# Basic Concepts I

Example ‑ Setting variable X to a value with the Set command

Set X=12

Set X="ABC"

Example ‑ Write command displays the value of variables

Set X=12

Write X

12

Set X="ABC"

Write X

ABC

Example ‑ Write command with carriage return line feed

Set X=12

Write !,X

;carriage return and line feed inserted here

12

Example ‑ Write command displays all variables

Set X=12

Set Y=13

Set Z=14

W ;Write command with no parameters

X=12

Y=13

Z=14

Example ‑ Multiple commands

Set X=12,Y=13,Z=14 ;single Set command with variables separated by commas

Write !,X,!,Y,!,Z ;single Write command with variables

;separated by commas

12

13

14

Example ‑ If command with numeric operands

Set X=12

If X=12 Set X=13

Write X

13

Example ‑ If command with alphanumeric operands

Set X="ABC"

If X="ABC" Set X="XYZ"

Write X

XYZ

Example ‑ Structured Code with If command

Set X=12

If (X=12) {Set X=13}

Write X

13

Example ‑ Plus sign, set the variable X to a value of +12

Set X=+12

Write X

12

Example ‑ Plus sign, set the variable X to a value of +"ABC"

Set X=+"ABC"

Write X

0

Example ‑ Plus sign, set the variable X to "ABC" and write +X

Set X="ABC"

Write +X

0

Write X

ABC

Example ‑ Set the variable X to "ABC", if +X equals 0, write X

Set X="ABC"

If +X=0 Write X

ABC

Example ‑ Kill a variable

Set X="ABC"

Kill X

Write X

<UNDEFINED>

Example ‑ Kill all variables

Set X=12,Y=13,Z=14 ;set variables

W ;write all variables

X=12

Y=13

Z=14

K ;kill all variables

W ;write all variables, but none exist

<>

## Exercises on Set, Write, If, Plus and Kill commands

Write a line of COS code that will accomplish each of the following:

* Set the variable X to the value of "ABC".
* Set the variable X to the value of 12.
* Display the value of X.
* Set the value of X to 13 if the current value of X is 12.
* Set the value of X back to 12 if the current value of plus-X is 13.
* Delete the X variable.

Example ‑ Answers to Exercises

Set the variable X to the value of "ABC"

Set X="ABC"

Set the variable X to the value of 12

Set X=12

Display the value of X

Write X

Set the value of X to 13 if the current value of X is 12

If X=12 Set X=13

Set the value of X back to 12 if the current value of plus-X is 13

If +X=13 Set X=12

Delete the X variable

Kill X

Example ‑ Concatenating two Variables

Set X="My dog's name is"

Set Y="Teddy."

Set DOG=X\_" "\_Y

Write DOG

My dog's name is Teddy.

Example ‑ Concatenate versus the plus sign

Write "My dog's name is "\_"Teddy." ;concatenation used properly

My dog’s name is Teddy

Write "My dog's name is "+"Teddy." ;plus sign used improperly

0

Write 1\_1 ;concatenation used improperly

11

Write 1+1 ;plus sign used properly

2

Example ‑ Operator Precedence Comparison

Write 5\*7+6 ;Operator Precedence in Caché same as in Mathematics

41

Write 4+6\*10/5 ;Caché Operator Precedence 4+6=10, 10\*10=100, 100/5=20

20

Write 4+(6\*10/5) ;Mathematical Operator Precedence: 4+(60/5)

16

Example ‑ For Loop command

FOR VARIABLE=STARTVALUE:INCREMENTALVALUE:ENDVALUE CODE

Example ‑ For Loop command, English interpretation

Step 1: Set the VARIABLE to the STARTVALUE,

Step 2: If VARIABLE is not more than the ENDVALUE,

Write again, otherwise stop

Step 3: Increment the VARIABLE by the INCREMENTALVALUE

Step 4: Go to Step 2

Example ‑ For Loop command continued

For I=1:1:3 Write I

Example ‑ For Loop command interprets - "For I=1:1:3 Write I"

Set variable I to 1,

Variable I is not more than the end value (3), so Write I,

Increment variable I by 1, it is now 2,

Variable I is not more than the end value (3),

so Write I again,

Increment variable I by 1, it is now 3,

Variable I is not more than the end value (3),

so Write I again,

At this point the loop will stop because attempting

to incrementing variable I

will cause it to exceed the end value of 3,

End of For loop command.

Example ‑ For Loop command output

For I=1:1:3 Write I,!

1

2

3

Example ‑ For Loop command example

For I=4:1:7 Write I,!

4

5

6

7

Example ‑ For Loop command example

For I=2:2:8 Write I,!

2

4

6

8

Example ‑ For Loop command example

For I=7:-1:4 Write I,!

7

6

5

4

Example ‑ For Loop command, Second Format

FOR VARIABLE="VALUE1","VALUE2","VALUE3" Write !,VARIABLE

Example ‑ Output from

Example 1‑27

FOR VARIABLE="VALUE1","VALUE2","VALUE3" Write !,VARIABLE

VALUE1

VALUE2

VALUE3

Example ‑ For Loop command, Third Format

FOR Read VARIABLE WRITE !,VARIABLE If VARIABLE="END" QUIT

;notice the two spaces after the "FOR"

Example ‑ For Loop command, Third Format, output

FOR Read VARIABLE WRITE !,VARIABLE If VARIABLE="END" QUIT

First

Second

Third

END

Example ‑31 Halt command

Halt

“It is not how much you know, but how well you communicate it to others.”

# Basic Concepts II

Example ‑ Labels and Executable code lines

START ;Label

Set X=5 ;Executable Code line

Example ‑ Label and Executable code on the same line

START Set X=5

Example ‑ Routine Header Line

ROUTINEA ;name of the routine on the first line

Start ;Label

Set X=1 ;Code

.

.

Quit

Example ‑ Comment Lines

PROC ;comments may follow a label

;

; Everything that follows the ";" is a comment

;

Set X=1 ; comments may follow code

PROC2 //comments may follow a label

//

// Everything that follows the "//" is a comment

//

Set X=1 // comments may follow code

/\* starts a multi line comment

more comments

more comments

\*/ end a multi-line comment

Example ‑ Do and Quit commands

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

START

Write !,"At Start label"

Do PROC ;Do command

Write !,"At Start:Quit"

Quit ;Quit command

PROC

Write !,"At Proc label"

Set X=5

Write !,"At Proc:Quit"

Quit ;Quit command

Example ‑ Output from

Example 2‑5

At Start label

At Proc label

At Proc:Quit

At Start:Quit

Example ‑ Goto command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

START

Write !,"At Start label"

Goto PROC

Write !,"At Start:Quit"

Quit

PROC

Write !,"At Proc label"

Set X=5

Write !,"At Proc:Quit"

Quit

Example ‑ Output from

Example 2‑7

At Start label

At Proc label

At Proc:Quit

Example ‑ Inline Do

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For I=1:1:3 Do

. Write !,I

. Write !,I+10

Write !,"End of For Loop"

Example ‑ Inline Do output from

Example 2‑9

1

11

2

12

3

13

End of For Loop

Example ‑ Nested Inline Do commands

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For I=1:1:3 Do

. Write !,"First Level: ",I

. For II=1:1:3 Do

. . Write !," Second Level: ",I+II

Write !,"End of For Loop"

Example ‑ Output of

Example 2‑11

First Level: 1

Second Level: 2

Second Level: 3

Second Level: 4

First Level: 2

Second Level: 3

Second Level: 4

Second Level: 5

First Level: 3

Second Level: 4

Second Level: 5

Second Level: 6

End of For Loop

Example ‑ Inline Do and For Loop commands with Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For I=1:1:3 {

Write !,"First Level: ",I

For II=1:1:3 {

Write !," Second Level: ",I+II

}

}

Write !,"End of For Loop"

Example ‑ Executing a routine

ROUTINEA ;name of the routine on the first line

Do ^ROUTINEB ;execution jumps to ^ROUTINEB

Quit

Example ‑ Executing a LABEL in a routine

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

ROUTINEA

;

Do PROC ;execution jumps to PROC

; code

; code

Quit

PROC ;

; code

; code

Quit ;execution jumps back to the line following "Do PROC"

Example ‑ Executing a LABEL in another routine

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal. ROUTINEA and ROUTINEB need to be in separate routines.

ROUTINEA

;

Do PROC^ROUTINEB ;execution jumps to PROC in ^ROUTINEB

; code

; code

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

ROUTINEB

;

PROC

; code

; code

Quit ;execution jumps to the line following PROC in ^ROUTINEA

Example ‑ Routines and Variables

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal. ROUTINEA, ROUTINEB, and ROUTINEC need to be in separate routines.

ROUTINEA

Set X="A",Y="A",Z="A" ;X,Y and Z all set to "A"

Write !,"ROUTINEA – X: ",X

Write !,"ROUTINEA – Y: ",Y

Write !,"ROUTINEA – Z: ",Z

Do ^ROUTINEB

Write !!,"ROUTINEA – X: ",X

Write !,"ROUTINEA – Y: ",Y

Write !,"ROUTINEA – Z: ",Z

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

ROUTINEB

Set Y="B",Z="B" ;Y and Z set to "B"

Write !!,"ROUTINEB – X: ",X

Write !,"ROUTINEB – Y: ",Y

Write !,"ROUTINEB – Z: ",Z

Do ^ROUTINEC

Write !!,"ROUTINEB – X: ",X

Write !,"ROUTINEB – Y: ",Y

Write !,"ROUTINEB – Z: ",Z

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

ROUTINEC

Set Z="C" ;Z set to "C"

Write !!,"ROUTINEC – X: ",X

Write !,"ROUTINEC – Y: ",Y

Write !,"ROUTINEC – Z: ",Z

Quit

Example ‑ Output from running the routines in Example 2‑17

Do ^ROUTINEA

ROUTINEA – X: A

ROUTINEA – Y: A

ROUTINEA – Z: A

ROUTINEB – X: A

ROUTINEB – Y: B

ROUTINEB – Z: B

ROUTINEC – X: A

ROUTINEC – Y: B

ROUTINEC – Z: C

ROUTINEB – X: A

ROUTINEB – Y: B

ROUTINEB – Z: C

ROUTINEA – X: A

ROUTINEA – Y: B

ROUTINEA – Z: C

Example ‑ Parameter Passed by Value, ROUTINEA and ROUTINEB

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal. ROUTINEA and ROUTINEB need to be in separate routines.

ROUTINEA

Write !!,"ROUTINEA Starting"

Set PARAM1="Value for Param1"

Set PARAM2="Value for Param2"

Write !,"ROUTINEA-PARAM1: ",PARAM1

Write !,"ROUTINEA-PARAM2: ",PARAM2

Do PROC^ROUTINEB(PARAM1,PARAM2) ;call passing PARAM1, PARAM2

Write !!,"ROUTINEA-PARAM1: ",PARAM1

Write !,"ROUTINEA-PARAM2: ",PARAM2

Write !,"ROUTINEA Ending"

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

ROUTINEB

;

PROC(PAR1,PAR2)

Write !!,"ROUTINEB Starting"

Write !,"ROUTINEB-PAR1: ",PAR1

Write !,"ROUTINEB-PAR2: ",PAR2

Set PAR2="New value for PAR2" ;ROUTINEB changes PAR2

Write !!,"ROUTINEB-PAR1: ",PAR1

Write !,"ROUTINEB-PAR2: ",PAR2

Write !,"ROUTINEB Ending"

Quit

Example ‑ Running ROUTINEA (parameters passed by value)

Do ^ROUTINEA

ROUTINEA Starting

ROUTINEA-PARAM1: Value for Param1

ROUTINEA-PARAM2: Value for Param2

ROUTINEB Starting

ROUTINEB-PAR1: Value for Param1

ROUTINEB-PAR2: Value for Param2

ROUTINEB-PAR1: Value for Param1

ROUTINEB-PAR2: New value for PAR2 ;ROUTINEB changes PAR2

ROUTINEB Ending

ROUTINEA-PARAM1: Value for Param1

ROUTINEA-PARAM2: Value for Param2

ROUTINEA Ending

Example ‑ Parameter Passed by Reference, ROUTINEA and ROUTINEB

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal. ROUTINEA and ROUTINEB need to be in separate routines.

ROUTINEA

Write !!,"ROUTINEA Starting"

Set PARAM1="Value for Param1"

Set PARAM2="Value for Param2"

Write !,"ROUTINEA-PARAM1: ",PARAM1

Write !,"ROUTINEA-PARAM2: ",PARAM2

Do PROC^ROUTINEB(.PARAM1,.PARAM2) ;Period or dot before PARAMS

Write !!,"ROUTINEA-PARAM1: ",PARAM1

Write !,"ROUTINEA-PARAM2: ",PARAM2

Write !,"ROUTINEA Ending"

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

ROUTINEB

;

PROC(PAR1,PAR2)

Write !!,"ROUTINEB Starting"

Write !,"ROUTINEB-PAR1: ",PAR1

Write !,"ROUTINEB-PAR2: ",PAR2

Set PAR2="New value for PAR2"

Write !!,"ROUTINEB-PAR1: ",PAR1

Write !,"ROUTINEB-PAR2: ",PAR2

Write !,"ROUTINEB Ending"

Quit

Example ‑ Running ROUTINEA (parameters passed by reference)

Do ^ROUTINEA

ROUTINEA Starting

ROUTINEA-PARAM1: Value for Param1

ROUTINEA-PARAM2: Value for Param2

ROUTINEB Starting

ROUTINEB-PAR1: Value for Param1

ROUTINEB-PAR2: Value for Param2

ROUTINEB-PAR1: Value for Param1

ROUTINEB-PAR2: New value for PAR2

ROUTINEB Ending

ROUTINEA-PARAM1: Value for Param1

ROUTINEA-PARAM2: New value for PAR2 ;new value passed back

ROUTINEA Ending

Table ‑ Formats for Call Routines

|  |  |
| --- | --- |
| Code to Call | Descriptions |
| Do ^Routine | Calling a routine |
| Do ^Routine(Param) | Calling a routine with a parameter |
| Write $$^Function | Calling a function |
| Set X=$$^Function(Param) | Calling a function with a parameter |
| Do Tag^Routine | Calling a routine at a tag |
| Do Tag^Routine(Param) | Calling a routine at a tag with a parameter |
| Write $$Tag^Function | Calling a function at a tag |
| Write $$Tag^Function(Param) | Calling a function at a tag with a parameter |

Example ‑ Array Example

Set PEOPLE("DAVID")="DATA ABOUT DAVID"

Set PEOPLE("SUSAN")="DATA ABOUT SUSAN"

Set PEOPLE("MICHAEL")="DATA ABOUT MICHAEL"

Set PEOPLE("AMY")="DATA ABOUT AMY"

Table ‑ Differences between Global Arrays, Global variable, Local Arrays and Local Variables

|  |  |  |
| --- | --- | --- |
| What | Description | |
| Global Arrays | Multi-dimensional using subscripts | Persistent, permanent, reside on disk |
| Global Variables | One-dimensional, Scalar, no subscripts | Persistent, permanent, reside on disk |
| Local Arrays | Multi-dimensional using subscripts | Temporary, reside only in memory |
| Local Variables | One-dimensional, Scalar, no subscripts | Temporary, reside only in memory |

### 

Example ‑ Setting Globals, Local Variables and Arrays

Set ^X="ABC" ;set a Global variable

Set ^X(1,2)="ABC" ;set a Global array

Set X="ABC" ;set a local variable

Set X(1,2)="ABC" ;set a local array

“Crashed and burning on the learning curve.” – Dick Martel

# System-Supplied Functions I

Example ‑ $Length Example

Write $L("Page Title")

10

Set X="Page Title"

Write $L(X)

10

Example ‑ $Length to center text on a line

Set X="Page Title"

Write ?(80-$L(X)/2),X

Page Title

Example ‑ Delimited String of text

Jack Sampson^123 Any Street^Any Town^USA^12045

Example ‑ $Length to find the number of pieces in a string

Set X="This is a test"

Write $L(X," ")

4

Example ‑ $Length to write each piece in a delimited string

Set String="Jack Sampson^123 Any Street^Any Town^USA^12045"

For I=1:1:$L(String,"^") Write !, $P(String,"^",I)

Jack Sampson

123 Any Street

Any Town

USA

12045

Example ‑ $Extract selects a substring from a string

Set X="My dog Spot"

Write !,$E(X,1,2)

My

Example ‑ $Extract selects a substring from a string

Set X="My dog Spot"

Write !,$E(X,8,11)

Spot

Example ‑ $Extract with no second and third parameters

Set X="My dog Spot"

Write !,$E(X) ;with no second or third parameters

M ;the first character is returned

Write !,$E(X,1,1)

M ;the first character is returned

Example ‑ $Extract with no third parameter, becomes the same as the second parameter

Set X="My dog Spot"

Write !,$E(X,4) ;with no third parameter,

d;the third becomes the same as the second

Write !,$E(X,4,4)

d;the fourth character is returned

Example ‑ $Extract sets a variable

Set X="My dog Spot"

Set Y=$E(X,8,11)

Write !,Y

Spot

Example ‑ $Extract sets a substring in a string

Set X="My dog Spot"

Set $E(X,8,11)="Fred"

Write !,X

My dog Fred.

Example ‑ $Extract reformats a name

Set NAME1="John Doe"

Set NAME2=$E(NAME1,6,8)\_", "\_$E(NAME1,1,4)

Write !,NAME2

Doe, John

Example ‑ $Extract reformats a date

Set DATE1="20092205" ;A date string in the form "YYYYMMDD"

Set DATE2=$E(DATE1,7,8)\_"/"\_$E(DATE1,5,6)\_"/"\_$E(DATE1,3,4)

Write !,DATE2

05/22/09

Example ‑ $Extract deletes part of a variable string

Set X="111111111122222222223333333333"

Set $E(X,1,25)= ""

Write X

33333

Example ‑ $Find finds a "T"

Set X="This is a test"

Write $F(X,"T")

2

Example ‑ $Find locates "is"

Set X="This is a test"

Write $F(X,"is")

5

Example ‑ $Find finds "ABC"

Set X="This is a test"

Write $F(X,"ABC")

0

Example ‑ $Find locates "t" starting at the fifth position

Set X="This is a test"

Write $F(X,"t",5)

12

Example ‑ $Find finds the second occurrence of "is"

Set X="This is a test"

Write $F(X,"is",$F(X,"is"))

8

Example ‑ $Find locates the third occurrence of "is"

Set X="This is a test"

Write $F(X,"is",$F(X,"is",$F(X,"is")))

0

### Exercises on $Length, $Extract and $Find

Here is your chance to try out what you have learned. The answers are shown after the questions. Assume the following:

Assume X="My dog's name is Teddy"

Write the code that:

* Displays the length of X
* Displays the number of pieces in X using space as the delimiter
* Displays the first two characters in X
* Displays characters four through eight in X
* Displays the starting position of "Teddy" in X
* Replace the name "Teddy" with the name "Trixie" in X

Example ‑ Answers to Exercises on $Length, $Extract and $Find

Displays the length of X

Write !,$L(X)

22

Displays the number of pieces in X using space as the delimiter

Write !,$L(X," ")

5

Displays the first two characters in X

Write !,$E(X,1,2)

My

Displays characters four through eight in X

Write !,$E(X,4,8)

dog's

Displays the starting position of "Teddy" in X

Write !,$F(X,"Teddy")-$L("Teddy")

18

Replace the name "Teddy" with "Trixie" in X

Set $E(X,18,22)="Trixie"

Write !,X

My dog's name is Trixie

The next line is a more advanced method of replacing "Teddy" with "Trixie"

when the position of "Teddy" is not known.

Set $E(X,$F(X,"Teddy")-$L("Teddy"),$F(X,"Teddy"))="Trixie"

Write !,X

My dog's name is Trixie

Or using the variable DOG for "Teddy":

Set DOG="Teddy"

Set $E(X,$F(X,DOG)-$L(DOG),$F(X,DOG))="Trixie"

Write !,X

My dog's name is Trixie

Example ‑ Replace variable A with variable B

Set X="TEDDY IS A GREAT DOG"

Set A="GREAT"

Set B="BAD"

Set $E(X,$F(X,A)-$L(A),$F(X,A)-1)=B

Write X

TEDDY IS A BAD DOG

Example ‑ Separating a name

Set NAME1="John Q. Public"

Set FNAME=$E(NAME1,1,$F(NAME1," ")-2)

Set MI=$E(NAME1,$F(NAME1,".")-2,$F(NAME1,"."))

Set LNAME=$E(NAME1,$F(NAME1,".")+1,$L(NAME1))

Write !,FNAME

John

Write !,MI

Q.

