!"
    END-IF.
END DECLARATIVES.
                :
```

[例の説明]

[1]
例外種別が宣言されている登録集"FJCOSYMB.cbl"を取り込みます。

[2]
GET-CODEメソッドを実行し、例外種別を獲得します。

[3]
[2]で獲得した例外種別により例外処理を記述します。

10.2.4　GET-MESSAGEメソッド

説明

例外メッセージを返します。エラーの内容を表示するために使用します。

記述形式

```
INVOKE EXCEPTION-OBJECT "GET-MESSAGE"
                USING MESSAGE
                RETURNING MESSAGE-LENGTH
```

引数

MESSAGE **[属性: PIC X ANY LENGTH]**
例外メッセージを格納するデータ項目を指定します。

復帰値

MESSAGE-LENGTH **[属性: PIC S9(9) COMP-5]**
例外メッセージの長さ(バイト数)が返却されます。

解説

例外メッセージおよび例外メッセージの長さが返却されます。返却された例外メッセージは表示用だけに使用してください。

使用例

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MESSAGE-DATA         PIC X(100).
01  MESSAGE-DATA-LENGTH  PIC S9(9) COMP-5.
```

```
              :
PROCEDURE DIVISION.
DECLARATIVES.
ERR SECTION.
    USE AFTER EXCEPTION FJCOL-EXCEPTION
    INVOKE EXCEPTION-OBJECT "GET-MESSAGE"                    …[1]
                USING MESSAGE-DATA
                RETURNING MESSAGE-DATA-LENGTH
    DISPLAY "Exception! MESSAGE="                            …[2]
            MESSAGE-DATA(1:MESSAGE-DATA-LENGTH).
END DECLARATIVES.
              :
```

[例の説明]

[1]

GET-MESSAGEメソッドを実行し、例外メッセージを獲得します。

[2]

[1]で獲得した情報を出力します。

第11章　マルチスレッド

この章では、マルチスレッドについて説明します。

11.1　マルチスレッド環境下での動作

コレクションクラスライブラリは、他のマルチスレッドプログラムと組み合わせることにより、マルチスレッド環境下で動作可能です。

11.2　マルチスレッドプログラムの作成

コレクションクラスライブラリを利用したCOBOLプログラムを、翻訳オプションTHREAD(MULTI)を指定して構築します。
マルチスレッドプログラムの作成方法の詳細については、“NetCOBOL 使用手引書”を参照してください。

11.3　オブジェクトのスレッド間共有

ここでは、スレッド間で同じコレクションクラスオブジェクトを共有する方法について説明します。

スレッドからスレッドへコレクションクラスオブジェクトのオブジェクト参照値を受け渡すことにより、コレクションクラスオブジェクトをスレッド間で共有することができます。COBOLランタイムシステムは、コレクションクラスオブジェクトに対して、同期制御を行いません。このため、コレクションクラスオブジェクトをスレッド間で共有して利用する場合には、オブジェクトロック関数を使用してオブジェクト単位で同期制御を行う必要があります。

ただし、後述のスレッド間共有時に同時動作可能なメソッドだけ呼び出している場合にはロックを省略することができます。オブジェクトロック関数の詳細については、"NetCOBOL 使用手引書" の『スレッド同期制御サブルーチン』を参照してください。