Write !,LNAME

Public

Example ‑ $Replace to replace a string

Set TargetString="My dog is ugly"

Set OldString="ugly"

Set NewString="smart"

Set ChangedString=$Replace(TargetString,OldString,NewString)

Write !,ChangedString

My dog is smart

Example ‑ $Replace to replace a string starting at a specific location

Set TargetString="My dog is ugly and your dog is not"

Set OldString="dog"

Set NewString="cat"

Set ChangedString=$Replace(TargetString,OldString,NewString,16)

Write !,ChangedString

and your cat is not

Example ‑ $Replace to replace one occurrence of a string

Set TargetString="My dog is ugly and your dog is not"

Set OldString="dog"

Set NewString="cat"

Set ChangedString=$Replace(TargetString,OldString,NewString,,1)

Write !,ChangedString

My cat is ugly and your dog is not

Example ‑ $Replace with case sensitivity

Set TargetString="My dog is ugly and your dog is not"

Set OldString="DOG"

Set NewString="cat"

Set ChangedString=$Replace(TargetString,OldString,NewString,,,0)

Write !,ChangedString

My dog is ugly and your dog is not

Example ‑ $Replace with case insensitivity

Set TargetString="My dog is ugly and your dog is not"

Set OldString="DOG"

Set NewString="cat"

Set ChangedString=$Replace(TargetString,OldString,NewString,,,1)

Write !,ChangedString

My cat is ugly and your cat is not

Example ‑ $Translate to remove a comma

Set X="123,456.00"

Set X=$TR(X,",") ;remove comma

Write !,X

123456.00

Example ‑ $Translate to remove a comma and period

Set X="123,456.00"

Set X=$TR(X,",.") ;remove commas and period

Write !,X

12345600

Example ‑ $Translate with three parameters, reformates a date

Set DATE="01/01/2003"

Set DATE=$TR(DATE,"/","-") ;replaces "/" with "-"

Write !,DATE

01-01-2003

Example ‑ Convert a string to Uppercase

Set X="abc"

Set X=$ZCVT(X,"U")

Write !,X

ABC

Example ‑ Convert a string to Lowercase

Set X="ABC"

Set X=$ZCVT(X,"L")

Write !,X

abc

Example ‑ Strip leading spaces

Set X=" ABC DEF "

Set X=$ZSTRIP(X,"<W")

Write !,"-",X,"-"

-ABC DEF -

Example ‑ Strips trailing spaces

Set X=" ABC DEF "

Set X=$ZSTRIP(X,">W")

Write !,"-",X,"-"

- ABC DEF-

Example ‑ Strip leading and trailing spaces

Set X=" ABC DEF "

Set X=$ZSTRIP(X,"<>W")

Write !,"-",X,"-"

-ABC DEF-

Example ‑ Strip all spaces

Set X=" ABC DEF "

Set X=$ZSTRIP(X,"\*W")

Write !,"-",X,"-"

-ABCDEF-

Example ‑ $Zstrip Strip Characters

Set X="ABC123DEF456"

Set X=$ZSTRIP(X,"\*","123")

Write !,"-",X,"-"

-ABCDEF456-

*“Few things are harder to put up with than the annoyance of a good example.” – Mark Twain*

# System-Supplied Functions II

Table ‑ $Data Table of Returned Values

|  |  |  |
| --- | --- | --- |
| What $Data Returns | Does the Array Node  have Value? | Does the Array Node  Have Descendants? |
| 0 | No | No |
| 1 | Yes | No |
| 10 | No | Yes |
| 11 | Yes | Yes |

Example ‑ Local Array

Set A(1)="data" ; this node has a value but no descendants

Set A(2)="" ; this node has a null value but no descendants

Set A(3,1)="data" ; this node has a value and by implication is a

; descendant of the nonexistent node A(3)

Set A(4)="data" ; this node has a value and descendants

Set A(4,1)="data" ; this node is a descendant and has descendants

Set A(4,1,2)="data" ; this node is a descendant

Example ‑ Array node that has a value but no descendants

Write !,$D(A(1))

1

Example ‑ Array node that has a null value and no descendants

Write !,$D(A(2))

1

Example ‑ Array node that has no value but has descendants

Write !,$D(A(3))

10

Example ‑ Array node that has a value and descendants

Write !,$D(A(4))

11

Example ‑ Array node that has a value but no descendants

Write !,$D(A(3,1))

1

Example ‑ Array node that has a value and descendants

Write !,$D(A(4,1))

11

Example ‑ Array node does not exist and has no descendants.

Write !,$D(A(5))

0

Example ‑ $Data with the If command

Set A(1)="data"

Set A(2)=""

Set A(3,1)="data"

Set A(4)="data"

Set A(4,1)="data"

If $D(A(1)) Write "True" ;returns a 1, node has value but no descendants

True

If $D(A(2)) Write "True" ;returns a 1, node has value but no descendants

True

If $D(A(3)) Write "True" ;returns 10, node has no value but has descendants

True

If $D(A(3,1)) Write "True" ;returns 1, node has value but no descendants

True

If $D(A(4)) Write "True" ;returns 11, node has value and descendants

True

If $D(A(5)) Write "True" ;returns 0, node has no value nor descendants

<>

### Exercises on $Data Exercises

Here is your chance to try out what you have learned. Try to guess what each of the *Write*commands will produce. The answers are shown after the questions.

* Kill A Write $D(A)
* If $D(A) Write "hit"
* Set A="" Write $D(A)
* Set A(1)="data" Write $D(A)
* Kill A Set A(1)="data" Write $D(A)
* If $D(A) Write "hit"

Example ‑ Answers

Kill A

Write $D(A) ;variable does not exist

0

If $D(A) Write "Hit" ;variable does not exist

<>

Set A=""

Write $D(A) ;variable exists but with a null value

1

Set A(1)="data" ;both node and descendants exist

Write $D(A)

11

Kill A

Set A(1)="data"

Write $D(A) ;node has no value but does have a descendant

10

If $D(A) Write "Hit" ;node has no value but does have descendants

Hit

Example ‑ $Get returns the variables' value

Set X="ABC"

Write !,$G(X)

ABC

Example ‑ $Get returns null

Set Y=""

Write $G(Y) ; Y has a value of null

<>

Kill Z

Write $G(Z) ; Z does not exist

<>

Example ‑ $Get with a default parameter

Set X=1

Write !, $G(X,"DEF") ; If X has null as a value, the default is not used

1

Set X=""

Write !, $G(X,"DEF") ; If X has null as a value, the default is not used

<>

Kill X

Write !, $G(X,"DEF") ; If X does not exist, then the default is used

DEF

Example ‑ $FNumber inserts commas in a number

Write $FN(1234,",")

1,234

Set X=1234

Write $FN(X,",")

1,234

Example ‑ $FNumber inserts commas in a negative number

Set X=-1234

Write $FN(X,",")

-1,234

Example ‑ $FNumber inserts a comma and a decimal point

Set X=123456

Write $FN(X,",",2) ; 2 – indicates 2 decimal places

123,456.00

Example ‑ $FNumber inserts commas and rounds a number

Set X=123456.55

Write $FN(X,",",0) ; 0 – indicates no decimal places

123,457

Example ‑ $Justify, Ten Character field, right justified

Set X=123657

Write $J(X,10)

123657

Example ‑ $Justify, Ten Character field, right justified with 2 decimal places

Set X=123657

Write $J(X,10,2)

123657.00

Example ‑ $Justify, Ten Character field, alpha text, right justified

Set X="ABCDEF"

Write $J(X,10)

ABCDEF

Example ‑ $Justify produces tabular output

Set X1=123

Set X2=65432

Set X3=546.44

Set X4=0505.22

Write $J(X1,10,2)

Write $J(X2,10,2)

Write $J(X3,10,2)

Write $J(X4,10,2)

Write $J("----------",10)

Write $J((X1+X2+X3+X4),10,2)

123.00

65432.00

546.44

505.22

----------

66606.66

Example ‑ Format 12 digits, commas, right justified, 2 decimal

Set X=12345

Write $J($FN(X,",",2),12)

12,345.00

Example ‑ Format 12 digits, commas, right justified, 2 decimal, negative

Set X=-12345

Write $J($FN(X,"P,",2),12)

(12,345.00)

Example ‑ Format 12 digits, commas, right justified, 2 decimal, trailing minus sign

Set X=-12345

Write $J($FN(X,"T,",2),12)

12,345.00-

### Exercises on $FNumber and $Justify

Here is your chance to try out what you have learned. The answers are shown after the questions.

* Write this number with commas inserted

Set X=1234

* Write this number with commas and two decimal places

Set X=123456

* Write this number with no decimal places and round

Set X=123456.55

* Write this number right justified in a 10 character field with 2 decimal places

Set X=657

* Combine $Justify and $FNumber to give 12,345.00, right-justified, in a 12 character field

Set X=12345

### Answer to Exercises on $FNumber and $Justify

Example ‑ $FNumber and $Justify Exercises answers

Write this number with commas inserted

Set X=1234

Write $FN(X,",")

1,234

Write this number with commas and two decimal places

Set X=123456

Write $FN(X,",",2)

123,456.00

Write this number with no decimal places and round

Set X=123456.55

Write $FN(X,",",0)

123,457

Write this number right justified in a 10 character field with 2 decimal places

Set X=657

Write $J(X,10,2)

657.00

Combine $Justify and $FNumber to give 12,345.00, right-justified, in a

12 character field

Set X=12345

Write $J($FN(X,",",2),12)

12,345.00

Example ‑ $Ascii and $Char

Write $Ascii("A")

65

Write $Char(66)

B

Example ‑ Convert uppercase to lowercase

Set X=$ASCII("A")+32

Write $CHAR(X)

a

Example ‑ Convert lowercase to uppercase

Set X=$ASCII("a")-32

Write $CHAR(X)

A

Example ‑ $Select Example

Write $S(X=1:"One",X=2:"Two",X=3:"Three",1:"None")

The above $Select command is equivalent to:

If X=1 {Write "One"}

ElseIf X=2 {Write "Two"}

ElseIf X=3 {Write "Three"}

Else {Write "None"}

Example ‑ $Case Example

Write $CASE(X,1:"One",2:"Two",3:"Three",:"None")

The above $Case command is equivalent to:

For the variable X:

If X=1 {Write "One"}

ElseIf X=2 {Write "Two"}

ElseIf X=3 {Write "Three"}

Else {Write "None"}

Example ‑ $Case Example for Day of Week

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set DayOfWeek=2  
 Set Day=$CASE(DayOfWeek,  
 1:"Sunday",  
 2:"Monday",  
 3:"Tuesday",  
 4:"Wednesday",  
 5:"Thursday",  
 6:"Friday",  
 7:"Saturday",  
 :"error")

Write !,Day

Monday

Example ‑ $Test with the Read command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

R "Prompt: ",X:3 ;timed read command, 3 seconds  
 If $Test {  
 Write !,"The user answered the prompt"  
 }  
 Else {  
   Write !,"The user did not answer the prompt"  
 }

Example ‑ $Increment

Set ^CNTR=""

Set X=$INCREMENT(^CNTR)

Write ^CNTR

1

Set X=$INCREMENT(^CNTR)

Write ^CNTR

2

Example ‑ $Text - Error code table

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

ROUTINEA

For I=1:1:3 Write !,$T(TABLE+I)

Q

TABLE

;;A01^Error code 1

;;A02^Error code 2

;;A03^Error code 3

Example ‑ Error code table output

;;A01^Error code 1

;;A02^Error code 2

;;A03^Error code 3

Example 4‑35 shows the output of running the code in

Example 4‑34.

## 

“When the only tool you have is a hammer, everything looks like a nail.” – Abraham H. Maslow

# List Processing

Example ‑ $ListBuild defines a list of Pets using literal parameters

Set Pets=$LB("Dog","Cat","Fish")

Write !,Pets

DogCatFish

Example ‑ $ListBuild defines a list of Pets using variable parameters

Set Pet1="Dog"

Set Pet2="Cat"

Set Pet3="Fish"

Set Pets=$LB(Pet1,Pet2,Pet3)

Write !,Pets

DogCatFish

Example ‑ $List displays elements from the list of Pets

Set Pets=$LB("Dog","Cat","Fish")

Write !,$LI(Pets,1)

Dog

Write !,$LI(Pets,2)

Cat

Write !,$LI(Pets,3)

Fish

Example ‑ $List used to set variables

Set Pets=$LB("Dog","Cat","Fish")

Set Pet1=$LI(Pets,1)

Write !,Pet1

Dog

Set Pet2=$LI(Pets,2)

Write !,Pet2

Cat

Set Pet3=$LI(Pets,3)

Write !,Pet3

Fish

Example ‑ $List displays the list of Pets

Set Pets=$LB("Dog","Cat","Fish")

Write !,$LI(Pets,1,3)

DogCatFish

Example ‑ Display the entire list of Pets

Set Pets=$LB("Dog","Cat","Fish")

Write !,Pets

DogCatFish

Example ‑ $List to add a fourth element to the list of Pets

Set Pets=$LB("Dog","Cat","Fish")

Set $LI(Pets,4)="Bird"

Write !,Pets

DogCatFishBird

Example ‑ $ListData demonstration

Set Pets=$LB("Dog","Cat","Fish","") ;4 elements are defined

If $LD(Pets,3) Write !,"Element 3 does exist"

Element 3 does exist

Write !,$LD(Pets,3) ;$LD returns a 1 when the element exists

1

If $LD(Pets,4) Write !,"Element 4 is null but does exist"

Element 4 is null but does exist

Write !,$LD(Pets,4) ;$LD returns a 1 when the element is null

1

If '$LD(Pets,5) Write !,"Element 5 does not exist" ;Single quote before the

*Element 5 does not exist*;$LD is a "not" and is interperted as

;If $LD(Pets,5)=0. See Chapter 11 for

;for more information on the "not"

Write !,$LD(Pets,5) ;$LD returns a 0 when the element does

*0*;not exist

Example ‑ $ListGet demonstration

Set Pets=$LB("Dog","Cat","Fish","") ;4 elements are defined

Write !,$LG(Pets,3)

Fish

Write !,$LG(Pets,4) ;$LG returns null because the element is null

<>

Write !,$LG(Pets,5) ;$LG returns null because the element does not exist

<>

Example ‑ $ListFind returns the position of the element found

Set Pets=$LB("Dog","Cat","Fish","")

Write !,$LF(Pets,"Cat") ;"Cat" is the second element

2

Write !,$LF(Pets,"Snake") ;"Snake" does not exist

0

Write !,$LF(Pets,"") ;null does exist

4

Example ‑ $ListFind returns the position of the element found after the third position

Set Pets=$LB("Dog","Cat","Fish","Cat")

Write !,$LF(Pets,"Cat",3) ;Find "Cat" after the third position

4

Example ‑ $ListFind in a compound find

Set Pets=$LB("Dog","Cat","Fish","Cat")

Write !,$LF(Pets,"Cat",$LF(Pets,"Cat"))

4

Example ‑ $ListLength returns the number of elements in Pets

Set Pets=$LB("Dog","Cat","Fish")

Write !,$LL(Pets)

3

Example ‑ $ListLength and the For Loop command

Set Pets=$LB("Dog","Cat","Fish")

For I=1:1:$LL(Pets) Write !,$LI(Pets,I)

Dog

Cat

Fish

Example ‑ $ListNext displays all elements in a list

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Pets=$LB("Dog","Cat","","Fish")

Set Pointer=0 ;Set pointer to 0 to start at top

While $ListNext(Pets,Pointer,Value) {

Write !,Value

}

Dog

Cat

<>

Fish

Example ‑ $ListSame compares two lists

Set Pets1=$LB("Dog","Cat","Fish")

Set Pets2=$LB("Dog","Cat","Fish")

If $LS(Pets1,Pets2) Write !,"The two lists are the same"

The two lists are the same

Write !,$LS(Pets1,Pets2)

1

Set Pets1=$LB("Dog","Cat")

Set Pets2=$LB("Dog","Cat","Fish")

If '$LS(Pets1,Pets2) Write !,"The two lists are not the same"

The two lists are not the same

Write !,$LS(Pets1,Pets2)

0

Example ‑ $ListToString creates a string from a list

Set Pets=$LB("Dog","Cat","Fish")

Set String=$LTS(Pets,"^") ;delimiter ^

Write String

Dog^Cat^Fish

Set Pets=$LB("Dog","Cat","Fish")

Set String=$LTS(Pets,"~") ;delimiter ~

Write String

Dog~Cat~Fish

Example ‑ $ListFromString creates a list from a string

Set String="Dog^Cat^Fish" ;delimited string

Set Pets=$LFS(String,"^") ;delimiter ^

Write Pets

DogCatFish ;list

Set String="Dog~Cat~Fish" ;delimited string

Set Pets=$LFS(String,"~") ;delimiter ~

Write Pets

DogCatFish ;list

Example ‑ $ListValid – checks for a valid list

Set String=$LB("Dog","Cat","Fish")

If $LV(String) Write !,"This is a valid list"

This is a valid list

Write !,$LV(String) ;$LV returns a 1 if the list is valid

1

Set String=$LB("") ;null list is still valid

If $LV(String) Write !,"This is a valid list"

This is a valid list

Write !,$LV(String)

1

Set String="ABC"

If '$LV(String) Write !,"This is NOT a valid list"

This is NOT a valid list

Write !,$LV(String) ;$LV returns a 0 because the list is not valid

0

Example ‑ List Processing Exercise 1, First solution

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Pets=$LB("Dog","Cat","Fish","")

For I=1:1:$LL(Pets) {

If $LD(Pets,I)=1 Write !,$LI(Pets,I)

}

Dog

Cat

Fish

<> ;fourth element displayed, even though it is null

Example ‑ List Processing Exercise 1, Second solution

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Pets=$LB("Dog","Cat","Fish","")

For I=1:1:$LL(Pets) {

If $LG(Pets,I)'="" Write !,$LI(Pets,I)

}

Dog

Cat

Fish

Example ‑ List Processing Exercise 1, Third solution

To run this code you must put the code in a routine, save 7the routine, and then run the routine from the terminal.

Set Pets=$LB("Dog","Cat","Fish","")

For I=1:1:$LL(Pets) {

If $LI(Pets,I)'="" Write !,$LI(Pets,I)

}

Dog

Cat

Fish

Example ‑ List Processing Exercise 1, Fourth solution

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set String="Dog^Cat^Fish"

Set Pets=$LFS(String,"^")

For I=1:1:$LL(Pets) {

If $LI(Pets,I)'="" Write !,$LI(Pets,I)

}

Dog

Cat

Fish

Example ‑ List Processing Exercise 1, Fifth solution

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set String="Dog^Cat^Fish"

Set Pets=$LFS(String,"^")

Set Pointer=0

While $ListNext(Pets,Pointer,Value) {

Write !,Value

}

Dog

Cat

Fish

Example ‑ List Processing Exercise 2, First solution

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Pets=$LB("Dog","Cat","Fish")

For I=1:1:$LL(Pets) {

If $LI(Pets,I)="Dog" Set $LI(Pets,I)=""

}

For I=1:1:$LL(Pets) Write !,$LI(Pets,I)

<>

Cat

Fish

Example ‑ List Processing Exercise 2, Second solution

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Pets=$LB("Dog","Cat","Fish")

Set Position=$LF(Pets,"Dog") ;List Find

If Position'=0 {

Set $LI(Pets,Position)="" ;Position of "Dog"

Write !,"Position of ""Dog"" was: ",Position

}

Position of "Dog" was: 1

For I=1:1:$LL(Pets) Write !,$LI(Pets,I)

<>

Cat

Fish

Example ‑ List Processing Exercise 2, Third solution

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set String="Dog^Cat^Fish"

Set Pets=$LFS(String,"^")

Set Position=$LF(Pets,"Dog")

If Position'=0 {

Set $LI(Pets,Position)="" ;Position of "Dog"

Write !,"Position of ""Dog"" is: ",Position

}

Position of "Dog" is 1

For I=1:1:$LL(Pets) Write !,$LI(Pets,I)

<>

Cat

Fish

Example ‑ Switch Dog and Fish

Set Pets=$LB("Dog","Cat","Fish")

Set Pos1=$LF(Pets,"Dog") ;position of Dog

Set Pos2=$LF(Pets,"Fish") ;position of Fish

If Pos1=0!(Pos2=0) Write "Not found" ;if either Dog or Fish is not found Quit

Set $LI(Pets,Pos1)="Fish"

Set $LI(Pets,Pos2)="Dog"

For I=1:1:$LL(Pets) Write !,$LI(Pets,I)

Fish

Cat

Dog

Example ‑ Defines delimited list of Pets

Set Pets="Dog^Cat^Fish"

Write !,Pets

Dog^Cat^Fish

Example ‑ Define a delimited list of Pets using variable parameters

Set Pet1="Dog"

Set Pet2="Cat"

Set Pet3="Fish"

Set Pets=Pet1\_"^"\_ Pet2\_"^"\_ Pet3

Write !,Pets

Dog^Cat^Fish

Example ‑ $Piece defines a delimited list

Set Pets=""

Set $P(Pets,"^",1)="Dog" ;define 1st piece

Set $P(Pets,"^",2)="Cat" ;define 2nd piece

Set $P(Pets,"^",3)="Fish" ;define 3rd piece

Write !,Pets

Dog^Cat^Fish

Example ‑ $Piece displays elements in a delimited list

Set Pets="Dog^Cat^Fish"

Write !,$P(Pets,"^",1) ;display the 1st piece

Dog

Write !,$P(Pets,"^",2) ;display the 2nd piece

Cat

Write !,$P(Pets,"^",3) ;display the 3rd piece

Fish

Example ‑ $Piece breaks out elements from a delimited list

Set Pet1=$P(Pets,"^",1) ;1st piece set to variable Pet1

Write !,Pet1

Dog

Set Pet2=$P(Pets,"^",2) ;2nd piece set to variable Pet2

Write !,Pet2

Cat

Set Pet3=$P(Pets,"^",3) ;3rd piece set to variable Pet3

Write !,Pet3

Fish

Example ‑ $Piece displays all of the delimited list elements

Write !,$P(Pets,"^",1,3)

Dog^Cat^Fish

Example ‑ Write command displays all the delimited list elements

Write !,Pets

Dog^Cat^Fish

Example ‑ $Piece sets piece four to "Bird"

Set $P(Pets,"^",4)="Bird"

Write !,Pets

Dog^Cat^Fish^Bird

Example ‑ $Data and $Piece

Set Pets="Dog^Cat^Fish"

Set Pet3=$P(Pets,"^",3)

Write !,$D(Pet3)

1

Set Pet4=$P(Pets,"^",4)

Write !,$D(Pet4)

1

Example ‑ $Get and $Piece

Set Pets="Dog^Cat^Fish"

Set Pet3=$P(Pets,"^",3)

Write !,$G(Pet3)

Fish

Set Pet4=$P(Pets,"^",4)

Write !,$G(Pet4)

<>

Example ‑ Delimited List Processing Exercise

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Pets=""

Set $P(Pets,"^",1)="Dog"

Set $P(Pets,"^",2)="Cat"

Set $P(Pets,"^",3)="Fish"

Set $P(Pets,"^",4)=""

For I=1:1:$L(Pets,"^") {

If $P(Pets,"^",I)'="" Write !,$P(Pets,"^",I)

}

Dog

Cat

Fish

Example ‑ Delimited problem demonstrated

Set DOG="Rover"

Set CAT="Tiger"

Set FISH="Lamont"

Set PIG="Cud^dles"

Set $P(Pets,"^",1)=DOG

Set $P(Pets,"^",2)=CAT

Set $P(Pets,"^",3)=FISH

Set $P(Pets,"^",4)=PIG

For I=1:1:$L(Pets,"^") Write !,I," – ",$P(Pets,"^",I)

1 – Rover

2 – Tiger

3 – Lamount

4 – Cud

5 - dles

Example ‑ Delimited problem solved using the $ListBuild

Set DOG="Rover"

Set CAT="Tiger"

Set FISH="Lamont"

Set PIG="Cud^dles"

Set Pets=$LB(DOG,CAT,FISH,PIG)

For I=1:1:$LL(Pets) Write !,I," – ",$LI(Pets,I)

1 – Rover

2 – Tiger

3 – Lamount

4 – Cud^dles

Example ‑ Correctly setting a Global array using $Piece or $ListBuild

Set ^Global1="Rover^Tiger^Lamount^Cuddles"

Set ^Global2=$LB("Rover","Tiger","Lamont","Cuddles")

Set $P(^Global1,"^",2)="Kitty"

Set $LI(^Global2,2)="Kitty"

Example ‑ Retrieving an element using $Piece

Set ITEM="ABC^123\*456^DEF"

Write $P($P(ITEM,"\*",1),"^",2)

123

Example ‑ It is Illegal to set an element using $Piece in a compounded manner.