ファクトリデータを介して、コレクションクラスオブジェクトをスレッド間で共有する例を以下に示します。

```
                                         ┌─────────────────────────────────┐
                                         │IDENTIFICATION DIVISION.         │
                                         │CLASS-ID. C1 INHERITS FJBASE.    │
○─初期化スレッド──────────────────┐    │    :                            │
│    :                               │    │FACTORY.                         │
│WORKING-STORAGE SECTION.            │    │DATA DIVISION.                   │
│01 OBJ1 USAGE OBJECT REFERENCE      │    │WORKING-STORAGE SECTION.         │
│              FJCOL-LINKED-LIST.    │    │01 SOBJ USAGE OBJECT REFERENCE   │
│    :                               │    │    FJCOL-LINKED-LIST PROPERTY.  │
│PROCEDURE DIVISION.                 │    │    :                            │
│  INVOKE FJCOL-LINKED-LIST "NEW"    │    │END FACTORY.                     │
│            RETURNING OBJ1.         │    │OBJECT.                          │
│    :                               │    │    :                            │
│  SET SOBJ OF C1 TO OBJ1. … [1]     │    │METHOD-ID. M1.                   │
│    :                               │    │    :                            │
└────────────────────────────────────┘    └─────────────────────────────────┘

    ○──────────────────────────────────────────────────────────────┐
  ○─┴────────────────────────────────────────────────────────────┐ │
○─┴────────────────────────────────────────────────────────────┐ │ │
│    :                                                         │ │ │
│WORKING-STORAGE SECTION.                                      │ │ │
│01  OBJ2            USAGE OBJECT REFERENCE                    │ │ │
│                      FJCOL-LINKED-LIST.                      │ │ │
│01  WAIT-TIME       PIC S9(9) COMP-5 VALUE -1.                │ │ │
│01  ERR-DETAIL      PIC 9(9) COMP-5.                          │ │ │
│01  RET-VALUE       PIC S9(9) COMP-5.                         │ │ │
│    :                                                         │ │ │
│PROCEDURE DIVISION.                                           │ │ │
│  SET OBJ2 TO SOBJ OF C1.                          … [2]      │ │ │
│  CALL "COB_LOCK_OBJECT" WITH C LINKAGE                       │ │ │
│                         USING BY REFERENCE OBJ2              │ │ │
│                               BY VALUE WAIT-TIME             │ │ │
│                               BY REFERENCE ERR-DETAIL        │ │ │
│                         RETURNING RET-VALUE.      … [3]      │ │ │
```

```
┌──────────────────────────────────────────────────┐
│                                                  │
│  INVOKE OBJ2 "ADD-LAST-ELEMENT" USING ...  … [4] │
│                                                  │
│  CALL "COB_UNLOCK_OBJECT" WITH C LINKAGE         │
│                           USING BY REFERENCE OBJ2│
│                                 BY REFERENCE ERR-DETAIL
│                           RETURNING  RET-VALUE.  … [5]
│                                                  │
└──────────────────────────────────────────────────┘

○―終了スレッド―――――――――――┐
│    :                        │
│WORKING-STORAGE SECTION.     │
│    :                        │
│PROCEDURE DIVISION.          │
│    :                        │
│  SET SOBJ OF C1 TO NULL. … [6]
│    :                        │
└─────────────────────────────┘
```

[図の説明]

[1]は、C1クラスのファクトリオブジェクトのプロパティメソッドを呼び出し、FJCOL-LINKED-LISTクラスのオブジェクトをファクトリデータに設定しています。
[2]は、C1クラスのファクトリオブジェクトのプロパティメソッドを呼び出し、ファクトリデータから[1]で設定されたFJCOL-LINKED-LISTクラスのオブジェクトを取得します。
[3]は、FJCOL-LINKED-LISTクラスのオブジェクトが複数のスレッドで同時に使用されないようにするため、オブジェクトロック関数を使用して、オブジェクトのロックを獲得します。このとき、すべてのスレッドで、同時動作可能なメソッドだけ呼び出している場合には、ロックを省略することができます。
[4]は、FJCOL-LINKED-LISTクラスのオブジェクトのADD-LAST-ELEMENTメソッドを呼び出し、処理を行っています。この処理は、[3]でロックを獲得したスレッドにより実行されます。
[5]は、オブジェクトのロックを解放しています。ロックの解放により、別のスレッドが[3]でロックを獲得できます。
[6]は、C1クラスのファクトリオブジェクトのプロパティメソッドを呼び出し、ファクトリデータに設定されているオブジェクトを削除しています。

- イテレータオブジェクト(FJCOL-ITERATORおよびFJCOL-LIST-ITERATOR)は、スレッド間で共有できません。

スレッド間共有時に同時動作可能なメソッド

コレクションオブジェクトをスレッド間で共有した場合の各メソッドの同時動作の可否は、そのメソッドの持つ性格に依存します。基本的な性格分けは以下のようになります。

- 参照系メソッド(データあるいはオブジェクトの参照だけ行い、追加・削除を行わないメソッド)は同時動作可能。
- 更新系メソッド(データあるいはオブジェクトの追加･削除を行うメソッド)は同時動作不可。

各メソッドごとの同時動作の可否を以下に示します。

○: 同時動作可能(ロック省略可能)
×: 同時動作不可(ロック省略不可)

● FJCOL-COLLECTIONクラス

メソッド名	同時動作	機能概要
CLEAR-COLLECTION	×	現在のコレクションを空にします
CLONE-COLLECTION	○	現在のコレクションの複製を作成します
CONTAINS-ELEMENT	○	現在のコレクションに指定した要素が含まれるか検査します
CONTAINS-ALL-ELEMENT	○	現在のコレクションに指定したコレクションの要素がすべて含まれるか検査します
CREATE-ITERATOR	○	現在のコレクションに対してイテレータオブジェクトを作成します
GET-SIZE	○	現在のコレクションの要素数を返します
IS-EMPTY	○	現在のコレクションが空か検査します

● FJCOL-LISTクラス

メソッド名	同時動作	機能概要
CREATE-LIST-ITERATOR	○	現在のリストに対してリストイテレータオブジェクトを作成します
EQUALS-LIST-COLLECTION	○	現在のリストと指定したリストが同等か検査します

● FJCOL-LINKED-LISTクラス

メソッド名	同時動作	機能概要
ADD-FIRST-ELEMENT	×	現在のリストの先頭に要素を追加します
ADD-LAST-ELEMENT	×	現在のリストの末尾に要素を追加します
GET-FIRST-ELEMENT	○	現在のリストの先頭の要素を返します
GET-LAST-ELEMENT	○	現在のリストの末尾の要素を返します
REMOVE-FIRST-ELEMENT	×	現在のリストの先頭の要素を削除します
REMOVE-LAST-ELEMENT	×	現在のリストの末尾の要素を削除します

● FJCOL-SETクラス

メソッド名	同時動作	機能概要
ADD-ELEMENT	×	現在のセットに要素を追加します
ADD-ALL-ELEMENT	×	現在のセットに指定したコレクションが持つ要素を追加します
EQUALS-SET-COLLECTION	○	現在のセットと指定したセットが同等か検査します
REMOVE-ELEMENT	×	現在のセットから要素を削除します
REMOVE-ALL-ELEMENT	×	現在のセットから指定したコレクションが持つ要素を削除します

● FJCOL-MAPクラス

メソッド名	同時動作	機能概要
CLEAR-COLLECTION	×	現在のマップを空にします
CLONE-COLLECTION	○	現在のマップの複製を作成します
CREATE-ITERATOR	○	現在のマップに対してイテレータオブジェクトを作成します
GET-SIZE	○	現在のマップのエントリ数を返します
IS-EMPTY	○	現在のマップが空か検査します

● FJCOL-ALPHANUMERIC-MAPクラス

メソッド名	同時動作	機能概要
CONTAINS-KEY	○	現在のマップに指定したキーが含まれるか検査します
CONTAINS-VALUE	○	現在のマップに指定した値（オブジェクト）が含まれるか検査します
EQUALS-MAP-COLLECTION	○	現在のマップと指定されたマップが同等か検査します
GET-VALUE	○	現在のマップから指定したキーに対する値（オブジェクト）を返します
PUT-ENTRY	×	現在のマップにエントリ（キーと値）を追加します
PUT-ALL-ENTRY	×	現在のマップに、指定したマップに含まれるすべてのエントリ（キーと値）を追加します
REMOVE-ENTRY	×	現在のマップから指定したキーに対するエントリ（キーと値）を削除します

● FJCOL-ITERATORクラス（スレッド間共有できません）

メソッド名	同時動作	機能概要
HAS-NEXT-ELEMENT	×	イテレータオブジェクトが示す要素があるか検査します
NEXT-ELEMENT	×	イテレータオブジェクトが示す要素を返します
REMOVE-ELEMENT	×	イテレータオブジェクトが直前に返した要素を削除します

● FJCOL-LIST-ITERATORクラス（スレッド間共有できません）

メソッド名	同時動作	機能概要
ADD-ELEMENT	×	指標の値が示す位置に要素を追加します
HAS-NEXT-ELEMENT	×	直後にNEXT-ELEMENTメソッドを実行した場合に返却する要素があるか検査します
HAS-PREVIOUS-ELEMENT	×	直後にPREVIOUS-ELEMENTメソッドを実行した場合に返却する要素があるか検査します
NEXT-ELEMENT	×	指標の値が示す要素を返し、指標の値を＋１します
NEXT-INDEX	×	指標の値を返します
PREVIOUS-ELEMENT	×	指標の値を－１してから、指標の値が示す要素を返します
PREVIOUS-INDEX	×	指標の値より－１した値を返します
REMOVE-ELEMENT	×	直前に操作した要素を削除します
SET-ELEMENT	×	直前に操作した要素を変更します