Set ITEM="ABC^123\*456^DEF"

Set $P($P(ITEM,"\*",1),"^",2)=789

^

<SYNTAX>

Example ‑ $ListBuild compounded upon itself

Set Pets=$LB("Dogs",$LB("Rover","Teddy"),"Cats",$LB("Tiger","Kitty"))

Example ‑ Array to represent a compounded list

Set Pets("Dogs","Rover")=""

Set Pets("Dogs","Teddy")=""

Set Pets("Cats","Tiger")=""

Set Pets("Cats","Kitty")=""

## 

## 

*“Failure is not an option, it's included with the software.”*

*– Murphy’s Laws*

# Global Processing I

Table ‑ Differences between Global Arrays, Global Variable, Local Arrays, and Local Variables

|  |  |  |
| --- | --- | --- |
| What | Description | |
| Global Arrays | Multi-dimensional using subscripts | Persistent, permanent, resides on disk |
| Global Variables | One-dimensional, Scalar, no subscripts | Persistent, permanent, resides on disk |
| Local Arrays | Multi-dimensional using subscripts | Temporary, resides only in memory |
| Local Variables | One-dimensional, Scalar, no subscripts | Temporary, resides only in memory |

Table ‑ Transportation Machines Outline Structure

|  |  |
| --- | --- |
| Outline Structure Entries | Data or Information |
| I. Cars | Data about Cars |
| A. Domestic |  |
| i. Dodge |  |
| a. Caravan |  |
| b. 150 Truck |  |
| B. Foreign |  |
| i. Toyota | Data about Toyota |
| a. Tercel |  |
| ii. BMW |  |
| II. Airplanes |  |
| A. Military |  |
| i. Jets |  |
| a. F-14 | Data about F-14s |
| b. F-16 |  |
| ii. Prop planes |  |
| a. P-38 |  |
| B. Commercial | Data about commercial planes |
| ii. Jets |  |
| a. 707 |  |
| b. 747 | Data about 747s |

Table ‑ Outline Structure and Global Structure

|  |  |  |
| --- | --- | --- |
| Outline Structure Entries | Data | Global Structure |
| I. Cars | Data | ^TM("Cars")="Data" |
| A. Domestic |  | ^TM("Cars","Domestic")="" |
| i. Dodge |  | ^TM("Cars","Domestic","Dodge")="" |
| a. Caravan |  | ^TM("Cars","Domestic","Dodge","Caravan")="" |
| b. 150 Truck |  | ^TM("Cars","Domestic","Dodge","150 Truck")="" |
| B. Foreign |  | ^TM("Cars","Foreign")="" |
| i. Toyota | Data | ^TM("Cars","Foreign","Toyota")="Data" |
| a. Tercel |  | ^TM("Cars","Foreign","Toyota","Tercel")="" |
| ii. BMW |  | ^TM("Cars","Foreign","BMW")="" |
| II. Airplanes |  | ^TM("Airplanes")="" |
| A. Military |  | ^TM("Airplanes","Military")="" |
| i. Jets |  | ^TM("Airplanes","Military","Jets")="" |
| a. F-14 | Data | ^TM("Airplanes","Military","Jets","F-14")="Data" |
| b. F-16 |  | ^TM("Airplanes","Military","Jets","F-16")="" |
| ii. Prop planes |  | ^TM("Airplanes","Military","Prop planes")="" |
| a. P-38 |  | ^TM("Airplanes","Military","Prop planes","P-38")="" |
| B. Commercial | Data | ^TM("Airplanes","Commercial")="Data" |
| ii. Jets |  | ^TM("Airplanes","Commercial","Jets")="" |
| a. 707 |  | ^TM("Airplanes","Commercial","Jets","707")="" |
| b. 747 | Data | ^TM("Airplanes","Commercial","Jets","747")="Data" |

Example ‑ Traversing subscript one of a Global

Line 1: Set S1=""

Line 2: For Do Quit:S1="" ;2 spaces after the For and Do

Line 3: . Set S1=$O(^TM(S1)) Quit:S1=""

Line 4: . Write !,"S1: ",S1

Example ‑ Output from

Example 6‑1

Airplanes

Cars

Example ‑ Rewrite of

Example 6‑1 using Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" Do {

Set S1=$O(^TM(S1)) Quit:S1=""

Write !,S1

} While S1'=""

Example ‑ Traversing subscripts one and two of a Global

Line 1: Set S1=""

Line 2: For Do Quit:S1="" ;2 spaces after the For and Do

Line 3: . Set S1=$O(^TM(S1)) Quit:S1=""

Line 4: . Write !,"S1: ",S1

Line 5: . Set S2=""

Line 6: . For Do Quit:S2="" ;2 spaces after the For and Do

Line 7: . . Set S2=$O(^TM(S1,S2)) Quit:S2=""

Line 8: . . Write !," S2: ",S2

Example ‑, Output from

Example 6‑4

S1: Airplanes

S2: Commercial

S2: Military

S1: Cars

S2: Domestic

S2: Foreign

In

Example 6‑5, the output from the code in

Example 6‑4 is displayed, for subscripts one and two entries.

Example ‑ Do While command (Structured Code)

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Counter=0 Do {  
 Set Counter=Counter+1  
 Write !,Counter  
 } While Counter'=5

Example ‑ Output from code in

Example 6‑6

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

1

2

3

4

5

Example ‑ Nested Do While command (Structured Code)

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Counter=0 Do {

Set Counter=Counter+1

Set Counter2=0 Do {

Set Counter2=Counter2+1

Write !,"Counter: ",Counter

Write !," Counter2: ",Counter2

} While Counter2<3

} While Counter'=3

Example ‑ Output from code in

Example 6‑8

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Counter: 1

Counter2: 1

Counter: 1

Counter2: 2

Counter: 1

Counter2: 3

Counter: 2

Counter2: 1

Counter: 2

Counter2: 2

Counter: 2

Counter2: 3

Counter: 3

Counter2: 1

Counter: 3

Counter2: 2

Counter: 3

Counter2: 3

Example ‑ While command (Structured Code)

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Counter=0

While Counter'=5 {  
 Set Counter=Counter+1  
 Write !,Counter  
 }

Example ‑ Output from code in

Example 6‑10

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

1

2

3

4

5

Example ‑ Nested While command (Structured Code)

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Counter=0   
 While Counter'=3 {  
 Set Counter=Counter+1  
 Set Counter2=0   
 While Counter2<3 {  
 Set Counter2=Counter2+1  
 Write !,"Counter: ",Counter   
 Write !," Counter2: ",Counter2  
 }  
  }

Example ‑ Output from code in

Example 6‑12

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Counter: 1

Counter2: 1

Counter: 1

Counter2: 2

Counter: 1

Counter2: 3

Counter: 2

Counter2: 1

Counter: 2

Counter2: 2

Counter: 2

Counter2: 3

Counter: 3

Counter2: 1

Counter: 3

Counter2: 2

Counter: 3

Counter2: 3

Example ‑ Rewrite of

Example 6‑4 using Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" Do {

Set S1=$O(^TM(S1)) Quit:S1=""

Write !,"S1: ",S1

Set S2="" Do {

Set S2=$O(^TM(S1,S2)) Quit:S2=""

Write !," S2: ",S2

} While S2'=""

} While S1'=""

Example ‑ Traversing subscripts one, two and three of a Global

Line 1: Set S1=""

Line 2: For Do Quit:S1="" ;2 spaces after the For and Do

Line 3: . Set S1=$O(^TM(S1)) Quit:S1=""

Line 4: . Write !,"S1: ",S1

Line 5: . Set S2=""

Line 6: . For Do Quit:S2="" ;2 spaces after the For and Do

Line 7: . . Set S2=$O(^TM(S1,S2)) Quit:S2=""

Line 8: . . Write !," S2: ",S2

Line 9: . . Set S3=""

Line 10: . . For Do Quit:S3="" ;2 spaces after the For and Do

Line 11: . . . Set S3=$O(^TM(S1,S2,S3)) Quit:S3=""

Line 12: . . . Write !," S3: ",S3

Example ‑ Output from

Example 6‑15

S1: Airplanes

S2: Commercial

S3: Jets

S2: Military

S3: Jets

S3: Prop planes

S1: Cars

S2: Domestic

S3: Dodge

S2: Foreign

S3: BMW

S3: Toyota

Example ‑ Rewrite of

Example 6‑15 using Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" Do {

Set S1=$O(^TM(S1)) Quit:S1="" ;get the next S1 subscript

Write !,"S1: ",S1

Set S2="" Do {

Set S2=$O(^TM(S1,S2)) Quit:S2="" ;get the next S2 subscript

Write !," S2: ",S2

Set S3="" Do {

Set S3=$O(^TM(S1,S2,S3)) Quit:S3="" ;get the next S3 subscript

Write !," S3: ",S3

} While S3'=""

} While S2'=""

} While S1'=""

#### 

Example ‑ Command Structure One for Traversing a Global

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" For Do Quit:S1="" ;2 spaces after the For and Do

.Set S1=$O(^GLOBAL(S1)) Quit:S1=""

.Set S2="" For Do Quit:S2="" ;2 spaces after the For and Do

..Set S2=$O(^GLOBAL(S1,S2)) Quit:S2=""

..Set S3="" For Do Quit:S3="" ;2 spaces after the For and Do

...Set S3=$O(^GLOBAL(S1,S2,S3)) Quit:S3=""

Example ‑ Command Structure Two for Traversing a Global

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" For Set S1=$O(^GLOBAL(S1)) Quit:S1="" Do ;2 sp after For and before Do

.Set S2="" For Set S2=$O(^GLOBAL(S1,S2)) Quit:S2="" Do ;ditto

..Set S3="" For Set S3=$O(^GLOBAL(S1,S2,S3)) Quit:S3="" Do ;ditto

Example ‑ Command Structure Three for Traversing a Global

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set (S1,S2,S3)=""

For Set S1=$O(^GLOBAL(S1)) Quit:S1="" Do ;2 sp after For and before Do

.For Set S2=$O(^GLOBAL(S1,S2)) Quit:S2="" Do ;ditto

..For Set S3=$O(^GLOBAL(S1,S2,S3)) Quit:S3="" Do ;ditto

Example ‑ Command Structure Four for Traversing a Global, the While command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1=$O(^GLOBAL("")) ;initially set the S1 control param

While (S1'="") { ;S1'="" is the first control param

Set S2=$O(^GLOBAL(S1,"")) ;initially set the S2 control param

While (S2'="") { ;S2'="" is the second control param

Set S3=$O(^GLOBAL(S1,S2,"")) ;initially set the S3 control param

While (S3'="") { ;S3'="" is the third control param

; process data for S3

Set S3=$O(^GLOBAL(S1,S2,S3)) ;get next S3

}

; process data for S2

Set S2=$O(^GLOBAL(S1,S2)) ;get next S2 entry

}

; process data for S1

Set S1=$O(^GLOBAL(S1)) ;get next S1 entry

}

Example ‑ Command Structure Five for Traversing a Global, the Do While command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" Do {

Set S1=$O(^GLOBAL(S1)) Quit:S1=""

Set S2="" Do {

Set S2=$O(^GLOBAL(S1,S2)) Quit:S2=""

Set S3="" Do {

Set S3=$O(^GLOBAL(S1,S2,S3)) Quit:S3=""

} While S3'=""

} While S2'=""

} While S1'=""

Example ‑ Accessing Global data

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Line 1: Set S1=""

Line 2: For Do Quit:S1="" ;2 spaces after the For and Do

Line 3: . Set S1=$O(^TM(S1)) Quit:S1=""

Line 4: . Write !,"S1: ",S1

Line 4a: . Write " = ",^TM(S1)

Line 5: . Set S2=""

Line 6: . For Do Quit:S2="" ;2 spaces after the For and Do

Line 7: . . Set S2=$O(^TM(S1,S2)) Quit:S2=""

Line 8: . . Write !," S2: ",S2

Line 8a: . . Write " = ",^TM(S1,S2)

Line 9: . . Set S3=""

Line 10: . . For Do Quit:S3="" ;2 spaces after the For and Do

Line 11: . . . Set S3=$O(^TM(S1,S2,S3)) Quit:S3=""

Line 12: . . . Write !," S3: ",S3

Line 12a: . . . Write " = ",^TM(S1,S2,S3)

Example ‑ Output from

Example 6‑23

S1: Airplanes =

S2: Commercial = Data

S3: Jets =

S2: Military =

S3: Jets =

S3: Prop planes =

S1: Cars = Data

S2: Domestic =

S3: Dodge =

S2: Foreign =

S3: BMW =

S3: Toyota = Data

Example ‑ Rewrite of

Example 6‑23 using Structured Code, Do While command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" Do {

Set S1=$O(^TM(S1)) Quit:S1=""

Write !,"S1: ",S1

Write " = ",^TM(S1)

Set S2="" Do {

Set S2=$O(^TM(S1,S2)) Quit:S2=""

Write !," S2: ",S2

Write " = ",^TM(S1,S2)

Set S3="" Do {

Set S3=$O(^TM(S1,S2,S3)) Quit:S3=""

Write !," S3: ",S3

Write " = ",^TM(S1,S2,S3)

} While S3'=""

} While S2'=""

} While S1'=""

Example ‑ Setting up a simple Global Array of Pets

Set ^Pets("Dog","Boxer","Male","Buddy")="9^0"

Set ^Pets("Dog","Lab","Female","Loverly")="6^0"

Set ^Pets("Dog","Lab","Male","Tiny")="5^1"

Set ^Pets("Cat","Burmese","Male","BoyCat")="3^0"

Set ^Pets("Cat","Burmese","Female","TomBoy")="3^0"

Set ^Pets("Cat","Korat","Female","Fancy")="5^1"

Example ‑ Code to traverse and display the Global created in Example 6‑26

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Animal="" Do {

Set Animal=$O(^Pets(Animal)) Quit:Animal=""

Set Breed="" Do {

Set Breed=$O(^Pets(Animal,Breed)) Quit:Breed=""

Set Sex="" Do {

Set Sex=$O(^Pets(Animal,Breed,Sex)) Quit:Sex=""

Set Name="" Do {

Set Name=$O(^Pets(Animal,Breed,Sex,Name)) Quit:Name=""

Set Data=^Pets(Animal,Breed,Sex,Name)

Set Age=$P(Data,"^",1)

Set Adoption=$P(Data,"^",2)

If Adoption=1 Set Available="is"

If Adoption=0 Set Available="is not"

Write !,Animal," named ",Name,", Breed ",Breed,", Sex ",Sex

Write ", ",Age," years old, ",Available," available for adoption"

} While Name'=""

} While Sex'=""

} While Breed'=""

} While Animal'=""

Example ‑ Output from the routine in

Example 6‑27 based on the Global created in Example 6‑26

Cat named TomBoy, Breed Burmese, Sex Female, 3 years old, is not available for adoption

Cat named BoyCat, Breed Burmese, Sex Male, 3 years old, is not available for adoption

Cat named Fancy, Breed Korat, Sex Female, 5 years old, is available for adoption

Dog named Buddy, Breed Boxer, Sex Male, 9 years old, is not available for adoption

Dog named Loverly, Breed Lab, Sex Female, 6 years old, is not available for adoption

Dog named Tiny, Breed Lab, Sex Male, 5 years old, is available for adoption

Example ‑ Modification to the ^Pets Global created in Example 6‑26

Kill ^Pets("Dog","Lab","Female","Loverly")

Set $P(^Pets("Cat","Korat","Female","Fancy"),"^",2)=0

Set ^Pets("Rat","Rodent","Male","Ben")="2^1"

Example ‑ Output from the routine in Example 6‑26 with the Global modification made in

Example 6‑29

Cat named TomBoy, Breed Burmese, Sex Female, 3 years old, is not available for adoption

Cat named BoyCat, Breed Burmese, Sex Male, 3 years old, is not available for adoption

Cat named Fancy, Breed Korat, Sex Female, 5 years old, is not available for adoption

Dog named Buddy, Breed Boxer, Sex Male, 9 years old, is not available for adoption

Dog named Tiny, Breed Lab, Sex Male, 5 years old, is available for adoption

Rat named Ben, Breed Rodent, Sex Male, 2 years old, is available for adoption

Example ‑ Using $Query to traverse a Global array

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set TM="^TM"

For Do Quit:TM="" ;2 spaces after the For and Do

. Set TM=$Q(@TM) Quit:TM=""

. Write !,TM," = ",@TM

Example ‑ Output from

Example 6‑31

^TM("Airplanes") =

^TM("Airplanes","Commercial") = Data

^TM("Airplanes","Commercial","Jets") =

^TM("Airplanes","Commercial","Jets",707) =

^TM("Airplanes","Commercial","Jets",747) = Data

^TM("Airplanes","Military") =

^TM("Airplanes","Military","Jets") =

^TM("Airplanes","Military","Jets","F-14") = Data

^TM("Airplanes","Military","Jets","F-16") =

^TM("Airplanes","Military","Prop planes") =

^TM("Airplanes","Military","Prop planes","P-38") =

^TM("Cars") = Data

^TM("Cars","Domestic") =

^TM("Cars","Domestic","Dodge") =

^TM("Cars","Domestic","Dodge","150 Truck") =

^TM("Cars","Domestic","Dodge","Caravan") =

^TM("Cars","Foreign") =

^TM("Cars","Foreign","BMW") =

^TM("Cars","Foreign","Toyota") = Data

^TM("Cars","Foreign","Toyota","Tercel") =

Example ‑ Using Constant Subscripts

Kill ^Pets2

Set SUB1="First Subscript"

Set SUB3="Third Subscript"

Set ^Pets2(SUB1,"Dog",SUB3)=""

Set ^Pets2(SUB1,"Cat",SUB3)=""

Set ^Pets2(SUB1,"Fish",SUB3)=""

Set ^Pets2(SUB1,"Turtle",SUB3)=""

Example ‑ Traverse a Global with Constant Subscripts

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set (SUB1,SUB2,SUB3)=""

For Set SUB1=$O(^Pets2(SUB1)) Quit:SUB1="" Do ;2 spaces after the

. For Set SUB2=$O(^Pets2(SUB1,SUB2)) Quit:SUB2="" Do ;For and before

. . For Set SUB3=$O(^Pets2(SUB1,SUB2,SUB3)) Quit:SUB3="" Do ;the Do

. . . Write !,SUB1," - ",SUB2," - ",SUB3

Quit

First Subscript - Cat - Third Subscript

First Subscript - Dog - Third Subscript

First Subscript - Fish - Third Subscript

First Subscript - Turtle - Third Subscript

Example ‑ Traversing a Global specifying with Constant Subscripts

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set (SUB1,SUB3)=""

For Set SUB1=$O(^Pets2(SUB1)) Quit:SUB1="" Do

. For SUB2="Dog","Cat","Turtle" Do ;specify what values to look up

. . For Set SUB3=$O(^Pets2(SUB1,SUB2,SUB3)) Quit:SUB3="" Do

. . . Write !,SUB1," - ",SUB2," - ",SUB3

Quit

First Subscript - Dog - Third Subscript

First Subscript - Cat - Third Subscript

First Subscript - Turtle - Third Subscript

Example ‑ Building a Data Global

Set ^People(1)=$LB("Doe","John","5545-56-2322","1234","Los Angeles","CA","95111")

Set ^People(2)=$LB("Doe","Jane","5544-20-2232","2345","Los Angeles","CA","95111")

Set ^People(3)=$LB("Jacobs","Dawn","7894-11-4545","3456","Harvard","MA","01666")

Set ^People(4)=$LB("Dover","Ilene","1190-56-0933","4567","Dallas","TX","75211")

Set ^People(5)=$LB("Johnson","Mike","3406-44-3344","5678","Ink","AR","71933")

Set ^People(6)=$LB("Dover","Ben","3434-24-3344","6789","Inkwell","AR","71955")

Example ‑ Building an Index Global

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Num="" Do {

Set Num=$O(^People(Num)) Quit:Num=""

Set Data=^People(Num)

Set Ln=$Li(Data,1) ;last name

Set Fn=$Li(Data,2) ;first name

Set SSN=$Li(Data,3) ;social security number

Set MRN=$Li(Data,4) ;medical record number

Set City=$Li(Data,5) ;City

Set State=$Li(Data,6) ;State

Set Zip=$Li(Data,7) ;Zip

Set ^PeopleIndex("Name",Ln\_","\_Fn,Num)=""

Set ^PeopleIndex("Ln",Ln,Num)=""

Set ^PeopleIndex("SSN",SSN)=Num

Set ^PeopleIndex("MRN",MRN)=Num

Set ^PeopleIndex("City",City,Num)=""

Set ^PeopleIndex("State",State,Num)=""

Set ^PeopleIndex("Zip",Zip,Num)=""

} While Num'=""

Example ‑ Resulting Global from executing the routine in

Example 6‑37

ZW ^PeopleIndex

^PeopleIndex("City","Dallas",4)=""

^PeopleIndex("City","Harvard",3)=""

^PeopleIndex("City","Ink",5)=""

^PeopleIndex("City","Inkwell",6)=""

^PeopleIndex("City","Los Angeles",1)=""

^PeopleIndex("City","Los Angeles",2)=""

^PeopleIndex("Ln","Doe",1)=""

^PeopleIndex("Ln","Doe",2)=""

^PeopleIndex("Ln","Dover",4)=""

^PeopleIndex("Ln","Dover",6)=""

^PeopleIndex("Ln","Jacobs",3)=""

^PeopleIndex("Ln","Johnson",5)=""

^PeopleIndex("MRN",1234)=1

^PeopleIndex("MRN",2345)=2

^PeopleIndex("MRN",3456)=3

^PeopleIndex("MRN",4567)=4

^PeopleIndex("MRN",5678)=5

^PeopleIndex("MRN",6789)=6

^PeopleIndex("Name","Doe,Jane",2)=""

^PeopleIndex("Name","Doe,John",1)=""

^PeopleIndex("Name","Dover,Ben",6)=""

^PeopleIndex("Name","Dover,Ilene",4)=""

^PeopleIndex("Name","Jacobs,Dawn",3)=""

^PeopleIndex("Name","Johnson,Mike",5)=""

^PeopleIndex("SSN","1190-56-0933")=4

^PeopleIndex("SSN","3406-44-3344")=5

^PeopleIndex("SSN","3434-24-3344")=6

^PeopleIndex("SSN","5544-20-2232")=2

^PeopleIndex("SSN","5545-56-2322")=1

^PeopleIndex("SSN","7894-11-4545")=3

^PeopleIndex("State","AR",5)=""

^PeopleIndex("State","AR",6)=""

^PeopleIndex("State","CA",1)=""

^PeopleIndex("State","CA",2)=""

^PeopleIndex("State","MA",3)=""

^PeopleIndex("State","TX",4)=""

^PeopleIndex("Zip",71933,5)=""

^PeopleIndex("Zip",71955,6)=""

^PeopleIndex("Zip",75211,4)=""

^PeopleIndex("Zip",95111,1)=""

^PeopleIndex("Zip",95111,2)=""

^PeopleIndex("Zip","01666",3)=""

Example ‑ List People sorted by State

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set State="" Do {

Set State=$O(^PeopleIndex("State",State)) Quit:State=""

Set Num="" Do {

Set Num=$O(^PeopleIndex("State",State,Num)) Quit:Num=""

Set Data=^People(Num)

Set Ln=$Li(Data,1)

Set Fn=$Li(Data,2)

Write !,Fn," ",Ln," lives in: ",State

} While Num'=""

} While State'=""

Mike Johnson lives in: AR

Ben Dover lives in: AR

John Doe lives in: CA

Jane Doe lives in: CA

Dawn Jacobs lives in: MA

Ilene Dover lives in: TX

Example ‑ List all the Zip Code for all people whose last name begins with a "D"

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Ln="D" Do {

Set Ln=$O(^PeopleIndex("Ln",Ln)) Quit:$E(Ln,1,1)'="D"

Set Num="" Do {

Set Num=$O(^PeopleIndex("Ln",Ln,Num)) Quit:Num=""

Set Data=^People(Num)

Set Fn=$Li(Data,2)

Set Zip=$Li(Data,6)

Write !,Fn," ",Ln," has a Zip Code: ",Zip

} While Num'=""

} While $E(Ln,1,1)="D"

John Doe has a Zip Code: 95111

Jane Doe has a Zip Code: 95111

Ilene Dover has a Zip Code: 75211

Ben Dover has a Zip Code: 71955

Example ‑ Subscripts in the ^PeopleIndex Global

; These indices have 3 subscripts and have no data to the right

; of the equal sign

^PeopleIndex("City","Los Angeles",2)=""

78^PeopleIndex("Name","Dover,Ben",6)=""

^PeopleIndex("State","TX",4)=""

^PeopleIndex("Zip","01666",3)=""

; These indices have 2 subscripts and have data to the right

; of the equal sign

^PeopleIndex("SSN","7894-11-4545")=3

^PeopleIndex("MRN",4567)=4

# Global Processing II

Example ‑ Default Namespaces

USER> ;namespace USER

SAMPLES> ;namespace SAMPLES

%SYS> ;namespace %SYS

Example ‑ Display the default namespace

Write $ZU(5) ;older way to display the default namespace

USER

Write $SYSTEM.SYS.NameSpace() ;newer way to display the default namespace

USER

Example ‑ Changing Namespaces

ZN "SAMPLES" ;change your Namespace to SAMPLES

ZNSpace "SAMPLES" ;change your Namespace to SAMPLES

Set X=$ZU(5,"SAMPLES") ;change your Namespace to SAMPLES

;Note; the $ZU command is obsolete in

;newer versions of Caché

Do ^%CD ;%CD, utility to change Namespaces

Namespace: SAMPLES ;%CD will ask for a Namespace

Example ‑ Show all Namespaces

Do ^%CD ;%CD, utility to change namespaces

Namespace: *?* ;enter a question mark here

'?' for help.

'@' (at-sign) to edit the default, the last namespace

name attempted. Edit the line just as if it were

a line of code.

<RETURN> will leave you in the current namespace.

Here are the defined namespaces:

%SYS

DOCBOOK

SAMPLES

USER

Example ‑ Extended Syntax

Write ^["USER"]GLOBAL ;constant used as Extended Syntax, this *Global*

;is in the Namespace USER

Set EXTSYN="USER"

Write ^[EXTSYN]GLOBAL ;variable used as Extended Syntax, this *Global*

;is in the Namespace USER

Example ‑ ^CacheTempUser($J)

Kill ^CacheTempUser($J) ;clean your own allotment of CacheTempUser

Set ^CacheTempUser($J)="Abc"

Set ^CacheTempUser($J,"Sub1")="Def"

Write ^CacheTempUser($J)

Abc

Write ^CacheTempUser($J,"Sub1")

Def

Kill ^CacheTempUser($J) ;clean up before you exit

Quit

Example ‑ Process-Private Globals

;initially no need to kill the Global

Set ^||Global="Abc"

Set ^||Global("SUB1")="Def"

Write ^||Global

Abc

Write ^||Global("SUB1")

Def

;no need to kill the Global at the end of processing

Q

Example ‑ ^TMP($J) or ^TEMP($J)

Kill ^TMP($J) ;clean your own allotment of ^TMP

Set ^TMP($J)="Abc"

Set ^TMP($J,"Sub1")="Def"

Write ^TMP($J)

Abc

Write ^TMP($J,"Sub1")

Def

Kill ^TMP($J) ;clean up before you exit

Quit

Example ‑ Naked Indicators or Naked References

Kill

Set ^X("SUB1","SUB2","SUB3")="X-Data"

Write ^("SUB3") ;Naked Indicator reference

X-Data

Example ‑ Naked Indicators, new full reference inserted

Kill

Set ^X("SUB1","SUB2","SUB3")="X-Data"

Set ^Y("SUB3")="Y-Data"

Write ^("SUB3") ;Naked Indicator reference

Y-Data

Example ‑ Merge command

Kill

Set X(1,2)=12

Set X(1,3)=13

Set Y(2,1)=21

Set Y(2,2)=22

Merge Y=X

;After the Merge above, the Y array is as follows:

Write

X(1,2)=12

X(1,3)=13

Y(1,2)=12

Y(1,3)=13

Y(2,1)=21

Y(2,2)=22

Example ‑ Merge command

Kill

Set X("A","B1")="AB1"

Set X("A","B2")="AB2"

Set Y("C")="C"

Merge Y=X

;After the Merge above, the Y array is as follows:

Write

X("A","B1")="AB1"

X("A","B2")="AB2"

Y("A","B1")="AB1"

Y("A","B2")="AB2"

Y("C")="C"

Example ‑ Merge command

Kill

Set X("A","B1")="AB1"

Set X("A","B2")="AB2"

Set Y("C")="C"

Merge Y("C")=X

;After the Merge above, the Y array is as follows:

Write

X("A","B1")="AB1"

X("A","B2")="AB2"

Y("C")="C"

Y("C","A","B1")="AB1"

Y("C","A","B2")="AB2"

Example ‑ Merge command

Kill

Set X("A","B1")="AB1"

Set X("A","B2")="AB2"

Set Y("C","D")="C"

Merge Y("C")=X

;After the Merge above, the Y array is as follows:

Write

X("A","B1")="AB1"

X("A","B2")="AB2"

Y("C","A","B1")="AB1"

Y("C","A","B2")="AB2"

Y("C","D")="C"

Example ‑ Merge command difficulty

Kill ARRAY1

Merge ARRAY2=ARRAY1

Write $D(ARRAY2)

0

Example ‑ Lock command

Lock ^X

Example ‑ Multiple Lock commands

Lock ^X,^Y,^Z

Example ‑ Incremental Lock command

Lock +^Y

Example ‑ Incremental Unlock command

Lock -^Y

Example ‑ Unlock All command

Lock

Example ‑ Lock a Global Array

Set ^A(SUB1)=1

Set ^A(SUB1,SUB2)=2

Set ^A(SUB1,SUB2,SUB3)=3

Lock ^A

Example ‑ Lock part of a Global Array

Set ^A(SUB1)=1

Set ^A(SUB1,SUB2)=2

Set ^A(SUB1,SUB2,SUB3)=3

Lock ^A(SUB1,SUB2,SUB3)

Example ‑ Kill command

Set ^X=1

Set ^X("A")=2

Set ^X("B","C")=3

Kill ^X

ZW ^X

<>

Example ‑ Kill a Global Array

Set ^A(SUB1)=1

Set ^A(SUB1,SUB2)=2

Set ^A(SUB1,SUB2,SUB3)=3

. . . many sets representing many descendants

Kill ^A

ZW ^A

<>

Example ‑ Global of Pets

Set ^Pets("Dog","Boxer","Male","Buddy")="9^0"

Set ^Pets("Dog","Lab","Female","Loverly")="6^0"

Set ^Pets("Dog","Lab","Male","Tiny")="5^1"

Set ^Pets("Cat","Burmese","Male","BoyCat")="3^0"

Set ^Pets("Cat","Burmese","Female","TomBoy")="3^0"

Set ^Pets("Cat","Korat","Female","Fancy")="5^1"

Example ‑ Answers to the Exercises using Command Structure One for Traversing a Global

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

;1. Traverse down all four subscripts of the Pets Global using

;$Order and display each subscript and both pieces of data.

Set S1="" For Do Quit:S1="" ;two spaces after the For and Do

. Set S1=$O(^Pets(S1)) Quit:S1=""

. Set S2="" For Do Quit:S2=""

. . Set S2=$O(^Pets(S1,S2)) Quit:S2=""

. . Set S3="" For Do Quit:S3=""

. . . Set S3=$O(^Pets(S1,S2,S3)) Quit:S3=""

. . . Set S4="" For Do Quit:S4=""

. . . . Set S4=$O(^Pets(S1,S2,S3,S4)) Quit:S4=""

. . . . Set Data=^Pets(S1,S2,S3,S4)

. . . . Set Age=$P(Data,"^",1)

. . . . Set Adoption=$P(Data,"^",2)

. . . . If Adoption=1 Set Available="is"

. . . . If Adoption=0 Set Available="is not"

. . . . Write !,S1," named ",S4,", Breed ",S2,", Sex ",S3

. . . . Write ", ",Age," years old, ",Available," available for adoption"

;2. Traverse down the Global using $Query and display the

;subscripts on the left side of the equals sign and the data

;on the right side of the equals sign

Set Pets="^Pets"

For Do Quit:Pets="" ;2 spaces after the For and Do

. Set Pets=$Q(@Pets) Quit:Pets=""

. Write !,Pets," = ",@Pets

;3. Lock and unlock the Pets Global at the top level

Lock ^Pets ; lock

Lock ; unlock

; OR

Lock +^Pets ; lock

Lock -^Pets ; unlock

;4. Merge the Pets Global into a Pets2 Global

Merge ^Pets2=^Pets

Example ‑ Answers to the Exercises using Command Structure Two for Traversing a Global

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

;1. Traverse down all four subscripts of the Pets Global using

;$Order and display each subscript and both pieces of data.

Set S1="" For Set S1=$O(^Pets(S1)) Quit:S1="" Do ;two spaces after the For and before the Do

. Set S2="" For Set S2=$O(^Pets(S1,S2)) Quit:S2="" Do

. . Set S3="" For Set S3=$O(^Pets(S1,S2,S3)) Quit:S3="" Do

. . . Set S4="" For Set S4=$O(^Pets(S1,S2,S3,S4)) Quit:S4="" Do

. . . . Set Data=^Pets(S1,S2,S3,S4)

. . . . Set Age=$P(Data,"^",1)

. . . . Set Adoption=$P(Data,"^",2)

. . . . If Adoption=1 Set Available="is"

. . . . If Adoption=0 Set Available="is not"

. . . . Write !,S1," named ",S4,", Breed ",S2,", Sex ",S3

. . . . Write ", ",Age," years old, ",Available," available for adoption"

;2. Traverse down the Global using $Query and display the

;subscripts on the left side of the equals sign and the data

;on the right side of the equals sign

Set Pets="^Pets"

For Set Pets=$Q(@Pets) Quit:Pets="" Do

. Write !,Pets," = ",@Pets

;3. Lock and unlock the Pets Global at the top level

;Same as prior example

;4. Merge the Pets Global into a Pets2 Global

;Same as prior example

Example ‑ Answers to the Exercises using Command Structure Three for Traversing a Global

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

;1. Traverse down all four subscripts of the Pets Global using

;$Order and display each subscript and both pieces of data.

Set (S1,S2,S3,S4)=""

For Set S1=$O(^Pets(S1)) Quit:S1="" Do ;two spaces after the For and before the Do

. For Set S2=$O(^Pets(S1,S2)) Quit:S2="" Do

. . For Set S3=$O(^Pets(S1,S2,S3)) Quit:S3="" Do

. . . For Set S4=$O(^Pets(S1,S2,S3,S4)) Quit:S4="" Do

. . . . Set Data=^Pets(S1,S2,S3,S4)

. . . . Set Age=$P(Data,"^",1)

. . . . Set Adoption=$P(Data,"^",2)

. . . . If Adoption=1 Set Available="is"

. . . . If Adoption=0 Set Available="is not"

. . . . Write !,S1," named ",S4,", Breed ",S2,", Sex ",S3

. . . . Write ", ",Age," years old, ",Available," available for adoption"

2. Traverse down the Global using $Query and display the

;subscripts on the left side of the equals sign and the data

;on the right side of the equals sign

;Same as prior example

;3. Lock and unlock the Pets Global at the top level

;Same as prior example

;4. Merge the Pets Global into a Pets2 Global

;Same as prior example

Example ‑ Answers to the Exercises using Command Structure Four for Traversing a Global, the While command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

;1. Traverse down all four subscripts of the Pets Global using

;$Order and display each subscript and both pieces of data.

Set S1=$O(^Pets(""))

While (S1'="") {

Set S2=$O(^Pets(S1,""))

While (S2'="") {

Set S3=$O(^Pets(S1,S2,""))

While (S3'="") {

Set S4=$O(^Pets(S1,S2,S3,""))

While (S4'="") {

Set Data=^Pets(S1,S2,S3,S4)

Set Age=$P(Data,"^",1)

Set Adoption=$P(Data,"^",2)

If Adoption=1 Set Available="is"

If Adoption=0 Set Available="is not"

Write !,S1," named ",S4,", Breed ",S2,", Sex ",S3

Write ", ",Age," years old, ",Available," available for adoption"

Set S4=$O(^Pets(S1,S2,S3,S4))

}

Set S3=$O(^Pets(S1,S2,S3))

}

Set S2=$O(^Pets(S1,S2))

}

Set S1=$O(^Pets(S1))

}

;2. Traverse down the Global using $Query and display the

;subscripts on the left side of the equals sign and the data

;on the right side of the equals sign

Set Pets="^Pets"

Set Pets=$Q(@Pets)

While (Pets'="") {

Write !,Pets," = ",@Pets

Set Pets=$Q(@Pets)

}

;3. Lock and unlock the Pets Global at the top level

;Same as prior example

;4. Merge the Pets Global into a Pets2 Global

;Same as prior example

Example ‑ Answers to the Exercises using Command Structure Five for Traversing a Global, the Do While command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

;1. Traverse down all four subscripts of the Pets Global using

;$Order and display each subscript and both pieces of data.

Set S1="" Do {

Set S1=$O(^Pets(S1)) Quit:S1=""

Set S2="" Do {

Set S2=$O(^Pets(S1,S2)) Quit:S2=""

Set S3="" Do {

Set S3=$O(^Pets(S1,S2,S3)) Quit:S3=""

Set S4="" Do {

Set S4=$O(^Pets(S1,S2,S3,S4)) Quit:S4=""

Set Data=^Pets(S1,S2,S3,S4)

Set Age=$P(Data,"^",1)

Set Adoption=$P(Data,"^",2)

If Adoption=1 Set Available="is"

If Adoption=0 Set Available="is not"

Write !,S1," named ",S4,", Breed ",S2,", Sex ",S3

Write ", ",Age," years old, ",Available," available for adoption"

} While S4'=""

} While S3'=""

} While S2'=""

} While S1'=""

;2. Traverse down the Global using $Query and display the

;subscripts on the left side of the equals sign and the data

;on the right side of the equals sign

Set Pets="^Pets"

Do {

Set Pets=$Q(@Pets) Quit:Pets=""

Write !,Pets," = ",@Pets

} While Pets'=""

;3. Lock and unlock the Pets Global at the top level

;Same as prior example

;4. Merge the Pets Global into a Pets2 Global

;Same as prior example

*“We are not satisfied, until you are not satisfied.”*

*– www.despair.com*

# Commands Revisited

Example ‑ The Basic Read command

Read X

My dog has fleas

Write X

My dog has fleas

Read "Prompt: ",X ;display "Prompt: "

Prompt: My dog has fleas

Write X

My dog has fleas

Example ‑ Read and convert to uppercase

Read X

My dog has fleas

Set X=$ZCVT(X,"U") ;convert to uppercase

Write X

MY DOG HAS FLEAS

Example ‑ The Basic Write command

Set X="ABC"

Write X

ABC

Example ‑ Read or Write with carriage return, line feed

Read !!,"Prompt: ",X

<carriage return, line feed>

<carriage return, line feed>

Prompt: My dog has fleas

Write !,X,!

<carriage return, line feed>

My dog has fleas

<carriage return, line feed>

Example ‑ Read or Write with form feed

Read #,"Prompt: ",X

<Form Feed inserted here>

Prompt: My dog has fleas

Write #,X,!