● FJCOL-ALPHANUMERIC-MAP-ENTRYクラス

メソッド名	同時動作	機能概要
EQUALS-ENTRY	○	現在のエントリと指定されたエントリが同等か検査します
GET-KEY	○	現在のエントリに対するキーを返します
GET-VALUE	○	現在のエントリに対する値(オブジェクト)を返します
SET-VALUE	×	現在のエントリに対する値(オブジェクト)を指定された値(オブジェクト)に置き換えます

● FJCOL-EXCEPTIONクラス

メソッド名	同時動作	機能概要
GET-CLASS-NAME	○	例外オブジェクトが発生したクラスライブラリのクラス名を返します
GET-METHOD-NAME	○	例外オブジェクトが発生したクラスライブラリのメソッド名を返します
GET-CODE	○	例外種別を返します
GET-MESSAGE	○	例外メッセージを返します

● 同時動作可能なメソッド呼び出しであっても、他のスレッドで同時動作不可のメソッド呼び出しが存在する場合には、必ずロックを行ってください。

● スレッド間で共有するコレクションオブジェクトに対してスレッド固有のイテレータオブジェクトを作成することができます。
この時、コレクションオブジェクトに対して更新系メソッドを呼び出さないのであれば、通常通りの操作が可能です。しかし、コレクションオブジェクトに対して更新系のメソッド呼び出しを行うためにロックをしても、このロックは各イテレータオブジェクトに対しては有効ではありません。そのため、コレクションオブジェクトの更新系メソッドとイテレータオブジェクトのメソッドが同時に実行され、誤動作する可能性があります。
よって、コレクションオブジェクトに対する更新系メソッド呼び出しは、各イテレータオブジェクトにNULLをセットしてイテレータオブジェクトをすべて削除してから行うようにしてください。

第12章　Unicode

この章では、Unicodeについて説明します。

12.1　Unicode環境下での動作

コレクションクラスライブラリは、他のUnicodeプログラムと組み合わせることにより、Unicode環境下で動作可能です。

12.2　Unicodeプログラムの作成

コレクションクラスライブラリを利用したCOBOLプログラムを、翻訳オプションRCS(UCS2)を指定して構築します。

付録A 例外種別一覧

ここでは、コレクション例外クラスのGET-CODEメソッドにより獲得できる例外種別について説明します。

- FJCOL-SCE-UNSUPPORTED

  呼び出されたメソッドはサポートされていません。

  ［意味］

  サポートされていないメソッドを実行しました。

  ［プログラマの処置］

  例外が発生したメソッドを実行しないようにしてください。

- FJCOL-SCE-NOELEMENT

  対象となる要素がありません。

  ［意味］

  処理対象となる要素がありません。空のコレクションに対して要素の取り出しを行ったり、イテレータオブジェクトで最後まで要素を読み込んだ状態で次の要素を読みにいった場合などに発生します。

  ［プログラマの処置］

  コレクションに登録されている要素の数やイテレータオブジェクトの状態を確認し、正しく修正してください。

- FJCOL-SCE-INDEX

  指標の値が範囲内ではありません。

  ［意味］

  リストイテレータオブジェクトを作成する際の指標値が範囲外の値になっています。リストイテレータオブジェクト作成時に指定する指標の値は１から要素数＋１までの範囲です。

  ［プログラマの処置］

  指標の値を確認し、正しく修正してください。

- FJCOL-SCE-INVALID-ITERATOR

  イテレータオブジェクトが無効です。

  ［意味］

  指定したイテレータオブジェクトは使用できません。

  ［プログラマの処置］

  イテレータオブジェクトを使用したい場合は、再度、イテレータオブジェクトを作成してください。

- FJCOL-SCE-ILLEGAL-STATE

  実行されるメソッドの順番に誤りがあります。

  ［意味］

  メソッドの実行順序に誤りがあるため、メソッドが実行できません。イテレータオブジェクト作成直後に、REMOVE-ELEMENTメソッドを実行した場合などに発生します。

  ［プログラマの処置］

  メソッドの実行順序を確認し、正しく修正してください。

- FJCOL-SCE-ILLEGAL-PARAMETER

引数の値が正しくありません。
［意味］
　　例外が発生しているメソッドの引数に不当な値が指定されています。
［プログラマの処置］
　　例外が発生しているメソッドの引数の値を確認し、正しく修正してください。

● FJCOL-SCE-INTERNAL-ERROR
内部エラーです。
［意味］
　　コレクションクラスライブラリの障害と考えられます。
［プログラマの処置］
　　技術員（ＳＥ）に連絡してください。

索引

FUJITSU

このマニュアルはエコマーク認定の再生紙を使用しています。