<Form Feed inserted here>

My dog has fleas

Example ‑ Read or Write advancing a number of spaces

Read ?10,"Prompt: ",X

Prompt:

Write ?10,"New",?20,"Title"

New Title

Example ‑ Improper use of the Tab in the Write command

Write ?10,"New",?10,"Title"

NewTitle

Example ‑ Read command Timer

Read X:5

Example ‑ Read a specified number of characters

Read X#6

My dog<character acceptance is stopped here>

Write !,X

My dog

Write the code that will:

* Read a value into the variable X, with the prompt "Please enter value:"
* Read a value into X after advancing to the next page
* Read a value into X after advancing 5 lines
* Read a value into X from column 20
* Read a value into X, if after 5 seconds no value is entered, continue
* Read a value into X, but only accept 5 characters
* Write "My dog’s name is Teddy" after advancing to the next page
* Write "My cat’s name is fluffy" after advancing 5 lines
* Write "I have too many pets" after moving across the page 10 spaces

### Advanced Read and Write command Exercise Answers

Example ‑ Advanced Read and Write command Exercise Answers

Read a value into the variable X, with the prompt "Please enter value:"

Read "Please enter value:",X

Read a value into X after advancing to the next page.

Read #,X

Read a value into X after advancing 5 lines.

Read !!!!!,X

Read a value into X from column 20.

Read ?20,X

Read a value into X, if after 5 seconds no value is entered, continue.

Read X:5

Read a value into X, but only accept 5 characters.

Read X#5

Write "My dog’s name is Teddy" after advancing to the next page.

Write #,"My dog’s name is Teddy"

Write "My cat’s name is fluffy" after advancing 5 lines.

Write !!!!!,"My cat’s name is fluffy"

Write "I have too many pets" after moving across the page 10 spaces.

Write ?10,"I have too many pets"

Example ‑ If and Else commands in the older style of MUMPS

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set X=1

If X=1 Write !,"X=1"

Else Write !,"X is not = 1" ;Else must be followed by two spaces

Example ‑ If and Else commands in Structured Code.

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

; First Method

Set X=1

If X=1 {

Write !,"X=1"

}

Else {

Write !,"X is not = 1"

}

; Second Method

Set X=1

If X=1 {Write !,"X=1"}

Else {Write !,"X is not = 1"}

; Third Method

Set X=1

If X=1 {Write !,"X=1"} Else {Write !,"X is not = 1"}

Example ‑ Nested If and Else commands in Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

; First Method

Set X=1,Y=2

If X=1 {

Write !,"X=1"

If Y=2 {

Write !,"Y=2"

}

Else {

Write !,"Y is not = 2"

}

}

Else {

Write !,"X is not = 1"

If Y=2 {

Write !,"Y=2"

}

Else {

Write !,"Y is not = 2"

}

}

; Second Method

Set X=1,Y=2

If X=1 {Write !,"X=1"

If Y=2 {Write !,"Y=2"}

Else {Write !,"Y is not = 2"}

}

Else {Write !,"X is not = 1"

If Y=2 {Write !,"Y=2"}

Else {Write !,"Y is not = 2"}

}

; Third Method

Set X=1,Y=2

If X=1 {Write !,"X=1" If Y=2 {Write !,"Y=2"} Else {Write !,"Y is not = 2"}}

Else {Write !,"X is not = 1" If Y=2 {Write !,"Y=2"} Else {Write !,"Y is not = 2"}}

Example ‑ If, ElseIf and Else commands in Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

; First Method

Set X=1

If X=1 {

Write !,"X=1"

}

ElseIf X=2 {

Write !,"X=2"

}

Else {

Write !,"X not = 1 or 2"

}

; Second Method

Set X=1

If X=1 {Write !,"X=1"}

ElseIf X=2 {Write !,"X=2"}

Else {Write !,"X not = 1 or 2"}

Example ‑ If, (multiple) ElseIf and Else command in Structured Code.

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Day=1

If Day=1 {Write !,"Today is Sunday"}

ElseIf Day=2 {Write !,"Today is Monday"}

ElseIf Day=3 {Write !,"Today is Tuesday"}

ElseIf Day=4 {Write !,"Today is Wednesday"}

ElseIf Day=5 {Write !,"Today is Thursday"}

ElseIf Day=6 {Write !,"Today is Friday"}

ElseIf Day=7 {Write !,"Today is Saturday"}

Else {Write !,"I don't know what day it is!"}

Example ‑ Using the If command to verify user input

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

AskUser ;

Read !,"Please enter 'Y'es or 'N' ",Answer ;ask user for answer

Set Answer=$ZCVT(Answer,"U") ;convert to upper case

If $E(Answer)'="Y",($E(Answer)'="N") Goto AskUser ;$E looks at the

;first char

Example ‑ Using the While command to verify user input

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Answer=""

While $E(Answer)'="Y",($E(Answer)'="N") {

Read !,"Please enter 'Y'es or 'N' ",Answer ;ask user for answer

Set Answer=$ZCVT(Answer,"U") ;convert to upper case

}

Example ‑ Using the Do While command to verify user input

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Answer=""

Do {

Read !,"Please enter 'Y'es or 'N' ",Answer ;ask user for answer

Set Answer=$ZCVT(Answer,"U") ;convert to upper case

} While $E(Answer)'="Y",($E(Answer)'="N")

Example ‑ Do While command with embedded Quit command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Answer=""

Do {

Read !,"Please enter 'Y'es or 'N' ",Answer ;ask user for answer

If Answer="" Q

Set Answer=$ZCVT(Answer,"U") ;convert to upper case

} While $E(Answer)'="Y",($E(Answer)'="N")

# New Command

Example ‑ New command used without variables

New ; all variables created will be new

Set X=1

Quit ; New command cancelled

Example ‑ Newcommand Routine

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ; New command Routine

Start

Set X=1,Y=2,Z=3

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Do SubRoutine

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit

SubRoutine ; SubRoutine

New ; "New" all variables

Set X=100,Y=200,Z=300

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit ; All variables revert back to their original value

Example ‑ Running the Newcommand Routine

Do ^Newcommand

X: 1 **;**first X,Y,Z are set to 1,2,3 respectively

Y: 2

Z: 3

X: 100 **;**within SubRoutine, X,Y,Z are reset to 100,200,300 respectively

Y: 200

Z: 300

X: 1;after the quit command in the Subroutine, X,Y,X revert

Y: 2

Z: 3

Example ‑ New command used with Arrays

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ; New command Routine

Start ;

Set Array(1)=1

Set Array(2)=2

Set Array(3)=3

Write !!,"Array(1): ",Array(1)

Write !,"Array(2): ",Array(2)

Write !,"Array(3): ",Array(3)

Do SubRoutine

Write !!,"Array(1): ",Array(1)

Write !,"Array(2): ",Array(2)

Write !,"Array(3): ",Array(3)

Quit

SubRoutine

New ;"New" all variables and arays

Set Array(1)=100

Set Array(2)=200

Set Array(3)=300

Write !!,"Array(1): ",Array(1)

Write !,"Array(2): ",Array(2)

Write !,"Array(3): ",Array(3)

Quit ;All variables and arrays revert back to their original value

Example ‑ Running the Newcommand Routine

Do ^Newcommand

Array(1): 1 ;first Array(1),(2),(3) are set to 1,2,3 respectively

Array(2): 2

Array(3): 3

Array(1): 100 ;within SubRoutine, Array(1),(2),(3) are reset to 100,200,300

Array(2): 200

Array(3): 300

Array(1): 1 ;after the quit command in Subroutine, Array(1),(2),(3) revert

Array(2): 2

Array(3): 3

Example ‑ Calling with parameters (passed by value) creates an implicit New

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ; New command Routine  
 Start ;

Set X=1,Y=2,Z=3

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Do SubRoutine**(X,Y,Z)** ;X,Y,Z Parameter Passing

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit

;

SubRoutine**(X,Y,Z) ;**Subroutine with Parameter Passing

Set X=100,Y=200,Z=300

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit ;All variables revert back to their original value

Example ‑ Running the Newcommand Routine with Parameter Passing

Do ^Newcommand

X: 1 ;first X,Y,Z are set to 1,2,3 respectively

Y: 2

Z: 3

X: 100 ;within SubRoutine, X,Y,Z are reset to 100,200,300 respectively

Y: 200

Z: 300

X: 1 ;after the quit command in the Subroutine, X,Z,X revert

Y: 2

Z: 3

Example ‑ Calling with parameters - passed by reference

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ; New command Routine  
 Start ;

Set X=1,Y=2,Z=3

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Do SubRoutine(.X,.Y,.Z) ;X,Y,Z Parameter Passing (by reference)

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit

;  
 SubRoutine(X,Y,Z) ;Subroutine with Parameter Passing (by reference)

Set X=100,Y=200,Z=300

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit ;All variables do not revert back

Example ‑ Running Newcommand Routine

Do ^Newcommand

X: 1 ;first X,Y,Z are set to 1,2,3 respectively

Y: 2

Z: 3

X: 100 ;within SubRoutine, X,Y,Z are reset to 100,200,300 respectively

Y: 200

Z: 300

X: 100 ;after the quit command in Subroutine, X,Y,Z do not revert

Y: 200

Z: 300

Example ‑ New command used with Inline Do command or Implicit Quit

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ;New command Routine  
 Start ;

Set X=1,Y=2,Z=3

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

;

Do                        ;inline Do

. New

. Set X=100,Y=200,Z=300

. Write !!,"X: ",X

. Write !,"Y: ",Y

. Write !,"Z: ",Z

; ;implicit quit

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit

Example ‑ Running Newcommand Routine

Do ^Newcommand

X: 1 ;first X,Y,Z are set to 1,2,3 respectively

Y: 2

Z: 3

X: 100 ;within SubRoutine, X,Y,Z are reset to 100,200,300 respectively

Y: 200

Z: 300

X: 1 ;after the Implicit quit in Subroutine, X,Y,Z do revert

Y: 2

Z: 3

Example ‑ New command used with variables

New X,Y

Example ‑ Newcommand Routine

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ; New command Routine

Start

Set X=1,Y=2,Z=3

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Do SubRoutine

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit

SubRoutine ;

New X,Y ; "New" X and Y variables only

Set X=100,Y=200,Z=300

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit ; Only X and Y variables revert back to their

; original value

Example ‑ Running the Newcommand Routine

Do ^Newcommand

X: 1 ;first X,Y,Z are set to 1,2,3 respectively

Y: 2

Z: 3

X: 100 ;within SubRoutine, X,Y,Z are reset to 100,200,300 respectively

Y: 200

Z: 300

X: 1 ;after the quit command in the Subroutine, only X and Y revert

Y: 2

Z: 300

Example ‑ New command used with variables

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

SubroutineA ;

New X,Y,Z,A,B,D

Set X=1

Set Y=2

Set Z=3

Set A=4

Set B=5

Set D=6

Quit

Example ‑ New command used with variables

SubroutineA ;

New X,Y,Z,A,B,D

Set W=0

Set X=1

Set Y=2

Set Z=3

Set A=4

Set B=5

Set C=4

Set D=6

Quit

Example ‑ New command used with Variables and Arrays

New X ;if X is an array this command is legal

New X(1) ;this command is illegal

Example ‑ New command exhausts Memory

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For I=1:1:10000 New X Do  
 . Set X=I  
 . Write !,X  
<FRAMESTACK>

Example ‑ New command exhausts Memory - Corrected

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For I=1:1:10000 Do

. New X

. Set X=I

. Write !,X

Example ‑ New command used with variables in parenthesis

New (X,Y)

Example ‑ Newcommand Routine

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ; New command Routine

Start

Set X=1,Y=2,Z=3

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Do SubRoutine

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit

SubRoutine ;

New (X,Y) ; "New" all variables except X and Y

Set X=100,Y=200,Z=300

Write !!,"X: ",X

Write !,"Y: ",Y

Write !,"Z: ",Z

Quit ; All variables except X and Y variables revert

; back to their original value

Example ‑ Running the Newcommand Routine

Do ^Newcommand

X: 1 ;first X,Y,Z are set to 1,2,3 respectively

Y: 2

Z: 3

X: 100 ;within SubRoutine, X,Y,Z are reset to 100,200,300 respectively

Y: 200

Z: 300

X: 100 ;after the quit command in the Subroutine, X and Y do not revert

Y: 200

Z: 3

Example ‑ New command used improperly

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Newcommand ;  
 Loop ;

N X,Y

Set X=$G(X)+1

Set Y=$G(Y)+1

Set Z=$G(Z)+1

G Loop

Example ‑ Examples of not using a New command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For I=1:1:10 D Proc1

Quit

Proc1 ;

Write !,"Processing I of: ",I

Do Proc2

Quit

Proc2 ;

For I=1:1:5 {

Write !,"I number: ",I

}

Quit

Example ‑ Solution to assuming a variable is not in use Variable

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

New I ;New I

For I=1:1:10 Do Proc1

Quit

Proc1 ;

Write !,"Processing I of: ",I

Do Proc2

Quit

Proc2 ;

New I

For I=1:1:5 {

Write !,"I number: ",I ;New I

}

Quit

“The human brain starts working the moment you are born and never stops until you stand up to speak in public.”

- Sir George Jessel

# Pattern Matching

Example ‑ Validate an American Social Security Number with Pattern Matching

If SSN?3N1"-"2N1"-"4N Write !,"Valid Social Security Number"

Table ‑ *Pattern Matching* Codes

|  |  |
| --- | --- |
| Code | Meaning |
| A | Alphabetic characters, A thru Z, uppercase and lowercase |
| U | Uppercase characters |
| L | Lowercase characters |
| N | Numeric digits |
| P | Punctuation characters |
| C | Control Character |
| E | Any Character |

Table ‑ Pattern Length

|  |  |
| --- | --- |
| Length | Meaning |
| 4 | exactly four |
| 1.4 | from one to four |
| .4 | up to four |
| 4. | at least 4 |
| . | any number including zero |

Example ‑ Validating Alpha Characters

; Pattern Matching "A" – alpha characters, uppercase and lowercase

Set Data="ABCDEabcde" ;Data contains all alpha characters

Set Pattern=".A" ;Pattern of any number of alpha characters

Write Data?@Pattern ;checks for all alpha characters

1 ;if the data conforms to the pattern

; - a 1 (true) is returned

Set Data="ABCDEabcde" ;Data contains all alpha characters

If Data?.A Write !,"The data conforms to the pattern"

The data conforms to the pattern

Example ‑ Validating Uppercase Characters

; Pattern Matching "U" – uppercase characters

Set Data="ABCDE" ;Data contains all uppercase characters

Set Pattern=".U" ;Pattern of any number of uppercase characters

Write Data?@Pattern ;checks for all uppercase characters

1;if the data conforms to the pattern

; - a 1 (true) is returned

Set Data="ABCDE" ;Data contains uppercase characters

If Data?.U Write !,"The data conforms to the pattern"

The data conforms to the pattern

Example ‑ Validating a Capitalized Word

Set Data="California" ;Data contains a capitalized word

Set Pattern="1U.L" ;Pattern for capitalized word

Write Data?@Pattern

1;if the data conforms to the pattern

; - a 1 (true) is returned

Example ‑ Validating Numeric Digits

; Pattern Matching "N" – Numeric Digits

Set Data="1234" ;Data contains all Numeric Digits

Set Pattern=".N" ;Pattern for any number of Numeric Digits

Write Data?@Pattern ;checks for all Numeric Digits

1;if the data conforms to the pattern

; - a 1 (true) is returned

Set Data="1234" ;Data contains all Numeric Digits

If Data?.N Write !,"The data conforms to the pattern"

The data conforms to the pattern

Example ‑ Validating a Numeric with Two Decimal Positions

Set Data="12.34" ;Data contains Numerics, decimal and 2 numerics

Set Pattern=".N1"".""2N" ;Pattern for Numerics, decimal and 2 numerics

Write Data?@Pattern

1;if the data conforms to the pattern

;a 1 (true) is returned

Set Data="12.34" ;Data contains Numerics, decimal and 2 numerics

If Data?.N1"."2N Write !,"The data conforms to the pattern"

The data conforms to the pattern

Example ‑ Validating Punctuation Characters

; Pattern Matching "P" – Punctuation characters

Set Data="., ;" ;Data contains all Punctuation characters

Set Pattern=".P" ;Pattern for any number of Punctuation characters

Write Data?@Pattern ;checks for all Punctuation characters

1;if the data conforms to the pattern

; - a 1 (true) is returned

Set Data="., ;" ;Data contains all Punctuation characters

If Data?.P Write !,"The data conforms to the pattern"

The data conforms to the pattern

Example ‑ Search a string for a substring

Set String="Jack and Jill went down the hill."

Set Pattern=".E1.P1""Jill""1P.E" ;search for “Jill”

Write String?@Pattern

1

Set String="Jack and Jill went down the hill."

If String?.E1P1"Jill"1P.E Write !,"The string is found"

The string is found

Example ‑ Validating Control Characters

; Pattern Matching "C" – Control Characters

; The $Char System Supplied Function below produces the Bell (7),

; Backspace (8), Linefeed (10), Formfeed (12) and Carriage Return (13)

; Control Characters

Set Data=$C(7)\_$C(8)\_$C(10)\_$C(12)\_$C(13) ;Data contains Control Characters

Set Pattern=".C" ;Pattern for any number of Control Characters

Write Data?@Pattern ;checks for all Control Characters

1;if the data conforms to the pattern

; - a 1 (true) is returned

Set Data=$C(7)\_$C(8)\_$C(10)\_$C(12)\_$C(13) ;Data contains Control Characters

If Data?.C Write !,"The data conforms to the pattern"

The data conforms to the pattern

Example ‑ *Pattern Matching* Numeric Digits using Specific Lengths

Set Data="12"

If Data?1.5N Write !,"Data has numeric digits from 1 to 5 in length"

Data has numeric digits from 1 to 5 in length6

Set Data="54321"

If Data?.10N Write !,"Data has numeric digits from 0 to 10 in length"

Data has numeric digits from 0 to 10 in length

Set Data="123454321"

If Data'?1.5N Write !,"Data does not have numeric digits from 1 to 5 in length"

Data does not have numeric digits from 1 to 5 in length

Example ‑ *Pattern Matching* Alpha Characters using Specific Lengths

Set Data="ABC"

If Data?1.3A Write !,"Data has Alpha characters from 1 to 3 in length"

Data has Alpha characters from 1 to 3 in length

Set Data="ABCDE"

If Data'?1.3A Write !,"Data does not have Alpha characters from 1 to 3 in length"

Data does not have Alpha characters from 1 to 3 in length

Example ‑ *Pattern Matching* using Parenthesis or Logical “OR”

Set Name="Jack"

If Name?1(1"Jack",1"Jill",1"Fred") Write 1 ;Name must be Jack or Jill or Fred

1

Set Name="Jill"

If Name?1(1"Jack",1"Jill",1"Fred") Write 1 ;Name must be Jack or Jill or Fred

1

Set Name="Fred"

If Name?1(1"Jack",1"Jill",1"Fred") Write 1 ;Name must be Jack or Jill or Fred

1

Example ‑ *Pattern Matching* using Parenthesis or Logical “OR” for Zip Code (Postal Code)

Set Zip="16063"

If Zip?1(5N,5N1"-"4N) Write 1

1

Set Zip="16063-3015"

If Zip?1(5N,5N1"-"4N) Write 1

1

Example ‑ Numeric *Pattern Matching*

Set value=1234

If value?1N.N Write !,"Valid numeric" ;length from 1 numeric to any

Valid numeric ;length of numerics

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set value="abc"

If value'?1N.N Write !,"Not numeric" ;"Not" numeric of any length

Not numeric

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set value=1234

If value?2N.4N Write !,"Valid numeric" ;numeric of length 2 to 4

Valid numeric

Example ‑ Zip Code (Postal Code) *Pattern Matching*

; zip code in nnnnn or nnnnn-nnnn format

set pattern="5N"

If $l(zip)=10 set pattern="5N1""-""4N"

If zip?@pattern Write !,"Valid zip code" ;valid zip code

If zip'?@pattern Write !,"Invalid zip code" ;invalid zip code

Example ‑ Date *Pattern Matching*

; date format in mm/dd/yy or mm/dd/yyyy format

If date?1.2N1"/"1.2N1"/"2.4N Write !,"Valid date format" ;valid format

If date'?1.2N1"/"1.2N1"/"2.4N Write !,"Invalid date format" ;invalid format

Example ‑ Complex Date *Pattern Matching*

;date format in mm/dd/yy or mm-dd-yy or mm.dd.yy

If date?1.2N1(1"."1.2N1".",1"/"1.2N1"/",1"-"1.2N1"-")2.4N W !,"valid format"

If date'?1.2N1(1"."1.2N1".",1"/"1.2N1"/",1"-"1.2N1"-")2.4N Write !,"invalid format"

Example ‑ Phone Number *Pattern Matching*

Set pattern="3N1""-""4N"

Set phone=$TR(phone," ") ;remove spaces

If $l(phone)=14 Set pattern="1N1""(""3N1"")""3N1""-""4N"

If phone?@pattern Write !,"Valid phone format" ;valid phone

If phone'?@pattern Write !,"Invalid phone format" ;invalid phone number

;phone format nnn-nnnn or nnn.nnn.nnnn or (nnn)nnn-nnnn

If (PH?3N1"-"4N)!(PH?3N1"."3N1"."4N)!(PH?1"("3N1")"3N1"-"4N) Write !,"valid phone"

Example ‑ Dollar Number *Pattern Matching* Examples

Set dol="100,000,000.00"

If (dol?.3N1"."2N)!(dol?.3N1","3N1"."2N)!(dol?.3N1","3N1","3N1"."2N) Write “Valid”

Example ‑ Answers to Exercises on *Pattern Matching*

; Write a pattern that could be used to verify a name in the format:

; LAST, FIRST MI. All letters need to be uppercase.

Set Name="DOE, JOHN M."

If Name?.U1", ".U1" "1U1"." W "Format Valid"

Format Valid

; Modify the previous pattern to ensure that each of words is capitalized:

; Last, First Mi. Allow for the Middle initial to be either a 1 character

; initial or an entire name.

Set Name="Doe, John Michael"

If Name?1U.L1", "1U.L1" "1U.L Write "Format Valid"

Format Valid

; Write a pattern that could be used to verify an insurance number of the

; format: 1 uppercase alpha, 6 numbers, 1 dash followed by 3 to 7 numbers,

; U657823-123.

Set Number="U657823-123"

If Number?1U6N1"-"3.7N Write "Format Valid"

Format Valid

; Write a pattern that could be used to verify a dollar amount include

; a dollar sign, dollars, cents, decimal point and commas.

Set AMT1="$25.50"

Set AMT2="$23,125.50"

Set AMT3="$6,789,325.50"

If $L(AMT1)<8,AMT1?1"$"1.3N1"."2N Write "Format Valid"

Format Valid

If $L(AMT2)<12,AMT2?1"$"1.3N1","3N1"."2N Write "Format Valid"

Format Valid

If $L(AMT3)<16,AMT3?1"$"1.3N1","3N1","3N1"."2N Write "Format Valid"

Format Valid

*Customer: “My mouse is at the edge of my desk, what should I do?”*

*Tech Support: ”Get a bigger desk!”*

*-Scott Adams*

# Comparison Operators

Table ‑ Table of Comparison Operators

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Character** | **Type of Operands** | **Description** |
| Unary NOT | **'** | Numeric digits | Reverses the value of the operand |
| Binary Greater Than | **>** | Numeric digits | Tests whether the left operand is numerically greater than the right operand |
| Binary Less Than | **<** | Numeric digits | Test whether the right operand is numerically greater than the left operand |
| Binary And | & and && | Numeric digits and Alphanumeric Text | Test whether both operands are true |
| Binary Or | ! and || | Numeric digits and Alphanumeric Text | Test whether either operand is true |
| Binary Equal To | **=** | Numeric digits and Alphanumeric Text | Tests two operands for equality\* |
| Binary Contains | **[** | Alphanumeric Text | Test whether the characters in the right operand is a substring of the left operand |
| Binary Follows | **]** | Alphanumeric Text | Tests whether the characters in the left operand comes after the characters in the right operand according to the ASCII collating sequence. |
| Binary Sorts After | **]]** | Numeric digits and Alphanumeric Text | Tests whether the left operand sorts after the right operand in subscript collating sequence. |

Example ‑ True and False respectively

Set X=1

If X Write "True" ;1 is always true and 0 is always false

True

Set X=0

If X Write "True" ;1 is always true

<>

If 'X Write "False" ;0 is always false

False

Set X=10

If X Write "True" ;Any number other than 0 is always true

True

Set X=-10

If X Write "True" ;Any number other than 0 is always true

True

Example ‑ Alphanumeric text True and False

Set X="1"

If X Write "True" ;1 is always true even in Alphanumeric text

True

Set X="0"

If X Write "True" ;1 is always true even in Alphanumeric text

<>

If 'X Write "False ;0 is always false even in Alphanumeric text

False

Set X="1ABC"

If X Write "True" ;The 1 is used and the rest of the text is ignored

True

Set X="0ABC"

If X Write "True" ;The first numeric is used and the

<>

;rest of the text is ignored

If 'X Write "False

False

Set X="ABC1" ;If the numeric is not first in the text it is

If X Write "True" ;ignored and any text comes back as 0 or false

<>

Set X="ABC0"

If X Write "True" ;If the numeric is not first in the text it is

<> ;ignored and any text comes back as 0 or *false*

If 'X Write "False

False

Example ‑ Unary NOT Operator used on numeric digits

Set X=1

If X Write "True" ;1 is always true and 0 is always false

True

Set X=0

Write 'X ;X is 0, so "Not X" is 1 or true

1

Set X=1

Write 'X ;X is 1, so "Not X" is 0 or false

0

Set X=0

If 'X Write "True" ;X is 0 so "Not X" is 1 and 1 is true

True

Set X=5 ;Any "Non Zero" value is true

If X Write "True"

True

Set X=5

Write 'X ;"Not X", or "Not 5", is 0 or false

0

Example ‑ Unary NOT Operator used on Alphanumeric text

Set X="1"

If X Write "True" ;1 is always true even with Alphanumeric text

True

Set X="0"

Write 'X ;X is 0, so "Not X" is 1 even with Alphanumeric

1 ;test

Set X="1"

Write 'X ;X is 1, so "Not X" is 0 even with Alphanumeric

0

Set X="0ABC" ;The first numeric is used and the

If 'X Write "True" ;rest of the text is ignored

True

Set X="5ABC" ;"Not X", or "Not 5", is 0 or false

Write 'X ;The first numeric is used and the rest is ignored

0

Set X="ABC0" ;If the numeric is not the first character in

If 'X Write "True" ;the text it is ignored and any text comes back

True ;as 0 or false. Since the If command uses a "Not"

;it comes back as true.

Table ‑ $Data table of returned values

|  |  |  |
| --- | --- | --- |
| What $Data Returns | Array Item has  Value | Array Item has  Descendants |
| 0 | No | No |
| 1 | Yes | No |
| 10 | No | Yes |
| 11 | Yes | Yes |

Example ‑ True and False with $Data

Set A(1)="data"

Set A(2)=""

Set A(3,1)="data"

Set A(4)="data"

Set A(4,1)="data"

If $D(A(1)) Write "True" ;this will return a 1 or true

True

If '$D(A(1)) Write "True" ;this is not true

<>

If $D(A(3)) Write "True" ;this will return a 10 or true

True

If '$D(A(3)) Write "True" ;this is not true

<>

If $D(A(4)) Write "True" ;this will return a 11 or true

True

If '$D(A(4)) Write "True" ;this is not true

<>

If $D(A(5)) Write "True" ;this will return a 0 or false

<>

If '$D(A(5)) Write "True" ;this is true

True

If $D(A(2)) Write "True" ;this will return a 1 or true, even though

True;the data is blank or null

## Exercises on True, False, Not and $Data

For each of the expressions below, determine whether the variable X is true or false.

* Set X=1
* Set X=0
* Set X=10
* Set X=-10
* Set X="1"
* Set X="0"
* Set X="1ABC"
* Set X="0ABC"
* Set X="ABC1"
* Set X=’1
* Set X=’0
* Set X=5
* Set X=’5

For the following assume:

Set A(1)="data"

Set A(2)=""

Set A(3,1)="data"

Set A(4)="data"

Set A(4,1)="data"

* Set X=$D(A(1))
* Set X='$D(A(1))
* Set X=$D(A(3))
* Set X=$D(A(5))
* Set X=$D(A(2))

## Answers to Exercises on True, False, Not and $Data

Example ‑ Answers to Exercises on True, False, Not and $Data True and False with $Data

Set X=1 True

Set X=0 False

Set X=10 True

Set X=-10 True

Set X="1" True

Set X="0" False

Set X="1ABC" True

Set X="0ABC" False

Set X="ABC1" False

Set X=’1 False

Set X=’0 True

Set X=5 True

Set X=’5 False

For the following assume:

Set A(1)="data"

Set A(2)=""

Set A(3,1)="data"

Set A(4)="data"

Set A(4,1)="data"

Set X=$D(A(1)) True

Set X='$D(A(1)) False

Set X=$D(A(3)) True

Set X=$D(A(5)) False

Set X=$D(A(2)) True

Example ‑ Simple Numeric Comparisons

If 2>1 Write "True" ;2 is greater than 1

True

If 1<2 Write "True" ;1 is less than 2

True

If 1'>2 Write "True" ;1 is not greater than 2

True

If 2'<1 Write "True" ;2 is not less than 1

True

Example ‑ Binary Less Than and Binary Greater Than used on quoted numeric text

If "2">"1" Write "True" ;2 is greater than 1

True

If "1"<"2" Write "True" ;1 is less than 2

True

If "1"'>"2" Write "True" ;1 is not greater than 2

True

If "2"'<"1" Write "True" ;2 is not less than 1

True

Example ‑ Binary Less Than and Greater Than with variables that contain quoted numbers.

Set A="1"

Set B="2"

If B>A Write "True" ;B or 2 is greater than A or 1

True

If A<B Write "True" ;A or 1 is less than B or 2

True

If A'>B Write "True" ;A or 1 is not greater than B or 2

True

If B'<A Write "True" ;B or 2 is not less than A or 1

True

Example ‑ Binary Less Than and Greater Than used on alphanumeric text

Set A="A"

Set B="B"

If B>A Write "True" ;Any alphanumeric text is assumed to be 0

<>

If A<B Write "True" ;Any alphanumeric text is assumed to be 0

<>

If A'>B Write "True" ;"Not Greater Than" is the same as equal to or less than

True;and 0 is equal to 0

If B'<A Write "True" ;"Not Less Than" is the same as equal to or greater than

True;and 0 is equal to 0

Example ‑ Binary Less Than and Greater Than used on alphanumeric text

Set A="1A"

Set B="2B"

If B>A Write "True" ;As we saw before, the leading number is used and

True;the rest is ignored

If A<B Write "True" ;Ditto

True

If A'>B Write "True" ;Is 1 not greater than 2, yes it is

True

If B'<A Write "True" ;Is 2 not less than 1, yes it is

True

If B'>A Write "True" ;Is 2 not greater than 1, no

<>

If A'<B Write "True" ;Is 1 not less than 2, no

<>

## 

## Exercises on Binary Less Than and Greater Than

For each of the expressions below, determine whether the expression true or false.

* 2>1
* 1<2
* If 1'>2
* 2'<1
* If "2">"1"
* If "1"<"2"

For the following assume:

Set A="1"

Set B="2"

* B>A
* A'>B

For the following assume:

Set A="A"

Set B="B"

* B>A
* A<B
* A'>B
* B'<A

Example ‑ Answers to Exercises on Binary Less Than and Greater Than

2>1 True

1<2 True

If 1'>2 True

2'<1 True

If "2">"1" True

If "1"<"2" True

For the following assume:

Set A="1"

Set B="2"

B>A True

A'>B True

For the following assume:

Set A="A"

Set B="B"

B>A False

A<B False

A'>B True

B'<A True

Example ‑ Binary And using numbers

If 1&1 Write "True" ;Both 1 and 1 are true so the result is true

True

If 5&5 Write "True" ;Any non 0 value is considered true

True

If 1&0 Write "True" ;Both have to be true and 0 is not true

<>

If 1&'0 Write "True" ;Both have to be true and "Not 0" is true

True

If '1&'1 Write "True" ;Not 1 is 0, and "0 AND 0" is not true

<>

If 1,1 Write "True" ;Both 1 and 1 are true so the result is true

True

If 5,5 Write "True" ;Any non 0 value is considered true

True

If 1,0 Write "True" ;Both have to be true and 0 is not true

<>

If 1,'0 Write "True" ;Both have to be true and "Not 0" is true

True

If 1,0 Write "True" ;Both have to be true and 0 is not true

<>

If '1,'1 Write "True" ;Not 1 is 0, and "0 AND 0" is not true

<>

Example ‑ Binary And using variables

Set A=1

Set B=1

If A&A Write "True" ;Both A and B are true so the result is true

True

Set A=5

Set B=5

If A&B Write "True" ;Any non 0 value is considered true

True

Set A=1

Set B=0

If A&B Write "True" ;Both have to be true and B is not true

<>

Set A=1

Set B=0

If A&'B Write "True" ;Both have to be true and "Not B" is true

True

Set A=1

Set B=1

If A,B Write "True" ;Both A and B are true so the result is true

True

Set A=5

Set B=5

If A,B Write "True" ;Any non 0 value is considered true

True

Set A=1

Set B=0

If A,B Write "True" ;Both have to be true and B is not true

<>

Set A=1

Set B=0

If A,'B Write "True" ;Both have to be true and "Not B" is true

True

Example ‑ Binary Or using numbers

If 1!1 Write "True" ;If either literal is true the result is true

True

If 1!0 Write "True" ;If either literal is true the result is true

True

If 5!0 Write "True" ;Any non 0 value is considered true

True

If 0!0 Write "True" ;At least one operand needs to be true

<>

If 0!'0 Write "True" ;"Not 0" is considered true

True

If '1!'1 Write "True" ;Not 1 is 0, and "0 or 0" is not true

<>

Example ‑ Binary Or using variables

Set A=1

Set B=1

If A!B Write "True" ;If either variable is true the result is true

True

Set A=1

Set B=0

If 1!0 Write "True" ;If either variable is true the result is true

True

Set A=5

Set B=0

If A!B Write "True" ;Any non 0 variable is considered true

True

Set A=0

Set B=0

If A!B Write "True" ;At least one operand needs to be true

<>

Set A=0

Set B=0

If A!'B Write "True" ;"Not 0" is considered true

True

Set A=1

Set B=1

If 'A!'B Write "True" ;Not 1 is 0, and "0 or 0" is zero and not true

<>

Example ‑ Multiple Binary ORs and Binary ANDs

Set A=1

Set B=2

Set C=3

If A=1!B=2!C=3 Write "True"

<>

If A=1&B=2&C=3 Write "True"

<>

Example ‑ Multiple Binary ORs and Binary ANDs

Set A=1

Set B=2

Set C=3

If (A=1)!(B=2)!(C=3) Write "True"

True

If (A=1)&(B=2)&(C=3) Write "True"

True

## Exercises on Binary And and Binary Or

For each of the expressions, below and determine whether it is true or false.

* 1&1
* 5&5
* 1&0
* 1&'0
* '1&'1
* 1,1
* 1,0

For the following assume:

Set A=1

Set B=1

* A&B

For the following assume:

Set A=1

Set B=0

* A&B
* A&'B
* A,B
* A,'B
* 1!1
* 1!0
* 0!0
* 0!'0
* '1&'1

For the following assume:

Set A=0

Set B=0

* If A!'B

For the following assume:

Set A=1

Set B=1

* 'A&'B

For the following assume:

Set A=1

Set B=2

Set C=3

* A=1!B=2!C=3
* A=1&B=2&C=3
* (A=1)!(B=2)!(C=3)
* (A=1)&(B=2)&(C=3)

## Answers to Exercises on Binary And and Binary Or

Example ‑ Answers to Exercises on Binary And and Binary Or

1&1 True

5&5 True

1&0 False

1&'0 True

'1&'1 False

1,1 True

1,0 False

For the following assume:

Set A=1

Set B=1

A&B True

For the following assume:

Set A=1

Set B=0

A&B False

A&'B True

A,B False

A,'B True

For the following assume:

Set A=1

Set B=1

1!1 True

1!0 True

0!0 False

0!'0 True

'1&'1 False

For the following assume:

Set A=0

Set B=0

If A!'B True

For the following assume:

Set A=1

Set B=1

'A&'B False

For the following assume:

Set A=1

Set B=1

Set C=1

A=1!B=2!C=3 False

A=1&B=2&C=3 False

(A=1)!(B=2)!(C=3) True

(A=1)&(B=2)&(C=3) True

Example ‑ Equal sign as an assignment operator

Set A=1 ;set variable A to 1

Write !,A

1

Set B="Text" ;set variable B to Text

Write !,B

Text

Set C=1+2+3 ;set variable C to 6

Write !,C

6

Set ^Global(0)=6/3 ;set ^Global(0) to 2

Write ^Global(0)

2

Set D=^Global(0) ;set variable D to the data contained in ^Global(0) or 2

Write !,D

2

Set $E(E,3,4)=55 ;set the 3rd and 4th character E to 5

Write !,E

55

Set $P(F,"^",2)=10 ;set the second piece of variable F to 10

Write !,F

^10

Example ‑ Equal sign as a comparison operator between two operands

If 1=1 Write !,"Equal"

Equal

If 1'=1 Write !,"Equal"

<>

If "Text"="Text" Write !,"Equal"

Equal

If 1+2+3=1+2+3 Write !,"Equal"

Equal

If 1="1" Write "Equal"

Equal

If 1+1="2" Write "Equal"

Equal

If 1+1="1+1" Write "Equal" ;Quoting numbers does make a difference

<>

If "003"="3" Write !,"Equal"

<>

If 003=3 Write !,"Equal"

Equal

Set $E(E,3,4)=55

If E=E Write !,"Equal"

Equal

Set $P(F,"^",2)=10

If $P(F,"^",2)=$P(F,"^",2) Write !,"Equal"

Equal

Example ‑ Binary Contains

If "This is our Country"["Country" Write !,"Contains"

Contains

If "This is our Country"["is our" Write !,"Contains"

Contains

If "This is our Country"["this is" Write !,"Contains" ;cases must match

<>

If "This is our Country"'["ABC123" Write !,"Not Contains" ;not contains

Not Contains

If 002[2 Write !,"Contains"

Contains

If 2[002 Write !,"Contains" ;the 002 is reduced to 2

Contains

Example ‑ Binary Follows with alpha data

Write $Ascii("A") ;ASCII value of "A"

65

Write $Ascii("B") ;ASCII value of "B"

66

If "B"]"A" Write "True" ;ASCII value of "B" follows the ASCII value of "A"

True

Example ‑ Binary Follows Operator with numeric data

Write $Ascii(2)

50

Write $Ascii(1)

49

If 2]19 Write "True"

True

Example ‑ Setting the ^TMP Global

Set ^TMP("ABC")=""

Set ^TMP(1)=""

Set ^TMP(0)=""

Set ^TMP(-1)=""

Example 11‑25 uses four different subscripts in setting the ^TMP Global, when we display the Global, what do we see?

Example ‑ How a Global is sorted

D ^%G

Global ^TMP

^TMP(-1)=""

^TMP(0)=""

^TMP(1)=""

^TMP("ABC")=""

Example ‑ Binary Sort After Operator

I "-1"]]"" Write "True" ;"-1" Sorts After ""

True

I "0"]]"-1" Write "True" ;"0" Sorts After "-1"

True

I "1"]]"0" Write "True" ;"1" Sorts After "0"

True

I "ABC"]]"1" Write "True" ;"ABC" Sorts After "1"

True

# File Processing

Example ‑ Write to an External File

Line1:

WriteFile ;

Line2:

Set OutFile="FILE.TXT" ;name of out file – FILE.TXT

Line3:

Close OutFile ;close a file before you open it

Line4:

Open OutFile:"WNS":10 ;open the file for writing,

;WNS – write, new, stream

;timeout of 10 seconds

Line5:

If '$Test Write !,OutFile," cannot be opened." Quit ;cannot open

Line6:

Use OutFile ;Sets OutFile as the current device

Line7:

Write "First Record"

Line8:

Write !,"Second Record"

Line9:

Write !,"Third Record"

Line10:

Use 0 Write !,"End of Program reached"

Line11:

Quit

Example ‑ Read a file and display its records

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Line1:

ReadFile ;

Line2:

Set InFile="FILE.TXT" ;name of infile – FILE.TXT

Line3:

Close InFile ;close a file before you open it

Line4:

Open InFile:"R":10 ;open file for read, timeout 10 sec

Line5:

If '$Test Write !,InFile," cannot be opened." Quit

Line6:

Set InCount=0 ;init counter of records read

Line7:

Set X=$ZU(68,40,1) ;enable the $ZEOF special variable

Line7a:

Set system.Process.SetZEOF(1) ;enable the $ZEOF special variable

Line8:

Set EOF=0 Do { ;EOF is end of file switch

Line9:

Use InFile ;sets InFile as the Current Device

Line10:

Read InRecord ;read record from file

Line11:

If $ZEOF=-1 Set EOF=1 Quit ;$ZEOF=-1 when end of file reached

Line12:

Set X=$Increment(InCount) ;increment counter

Line13:

Use 0 Write !,InRecord ;display record

Line14:

} While EOF=0 ;read until end of file

Line15:

Use 0 Write !,InCount," Records read"

Line16:

Use 0 Write !,"End of File reached"

Line17:

Quit

Example ‑ Running Routine ^ReadFile

Do ^ReadFile

First Record

Second Record

Third Record

3 Records read

End of File reached

Example ‑ Read and Write a file

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

ReadAndWrite ;

Set InFile="FILE.TXT" ;name of infile – FILE.TXT

Set OutFile="FILE2.TXT" ;name of outfile – FILE2.TXT

Close InFile,OutFile ;close a file before you open it

Open InFile:"R":10 ;open the file for reading, timeout 10 sec

If '$Test Write !,InFile," cannot be opened." Quit ;cannot open file

Open OutFile:"WNS":10 ;open file for write, new, timeout of 10 sec

If '$Test Write !,OutFile," cannot be opened." Quit ;cannot open file

Set (InCount,OutCount)=0 ;init counter of records read and write

Set X=$ZU(68,40,1) ;enables the $ZEOF special variable

;Set system.Process.SetZEOF(1) ;enable the $ZEOF special variable

Set EOF=0 Do { ;EOF is the end of file switch

Use InFile ;sets InFile as the current device

Read InRecord ;read record from file

If $ZEOF=-1 Set EOF=1 Quit ;$ZEOF=-1 when end of file reached

Set X=$Increment(InCount) ;increment counter

Use 0 Write !,InRecord ;display record

Set X=$Increment(OutCount) ;increment counter

Use OutFile ;sets OutFile as the current device

Write InRecord,!

} While EOF=0 ;read until end of file

Use 0 Write !,InCount," Records read"

Use 0 Write !,OutCount," Records written"

Use 0 Write !,"End of File reached"

Quit

Example ‑ Running Routine ^ReadAndWrite

Do ^ReadAndWrite

First Record

Second Record

Third Record

3 Records read

3 Records written

End of File reached

Example ‑ Cycle through several files

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

CycleThruFiles ;

For File="FILE1.TXT","FILE2.TXT","FILE3.TXT" {

Open File:"WNS"

Close File

}

Set File=("FILE\*.TXT") ;use \* as a wildcard

Set File=$ZSearch(File)

Write !,File

Do {

Set File=$ZSearch("") ;use a blank parameter to get next file

If File="" Q

Write !,File

} While File'=""

Example ‑ Running Routine ^CycleThruFiles

Do ^CycleThruFiles

C:\Cache2010\mgr\user\FILE.TXT

C:\Cache2010\mgr\user\FILE1.TXT

C:\Cache2010\mgr\user\FILE2.TXT

C:\Cache2010\mgr\user\FILE3.TXT

Example ‑ Search multiple files for a specific string

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

SearchForString  
 For File="FILE1.TXT","FILE3.TXT","FILE5.TXT" { ;three empty files  
   Open File:"WNS"  
   Close File  
 }  
 For File="FILE2.TXT","FILE4.TXT","FILE6.TXT" { ;three files with "fleas"  
   Open File:"WNS"  
   Use File Write !,"My dog has fleas"  
   Close File  
 }  
  
 Set File=("FILE\*.TXT") ;set initial files to search  
 Set File=$ZSearch(File) ;set initial search  
 Do {  
 Set File=$ZSearch("") ;get next file  
 If File="" Quit ;end of file list  
    Open File:"R":10 ;open files  
    If '$Test Write !,File," cannot be opened." Quit  
    Set X=$ZU(68,40,1) ;set up $ZEOF  
    Set EOF=0 Do {   
      Use File ;set File as current device  
      Read Inrec ;read record  
      If $ZEOF=-1 Set EOF=1 Quit  
      If Inrec["fleas" {  
       Use 0 Write !,"File: "  
       Write File," contains string 'fleas'."  
      }  
   } While EOF=0  
  Close File  
 } While File'=""

Example ‑ Running Routine ^SearchForString

Do ^SearchForString

File: C:\Cache2010\mgr\user\FILE2.TXT contains string 'fleas'.

File: C:\Cache2010\mgr\user\FILE4.TXT contains string 'fleas'.

File: C:\Cache2010\mgr\user\FILE6.TXT contains string 'fleas'.

Example ‑ Finding a File

Set File=$ZSearch("FILE.TXT") ;Find a File that exists

Write !,File

C:\Cache2010\mgr\user\FILE.TXT

Set File=$ZSearch("FILEXXX.TXT") ;Find a file that does not exist

Write !,File

<>

Example ‑ Retrieve date information about a File

Set File="FILE.TXT"

Write $ZU(140,3,File) ; Create Date/Time

61371,68318

Write $ZDatetime($ZU(140,3,File))

01/10/2009 18:58:38

Write $ZU(140,2,File) ; Modified Date/Time

61371,68318

Write $ZDateTime($ZU(140,2,File))

01/10/2009 18:58:38

Example ‑ Check on the existence of a File

Set File="FILE.TXT"

Write $ZU(140,4,File) ; Existence of File

0 ; 0 – file exists

Set File="FILEXXX.TXT"

Write $ZU(140,4,File) ; Existence of File

-2 ; -2 – file does not exist

Example ‑ Copying a file

Set File="FILE.TXT"

Set NewFile="NEWFILE.TXT"

Write $ZU(140,11,File,NewFile) ; Copy FILE.TXT to NEWFILE.TXT

*0* ; copy successfull

Set File="NEWFILE.TXT"

Write $ZU(140,3,File) ; Create Date/Time of the new file

61372,35519

Example ‑ Renaming a file

Set File="FILE.TXT"

Set NewFile="NEWFILE.TXT"

Write $ZU(140,6,File,NewFile) ; Rename FILE.TXT to NEWFILE.TXT

Set File="FILE.TXT"

Write $ZU(140,4,File) ; Check the existence of File

-2 ; Old file does not exist,

; it has been renamed

Set File="NEWFILE.TXT"

Write $ZU(140,4,File) ; Check the existence of File

0 ; New file does exist

Example ‑ Delete a File

Set File="NEWFILE.TXT"

Write $ZU(140,5,File) ; Delete a file

0 ; 0 – delete successful

Set File="NEWFILE.TXT"

Write $ZU(140,5,File) ; Delete a file

-2 ; -2 – delete unsuccessful

; already deleted the file

# Error Processing

Example ‑ Testing for File Open

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set File="File.txt"  
 Close File ;good practice to close a file before opening it  
 Open File:"RS":5  
 If '$Test { ;test to see if the file is open  
 Write "File "\_File\_" not open"  
 ;do some sort of error handling  
 }

Quit

Example ‑ Testing for the existence of Parameters

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Do Procedure1("Par1","")

Quit

Procedure1(Param1,Param2)

If $G(Param1)="" {

Write "Parameter 1 is null."

;do some sort of error handling

}

If $G(Param2)="" {

Write "Parameter 2 is null."

;do some sort of error handling

}

Quit

Example ‑ Testing the Status Code that is returned from calling another routine

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Statuscode=^SomeRoutine

If $G(Statuscode)'=1 {

Write "Bad Statuscode returned from call to ^SomeRoutine"

;do some sort of error handling

}

Quit

Example ‑ Setting the Error Trap

Set $ZT="^%ETN" ; $ZT is the error trap special variable, it says when an

; error occurs, go to this routine, ^%ETN

Example ‑ Springing the Error Trap

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

Set $ZT="^%ETN" ; $ZT is the error trap special variable, it says when

; an error occurs, go to this routine, ^%ETN

Kill X

Write X ; variable X does not exist so trying to write variable X

; springs the error trap

Example ‑ Running RoutineA to Spring the Error Trap

Run ^RoutineA

Error has occurred: <UNDEFINED> at 10:30 AM

In

Example 13‑6 we run RoutineA and the result is as expected, and Undefined error.

Example ‑ Report upon the Error

Do ^%ERN

For Date: T 26 Oct 2008 3 Errors

Error: ?

Select one of the errors for this date. Enter ?L to list

all the errors which are defined for 26 Oct 2008.

Enter \* to enter a comment relating to all the errors

which exist for this date (e.g. 'all fixed')

Enter tag^routine to list this date's errors which

occurred in a specific routine.

Enter [text to list this date's errors which had 'text'

in either the error, line of code or comment.

Enter <error to list the errors with the specified error.

Error: ?L

1. <UNDEFINED>RoutineA+5^RoutineA \*X at 9:55 am. $I=|TRM|:|2524($X=0 $Y=136)

$J=2524 $ZA=0 $ZB=$c(13) $ZS=47630 ($S=48618256)

Write X ;variable X does not exist

Error: 1

1. <UNDEFINED>RoutineA+5^RoutineA \*X at 9:55 am. $I=|TRM|:|2524($X=0 $Y=136)

$J=2524 $ZA=0 $ZB=$c(13) $ZS=47630 ($S=48618256)

Write X ;variable X does not exist

Variable: ?

Enter the name of the variable you wish to view.

Enter the stack level you wish to view.

Enter ?# to view the variables defined for stack level #

Enter ?var to list levels where variable 'var' is defined

Enter \*S to view all the Process State Variables ($S,etc)

Enter \*F to view the execution Frame Stack

Enter \*C to enter a Comment for this error

Enter \*L to Load the variables into the current partition

Enter \*P to Print the Stack & Symbol information to a device

Enter \*A to to print ALL information, state variables, Stack

Frames, and Local Variables to a device.

Enter \*V to trace selected variables through the frame stack

Enter \*? to redisplay the error information

Variable:

Example ‑ Error Processing Routine for Other (less serious) Errors

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA  
 ;  
 Write !,"RoutineA starting"   
 ;   
 Label1  
 Write !,"RoutineA at Label1"  
 Set $ZT="ErrorProcessing" ; $ZT is the error trap special   
 ; variable, it says when an error   
 ; occurs, go to this label  
   
 Set ReturnLabel="RestartPoint1" ; This is the restart label  
 ; label after an error occurs  
 Kill X  
 Write X ; variable X does not exist so trying to write  
 ; variable X springs the error trap  
  
 RestartPoint1  
   Write !,"RoutineA at RestartPoint1"  
 Set $ZT="ErrorProcessing" ; $ZT is the error trap special   
 ; variable, it says when an error   
 ; occurs, go to this label  
   
 Set ReturnLabel="RestartPoint2" ; This is the restart label  
 ; label after an error occurs  
 Kill X  
 Write X ; variable X does not exist so trying to write  
 ; variable X springs the error trap  
  
RestartPoint2  
   Write !,"RoutineA at RestartPoint2"  
  
   Write !,"RoutineA exiting"  
   Quit  
  
ErrorProcessing(Location,Message)  
 Write !,"RoutineA at ErrorProcessing"  
 Write !,$ZE  
 Set DateTime=$ZDATETIME($H)  
 Set ^ApplError(DateTime)=$ZE  
 Write !,DateTime," ",$ZE  
 Goto @ReturnLabel ; branch back to the restart point.

Example ‑ Output from

Example 13‑8

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA starting

RoutineA at Label1

RoutineA at ErrorProcessing

<UNDEFINED>Label1+8^ROUTINEA \*X

06/09/2011 21:15:37 <UNDEFINED>Label1+8^ROUTINEA \*X

RoutineA at RestartPoint1

RoutineA at ErrorProcessing

<UNDEFINED>RestartPoint1+8^ROUTINEA \*X

06/09/2011 21:15:37 <UNDEFINED>RestartPoint1+8^ROUTINEA \*X

RoutineA at RestartPoint2

RoutineA exiting

Example ‑ Example of Syntax Errors

Set Set X=1 ;Two "Set" commands

<SYNTAX>

Set X=$O(^GLOBAL(X) ;Missing a second right parentheses

<SYNTAX>

Write $F(X) ;$Find needs more parameters

<SYNTAX>

Write $E(X,1,1,2) ;$Extract has too many parameters

<SYNTAX>

Write $P(X,,1) ;$Piece is missing the middle parameter

<SYNTAX>

QUIT: ;Quit is missing a post-conditional

<SYNTAX>

Example ‑ Examples of Undefined Errors

Kill X

Write X ;variable X is not defined

<UNDEFINED>

Kill Y

Write X=Y ;variable Y is not defined

<UNDEFINED>

Kill ^Global(Sub1)

Write ^Global(Sub1) ;Global node does not exist

<UNDEFINED>

Kill Sub1

Write ^Global(Sub1) ;it may be difficult to tell which is undefined,

;the variable used as a subscript, or the Global node

<UNDEFINED>

Example ‑ Undefined Errors and the New command

Set X=123 ;variable X is just defined

New X ;New command

Write X ;variable X is not defined because it was just Newed

<UNDEFINED>

Example ‑ Examples of a Subscript Error

Set Sub1=""

Set ^Global(Sub1)="" ;Sub1 is blank

<SUBSCRIPT>

Kill Sub1

Set ^Global(Sub1)="" ;If Sub1 did not exist,an <UNDEFINED> error is produced

<UNDEFINED>

Example ‑ Example of an EndOfFile Error

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

ReadFile ;

Set File="C:\FILE.DAT"

Open File:"R":10 ;open the file with a timeout of 10 sec

If '$T Write !,"Cannot open file" Q ;quit if the file cannot be open

Set RecordCount=0

Loop

Set RecordCount=RecordCount+1

Use File R Rec

Use 0 Write !,Rec," Record count: ",RecordCount

Go Loop

Example ‑ Example of running the routine ^ReadFile

Do ^ReadFile

First record in the file Record count: 1

Second record in the file Record count: 2

Third record in the file Record count: 3

Fourth record in the file Record count: 4

Fifth record in the file Record count: 5

Use File R Rec

^

<ENDOFFILE>Loop+2^ReadFile

Example ‑ Example of a Divide Error

Write 1/0

<DIVIDE>

Example ‑ Example of a NoLine Error

RoutineA ;

;

Start ;

Do Process^RoutineB ;The Label Process does not exist in RoutineB

<NOLINE>

Example ‑ Example of a NoRoutine Error

RoutineA ;

;

Start ;

Do ^RoutineD ;RoutineD does not exist

<NOROUTINE>

Example ‑ Example of a Call Stack

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA ;

Do ^RoutineB

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineB

Do ^RoutineC

Quit

;

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineC

; ;we shall consider the call stack at this point

Quit

Example ‑ Four Routine Call Stack

|  |
| --- |
| RoutineW |
| RoutineZ |
| RoutineY |
| RoutineZ |
| ^%STACK |

Example ‑ Call Stack Demonstration

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineW  
 ;  
 Write !,$T(+0)," Start"  
 Do ^RoutineX  
 Write !,$T(+0)," Finish"  
 Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineX  
 ;  
 Write !,$T(+0)," Start"  
 Do ^RoutineY  
 Write !,$T(+0)," Finish"  
 Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineY  
 ;  
 Write !,$T(+0)," Start"  
 Do ^RoutineZ  
 Write !,$T(+0)," Finish"  
 Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineZ  
 ;  
 Write !,$T(+0)," Start"  
 Do ^%STACK  
 Write !,$T(+0)," Finish"  
 Quit

Example ‑ Call Stack Demonstration – running RoutineW

Do ^RoutineW

RoutineW Start

RoutineX Start

RoutineY Start

RoutineZ Start 6

Process Stack:

Level Type Line Source

1 SIGN ON

2 DO ~D ^RoutineW

3 DO RoutineW+3^RoutineW ~D ^RoutineX

4 DO RoutineX+3^RoutineX ~D ^RoutineY

5 DO RoutineY+3^RoutineY ~D ^RoutineZ

6 DO RoutineZ+3^RoutineZ ~D ^%STACK

# Testing and Debugging

Example ‑ Break command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineDebug ;

Set X=1

Set Y=2

Set Z=3

Set A=Y\*Z

Break ;Break command

Write !,X

Write !,Y

Write !,Z

Write !,A

Example ‑ Executing the Break command

D ^RoutineDebug

Break ;Break command

^

<BREAK>RoutineDebug+5^RoutineDebug

USER 2d0>Write X ;at this point the programmer has control

1

USER 2d0>Write Y

2

USER 2d0>Write Z

3

USER 2d0>Write A

6

USER 2d0>G ;Go command given, the routine takes

;back control

1

2

5

6

Example ‑ Break command based on the DEBUG variable

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineDebug ;

Set X=1

Set Y=2

Set Z=3

Set A=Y\*Z

If $G(DEBUG)=1 Break ;Break command and DEBUG variable

Write !,X

Write !,Y

Write !,Z

Write !,A

Example ‑ Break command with Line Stepping

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineDebug ;

Set X=1

Set Y=2

Set Z=3

Set A=Y\*Z

Break "L" ;Break command with Line Stepping

Write !,X

Write !,Y

Write !,Z

Write !,A

Example ‑ Break command with Line Stepping Demonstrated

Do ^RoutineDebug

Write !,X

^

<BREAK>RoutineDebug+6^RoutineDebug

USER 2d0>G ;Go command

1

Write !,Y

^

<BREAK>RoutineDebug+7^RoutineDebug

USER 2d0>G ;Go command

2

Write !,Z

^

<BREAK>RoutineDebug+8^RoutineDebug

USER 2d0>G ;Go command

3

Write !,A

^

<BREAK>RoutineDebug+9^RoutineDebug

USER 2d0>G ;Go command

6

USER>

Example ‑ Disable the Break command

Do ^RoutineDebug

Write !,X

^

<BREAK>RoutineDebug+6^RoutineDebug

USER 2d0>G ;Go command

1

Write !,Y

^

<BREAK>RoutineDebug+7^RoutineDebug

USER 2d0>Break "OFF" ;break off command

USER 2d0>G ;Go command

2

3

6

Example ‑ ZBreak command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineDebug ;

Start ;

Set X=1

Set Y=2

Set Z=3

Set A=Y\*Z

Write !,X

Write !,Y

Write !,Z

Write !,A

Quit

Example ‑ Invoking the ZBreak command

ZBreak Start+1^RoutineDebug ;break at Start+1

Do ^RoutineDebug

Set X=1

^

<BREAK>Start+1^RoutineDebug

USER 2d0>G ;Go command

1

2

3

6

Example ‑ ZBreak command with Line Stepping

ZBreak Start+1^RoutineDebug:"L"

Do ^RoutineDebug

Set X=1

^

<BREAK>Start+1^RoutineDebug

USER 2d0>G

Set Y=2

^

<BREAK>Start+2^RoutineDebug

USER 2d0>G

Set Z=3

^

<BREAK>Start+3^RoutineDebug

USER 2d0>G

Set A=Y\*Z

^

<BREAK>Start+4^RoutineDebug

USER 2d0>G

Write !,X

^

<BREAK>Start+5^RoutineDebug

USER 2d0>G

1

Write !,Y

^

<BREAK>Start+6^RoutineDebug

USER 2d0>G

2

Write !,Z

^

<BREAK>Start+7^RoutineDebug

USER 2d0>G

3

Write !,A

^

<BREAK>Start+8^RoutineDebug

USER 2d0>G

6

USER>

Example ‑ Setting Watchpoints

USER>ZBreak \*Y ;set a watchpoint whenever the Y variable is modified

USER>Do ^RoutineDebug

Set Y=2

^

<BREAK>Start+2^RoutineDebug

USER 2d0>G

1

2

3

6

USER>

Example ‑ ZBreak On-line help

ZBreak ?

ZB location{:parms} Set breakpoint

ZB -location{#delay} Disable breakpoint

ZB +location Enable breakpoint

ZB --location Remove breakpoint

location is a line reference, or \*variable, or $

parms is action:{condition}:{execute}

action is B, L, L+, S, S+, T, or N which mean

BREAK, Line step, Single step, Trace, or No action

condition is a truth-valued expression

execute is code to be executed

/CLEAR Remove all breakpoints

/DEBUG{:device} Clear or set debug device

/TRACE:{ON,OFF,ALL}{:device} Enable or disable trace, or trace all lines

/ERRORTRAP:{ON,OFF} Enable or disable $ZTRAP and $ETRAP

/INTERRUPT:{NORMAL,BREAK} Specify Control-C action

# Procedures

Example ‑ Template of a COS Procedure

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

Write !,"Param1: ",Param1

Write !,"Param2: ",Param2

Write !,"PubVar1: ",PubVar1

Write !,"PubVar2: ",PubVar2

}

Example ‑ Calling a COS Procedure

Set Param1=2

Set Param2=3

Set PubVar1=5 ;public variable

Set PubVar2=5 ;public variable

Do ProcedureABC^RoutineA(Param1,Param2) ;calling a Procedure

Param1: 2

Param2: 3

PubVar1: 5

PubVar2: 5

Example ‑ Template of a Public Procedure

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

Example ‑ Template of a Private Procedure

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Private{

Example ‑ Parameters passed by a calling process

Do ProcedureABC^RoutineA(Param1,Param2) ;calling a Procedure

Example ‑ Parameters received by a called Procedure

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

Example ‑ Calling a Procedure with Parameters Passed by Value

Set Param1=2

Set Param2=3

Set PubVar1=5

Set PubVar2=5

Do ProcedureABC^RoutineA(Param1,Param2)

;

Write !,"Param1: ",Param1

Write !,"Param2: ",Param2

(RoutineA) Param1: NewValue1

(RoutineA) Param2: NewValue2

Param1: 2 ;original value displayed

Param2: 3 ;original value displayed

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Set Param1="NewValue1" ;set Param1 to a new value

Set Param2="NewValue2" ;set Param2 to a new value

Write !,"(RoutineA) Param1: ",Param1

Write !,"(RoutineA) Param2: ",Param2

}

Example ‑ Calling a Procedure with Parameters Passed by Reference

Set Param1=2

Set Param2=3

Set PubVar1=5

Set PubVar2=5

Do ProcedureABC^RoutineA(.Param1,.Param2) ;Period before Params, which

; ;indicates passed by reference

Write !,"Param1: ",Param1

Write !,"Param2: ",Param2

(RoutineA) Param1: NewValue1

(RoutineA) Param2: NewValue2

Param1: NewValue1 ;modified value displayed

Param2: NewValue2 ;modified value displayed

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Set Param1="NewValue1" ;set Param1 to a new value

Set Param2="NewValue2" ;set Param2 to a new value

Write !,"(RoutineA) Param1: ",Param1

Write !,"(RoutineA) Param2: ",Param2

}

Example ‑ Calling a Procedure using Default Parameters

Set Param2=3

Set PubVar1=5

Set PubVar2=5

Do ProcedureABC^RoutineA(,Param2) ;Comma indicates no first parameter

(RoutineA) Param1: Default1

(RoutineA) Param2: 3

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1="Default1",Param2="Default2") [PubVar1,PubVar2] Public {

;

Write !,"(RoutineA) Param1: ",Param1

Write !,"(RoutineA) Param2: ",Param2

}

Example ‑ Calling a Procedure with the Actual Parameter List less than the Formal Parameter List

Set Param1=2

Set Param2=3

Set PubVar1=5

Set PubVar2=5

Do ProcedureABC^RoutineA(Param1) ;Actual Parameter List

(RoutineA) Param1: 2

(RoutineA) Param2:

Write !,"(RoutineA) Param2: ",Param2

^

<UNDEFINED>ProcedureABC+3^RoutineA \*Param2

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

; (formal parameter list)

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Write !,"(RoutineA) Param1: ",Param1

Write !,"(RoutineA) Param2: ",Param2

}

Example ‑ Calling a Procedure with the Actual Parameter List less than the Formal Parameter List

Set Param1=2

Set Param2=3

Set PubVar1=5

Set PubVar2=5

Do ProcedureABC^RoutineA(Param1)

(RoutineA) Param1: 2

(RoutineA) Param2:

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

Set Param1=$G(Param1)

Set Param2=$G(Param2)

;

Write !,"(RoutineA) Param1: ",Param1

Write !,"(RoutineA) Param2: ",Param2

}

Example ‑ Template of a COS Procedure with Public Variables

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

Write !,"Param1: ",Param1

Write !,"Param2: ",Param2

Write !,"PubVar1: ",PubVar1

Write !,"PubVar2: ",PubVar2

}

Example ‑ Calling a Procedure with Public Variables

Set PubVar1=2

Set PubVar2=3

Do ProcedureABC^RoutineA()

(RoutineA) PubVar1: NewValue1

(RoutineA) PubVar2: NewValue2

Write !,"PubVar1: ",PubVar1

Write !,"PubVar2: ",PubVar2

PubVar1: NewValue1 ;new value displayed

PubVar2: NewValue2 ;new value displayed

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC() [PubVar1,PubVar2] Public {

;

Set PubVar1="NewValue1" ;set PubVar1 to a new value

Set PubVar2="NewValue2" ;set PubVar2 to a new value

Write !,"(RoutineA) PubVar1: ",PubVar1

Write !,"(RoutineA) PubVar2: ",PubVar2

}

Example ‑ Variables created and used outside the called Procedure

Set Param1=1

Set Param2=2

Set PubVar1=1

Set PubVar2=2

Set OutVar1=2

Set OutVar2=3

Do ProcedureABC^RoutineA(Param1,Param2)

<UNDEFINED>ProcedureABC+2^RoutineA \*OutVar1

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Write OutVar1

Write OutVar2

}

Example ‑ Variables created inside the called Procedure

Set Param1=1

Set Param2=2

Set PubVar1=1

Set PubVar2=2

Do ProcedureABC^RoutineA(Param1,Param2)

Write !,"PrivateVar1: ",PrivateVar1

PrivateVar1:

Write !,"PrivateVar1: ",PrivateVar1

^

<UNDEFINED> \*PrivateVar1

Write !,"PrivateVar2: ",PrivateVar2

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Set PrivateVar1=1

Set PrivateVar2=2

Write !,"PrivateVar1: ",PrivateVar1

Write !,"PrivateVar2: ",PrivateVar2

}

Example ‑ Variables created inside the called Procedure are not passed to other Modules

Set Param1=1

Set Param2=2

Set PubVar1=1

Set PubVar2=2

Do ProcedureABC^RoutineA(Param1,Param2)

(RoutineA) PrivateVar1: 1

(RoutineA) PrivateVar2: 2

(RoutineB) PrivateVar1:   
<UNDEFINED>ProcedureDEF+2^RoutineB \*PrivateVar1 ^

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Set PrivateVar1=1

Set PrivateVar2=2

Write !,"(RoutineA) PrivateVar1: ",PrivateVar1

Write !,"(RoutineA) PrivateVar2: ",PrivateVar2

;

Do ProcedureDEF^RoutineB()

}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineB

;

ProcedureDEF() Public {

;

Write !,"(RoutineB) PrivateVar1: ",PrivateVar1

Write !,"(RoutineB) PrivateVar2: ",PrivateVar2

;

}

Example ‑ Variables created inside the called Procedure passed to other Procedures through Parameters.

Set Param1=1

Set Param2=2

Set PubVar1=1

Set PubVar2=2

Do ProcedureABC^RoutineA(Param1,Param2)

(RoutineA) PrivateVar1: 1

(RoutineA) PrivateVar2: 2

(RoutineB) PriVar1: 1

(RoutineB) PriVar2: 2

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Set PrivateVar1=1

Set PrivateVar2=2

Write !,"(RoutineA) PrivateVar1: ",PrivateVar1

Write !,"(RoutineA) PrivateVar2: ",PrivateVar2

;

Do ProcedureDEF^RoutineB(PrivateVar1,PrivateVar2)

}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineB

;

ProcedureDEF(PriVar1,PriVar2) Public {

;

Write !,"(RoutineB) PriVar1: ",PriVar1

Write !,"(RoutineB) PriVar2: ",PriVar2

;

}

Example ‑ Variables created inside the called Procedure passed to other Procedures through the Public List.

Set Param1=1

Set Param2=2

Set PubVar1=1

Set PubVar2=2

Do ProcedureABC^RoutineA(Param1,Param2)

(RoutineA) PrivateVar1: 1  
 (RoutineA) PrivateVar2: 2  
 (RoutineB) PrivateVar1: 1  
 (RoutineB) PrivateVar2: 2

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2, PrivateVar1,PrivateVar2] Public {

;

Set PrivateVar1=1

Set PrivateVar2=2

Write !,"(RoutineA) PrivateVar1: ",PrivateVar1

Write !,"(RoutineA) PrivateVar2: ",PrivateVar2

;

Do ProcedureDEF^RoutineB(Param1,Param2)

}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineB

;

ProcedureDEF(Param1,Param2) [PrivateVar1,PrivateVar2] Public {

;

Write !,"(RoutineB) PrivateVar1: ",PrivateVar1

Write !,"(RoutineB) PrivateVar2: ",PrivateVar2

;

}

Example ‑ Calling a Private Procedure

Set Param1=1

Set Param2=2

Set PubVar1="1"

Set PubVar2="2"

Do ProcedureABC^RoutineA(Param1,Param2)

Set Param2=2

Set PubVar1="1"

Set PubVar2="2"

Do ProcedureABC^RoutineA(Param1,Param2) ;Calling a Private Procedure

Do ProcedureABC^RoutineA(Param1,Param2)

^

<NOLINE>

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Private {

;

Set PrivateVar1=1

Set PrivateVar2=2

Write !,"(RoutineA) PrivateVar1: ",PrivateVar1

Write !,"(RoutineA) PrivateVar2: ",PrivateVar2

;

}

Example ‑ Calling a Procedure like a Function

Set PubVar1=3

Set PubVar2=4

Write $$ProcedureABC^RoutineA(1,2) ;Called as a Function

(RoutineA) Param1: 1

(RoutineA) Param2: 2

(RoutineA) PubVar1: 3

(RoutineA) PubVar2: 4

1

Do ProcedureABC^RoutineA(1,2) Called as a Subroutine

(RoutineA) Param1: 1

(RoutineA) Param2: 2

(RoutineA) PubVar1: 3

(RoutineA) PubVar2: 4

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

RoutineA

;

ProcedureABC(Param1,Param2) [PubVar1,PubVar2] Public {

;

Write !,"(RoutineA) Param1: ",Param1

Write !,"(RoutineA) Param2: ",Param2

Write !,"(RoutineA) PubVar1: ",PubVar1

Write !,"(RoutineA) PubVar2: ",PubVar2,!

Quit 1

}

## Exercise on a COS Procedure

Write a new COS *Procedure* from scratch. First create a new routine called MyRoutine. Inside MyRoutine include the code to call the *Procedure*. Name your COS *Procedure* “MyProcedure”. Pass MyProcedure four parameters of your own naming. Two of the Parameters should be *Called by Value* and two *Called by Reference*. Change the value for all four parameters inside your procedure. Include two variables in the *Public Variable List*. Display the value of the Parameters as well as the variables in the *Public Variable List* before calling MyProcedure, inside MyProcedure and after calling MyProcedure. Compile and run MyRoutine and ensure you get correct output.

Your assignment should look something like the following:

Example ‑ Exercise on COS Procedure

MyRoutine ;

;

Set PubVar1="PubVar1"

Set PubVar2="PubVar2"

Set Param1="Param1"

Set Param2="Param2"

Set Param3="Param3"

Set Param4="Param4"

Write !,"Before calling MyProcedure - Param1: ",Param1

Write !,"Before calling MyProcedure - Param2: ",Param2

Write !,"Before calling MyProcedure - Param3: ",Param3

Write !,"Before calling MyProcedure - Param4: ",Param4

Write !,"Before calling MyProcedure - PubVar1: ",PubVar1

Write !,"Before calling MyProcedure - PubVar2: ",PubVar2

Do MyProcedure(Param1,Param2,.Param3,.Param4)

Write !,"After calling MyProcedure - Param1: ",Param1

Write !,"After calling MyProcedure - Param2: ",Param2

Write !,"After calling MyProcedure - Param3: ",Param3

Write !,"After calling MyProcedure - Param4: ",Param4

Write !,"After calling MyProcedure - PubVar1: ",PubVar1

Write !,"After calling MyProcedure - PubVar2: ",PubVar2

Quit

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

MyProcedure(Param1,Param2,Param3,Param4) [PubVar1,PubVar2] Public {

Set PubVar1="PubVar100"

Set PubVar2="PubVar200"

Set Param1="Param100"

Set Param2="Param200"

Set Param3="Param300"

Set Param4="Param400"

Write !,"Inside MyProcedure - Param1: ",Param1

Write !,"Inside MyProcedure - Param2: ",Param2

Write !,"Inside MyProcedure - Param3: ",Param3

Write !,"Inside MyProcedure - Param4: ",Param4

Write !,"Inside MyProcedure - PubVar1: ",PubVar1

Write !,"Inside MyProcedure - PubVar2: ",PubVar2

}

Do ^MyRoutine

Before calling MyProcedure - Param1: Param1

Before calling MyProcedure - Param2: Param2

Before calling MyProcedure - Param3: Param3

Before calling MyProcedure - Param4: Param4

Before calling MyProcedure - PubVar1: PubVar1

Before calling MyProcedure - PubVar2: PubVar2

Inside MyProcedure - Param1: Param100

Inside MyProcedure – Param2: Param200

Inside MyProcedure - Param3: Param300

Inside MyProcedure - Param4: Param400

Inside MyProcedure - PubVar1: PubVar100

Inside MyProcedure - PubVar2: PubVar200

After calling MyProcedure - Param1: Param1

After calling MyProcedure - Param2: Param2

After calling MyProcedure - Param3: Param300

After calling MyProcedure - Param4: Param400

After calling MyProcedure - PubVar1: PubVar100

After calling MyProcedure - PubVar2: PubVar200

# Structured Code

Example ‑ If command with Structured Code

Set X=12

If (X=12) {Set X=13}

Write X

13

Example ‑ Loop commands with Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For I=1:1:3 {

Write !,"First Level: ",I

For II=1:1:3 {

Write !," Second Level: ",I+II

}

}

Write !,"End of For Loop"

Example ‑ Output from

Example 16‑2

First Level: 1

Second Level: 2

Second Level: 3

Second Level: 4

First Level: 2

Second Level: 3

Second Level: 4

Second Level: 5

First Level: 3

Second Level: 4

Second Level: 5

Second Level: 6

End of For Loop

Example ‑ Setting up the (Transportation Machines) Global

Set ^TM("Cars")="Data"

Set ^TM("Cars","Domestic")=""

Set ^TM("Cars","Domestic","Dodge")=""

Set ^TM("Cars","Domestic","Dodge","Caravan")=""

Set ^TM("Cars","Domestic","Dodge","150 Truck")=""

Set ^TM("Cars","Foreign")=""

Set ^TM("Cars","Foreign","Toyota")="Data"

Set ^TM("Cars","Foreign","Toyota","Tercel")=""

Set ^TM("Cars","Foreign","BMW")=""

Set ^TM("Airplanes")=""

Set ^TM("Airplanes","Military")=""

Set ^TM("Airplanes","Military","Jets")=""

Set ^TM("Airplanes","Military","Jets","F-14")="Data"

Set ^TM("Airplanes","Military","Jets","F-16")=""

Set ^TM("Airplanes","Military","Prop planes")=""

Set ^TM("Airplanes","Military","Prop planes","P-38")=""

Set ^TM("Airplanes","Commercial")="Data"

Set ^TM("Airplanes","Commercial","Jets")=""

Set ^TM("Airplanes","Commercial","Jets","707")=""

Set ^TM("Airplanes","Commercial","Jets","747")="Data"

In

Example 16‑4, we setup the TM Global. The next few examples use this Global.

Example ‑ Traversing a Global Array with the Do While command and Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1="" Do {

Set S1=$O(^TM(S1)) Q:S1="" ;get the next S1 subscript

Write !,"S1: ",S1

Set S2="" Do {

Set S2=$O(^TM(S1,S2)) Q:S2="" ;get the next S2 subscript

Write !," S2: ",S2

Set S3="" Do {

Set S3=$O(^TM(S1,S2,S3)) Q:S3="" ;get the next S3 subscript

Write !," S3: ",S3

} While S3'=""

} While S2'=""

} While S1'=""

Example ‑ Output from

Example 16‑5

S1: Airplanes

S2: Commercial

S3: Jets

S2: Military

S3: Jets

S3: Prop planes

S1: Cars

S2: Domestic

S3: Dodge

S2: Foreign

S3: BMW

S3: Toyota

Example ‑ Traversing a Global Array with the While command and Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set S1=$O(^TM("")) ; set S1 control parameter  
 While (S1'="") { ; S1'="" is the first control parameter  
 Write !,"S1: ",S1  
 Set S2=$O(^TM(S1,"")) ; set S2 control parameter  
 While (S2'="") { ; S2'="" is the second control parameter  
 Write !," S2: ",S2  
 Set S3=$O(^TM(S1,S2,"")) ; set S3 control parameter  
 While (S3'="") { ; S3'="" - third control parameter  
 Write !," S3: ",S3  
 Set S3=$O(^TM(S1,S2,S3)) ;get next S3 entry  
 }  
 Set S2=$O(^TM(S1,S2)) ;get next S2 entry   
 }  
 Set S1=$O(^TM(S1)) ;get next S1 entry  
 }

Example ‑ Output from

Example 16‑7

S1: Airplanes

S2: Commercial

S3: Jets

S2: Military

S3: Jets

S3: Prop planes

S1: Cars

S2: Domestic

S3: Dodge

S2: Foreign

S3: BMW

S3: Toyota

Example ‑ If and Else commands with Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

; First Method

Set X=1

If X=1 {

Write !,"X=1"

}

Else {

Write !,"X is not = 1"

}

; Second Method

Set X=1

If X=1 {Write !,"X=1"}

Else {Write !,"X is not = 1"}

; Third Method

Set X=1

If X=1 {Write !,"X=1"} Else {Write !,"X is not = 1"}

Example ‑ Nested If and Else commands with Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

 ; First Method  
  Set X=1,Y=2  
  If X=1 {  
  Write !,"X=1"  
  If Y=2 {  
  Write !,"Y=2"  
  }  
  Else { ;this Else matches If Y=2  
  Write !,"Y is not = 2"  
 }  
 }  
 Else { ;this Else matches If X=1  
  Write !,"X is not = 1"  
  If Y=3 {  
     Write !,"Y=3"  
  }  
  Else { ;this Else matches If Y=3  
     Write !,"Y is not = 3"  
   }  
 }

; Second Method

 Set X=1,Y=2  
 If X=1 {Write !,"X=1"  
   If Y=2 {Write !,"Y=2"}  
   Else {Write !,"Y is not = 2"}  
 }  
 Else {Write !,"X is not = 1"  
   If Y=3 {Write !,"Y=3"}  
   Else {Write !,"Y is not = 3"}  
 }

; Third Method

Set X=1,Y=2

If X=1 {Write !,"X=1" If Y=2 {Write !,"Y=2"} Else {Write !,"Y is not = 2"}}

Else {Write !,"X is not = 1" If Y=3 {Write !,"Y=3"} Else {Write !,"Y is not = 3"}}

Example ‑ If, ElseIf and Else commands with Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

; First Method

Set X=1

If X=1 {

Write !,"X=1"

}

ElseIf X=2 {

Write !,"X=2"

}

Else {

Write !,"X not = 1 or 2"

}

; Second Method

Set X=1

If X=1 {Write !,"X=1"}

ElseIf X=2 {Write !,"X=2"}

Else {Write !,"X not = 1 or 2"}

Example ‑ If, (multiple) ElseIf and Else command with Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set Day=1

If Day=1 {Write !,"Today is Sunday"}

ElseIf Day=2 {Write !,"Today is Monday"}

ElseIf Day=3 {Write !,"Today is Tuesday"}

ElseIf Day=4 {Write !,"Today is Wednesday"}

ElseIf Day=5 {Write !,"Today is Tursday"}

ElseIf Day=6 {Write !,"Today is Friday"}

ElseIf Day=7 {Write !,"Today is Saturday"}

Else {Write !,"I don't know what day it is!"}

Example ‑ Simple Inline Do command with For Loop commands

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For Num=1:1:3 Do

. If Num=2 Quit

. Write !,Num

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

1 ;first 1 is written

3 ;then 3 is written, 2 is skipped

Example ‑ For Loop command in Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For Num=1:1:3 {

If Num=2 Quit

Write !,Num

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

1 ;first 1 is written

;2 and 3 are not written

Example ‑ Structure Code rewritten

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For Num=1:1:3 {

If Num'=2 {

Write !,Num

}

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

1 ;first 1 is written

3 ;3 is written but 2 is not

Example ‑ Structured Code using the Continue command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For Num=1:1:3 {

If Num=2 Continue

Write !,Num

}

Example ‑ Simple Inline Do command with For Loop command

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set X=1

If X=1 Do

. Write !,1

. Write !,2

. Quit

Write !,3

Write !,4

Quit

Example ‑ Code from

Example 16‑17 using Structured Code

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set X=1

If X=1 {

Write !,1

Write !,2

Quit

}

Write !,3

Write !,4

Quit

“*No one has a finer command of language than the person who keeps his mouth shut." —Sam Rayburn*

# Writing Robust Code

Example ‑ Customer Procedure version 1

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Customer ;

Write !!,"1 – Customer Registration"

Write !,"2 – Customer Name Change"

Write !,"3 – Customer Complaints"

Read !,"Enter 1,2 or 3 : ",Option

If Option=1 {Do CustReg}

Elseif Option=2 {Do CustName}

Elseif Option=3 {Do CustComplaint}

Quit

Example ‑ Customer Procedure version 2

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Customer ;

Write !!,"1 – Customer Registration"

Write !,"2 – Customer Name Change"

Write !,"3 – Customer Complaints"

Read !,"Enter 1,2 or 3 : ",Option:60 ;give user 60 second time out

If Option=1 {Do CustReg}

Elseif Option=2 {Do CustName}

Elseif Option=3 {Do CustComplaint}

Else {Write !,"Invalid input" Goto Customer} ;check for invalid input

Quit

Example ‑ Customer Procedure version 3

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Customer ;

Write !!,"1 – Customer Registration"

Write !,"2 – Customer Name Change"

Write !,"3 – Customer Complaints"

Write !,"""Exit"" to quit"

Read !,"Enter 1,2 or 3 : ",Option:60 ;give user 60 second time out

If Option=1 {Do CustReg}

Elseif Option=2 {Do CustName}

Elseif Option=3 {Do CustComplaint}

Elseif Option="Exit" {Quit}

Else {Write !,"Invalid input" Goto Customer}

Quit

Example ‑ Customer Procedure version 4

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Customer ;

Write !!,"1 – Customer Registration"

Write !,"2 – Customer Name Change"

Write !,"3 – Customer Complaints"

Write !,"""Exit"" to quit"

Read !,"Enter 1,2 or 3 : ",Option:60 ;give user 60 second time out

Set Option=$ZCVT(Option,"U") ;convert to uppercase

If Option=1 {Do CustReg}

Elseif Option=2 {Do CustName}

Elseif Option=3 {Do CustComplaint}

Elseif Option="EXIT" {Quit} ;check against uppercase Exit

Else {Write !,"Invalid input" Goto Customer}

Quit

Example ‑ Assuming a Variable is not in use

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

For I=1:1:10 Do Proc1

Quit

Proc1 ;

Write !,"Processing I of: ",I

Do Proc2

Quit

Proc2 ;

For I=1:1:5 {

Write !,"I number: ",I

}

Quit

Example ‑ Solution to assuming a variable is not in use

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

New I ;New I

For I=1:1:10 Do Proc1

Quit

Proc1 ;

Write !,"Processing I of: ",I

Do Proc2

Quit

Proc2 ;

New I ;New I

For I=1:1:5 {

Write !,"I number: ",I

}

Quit

Example ‑ Wrong Assumption concerning the Tab Character

Write ?10,"TAB",?10,"Character",?10,"Example"

Example ‑ Wrong Assumption concerning the Tab Character

Write ?10,"TAB",?10,"Character",?10,"Example"

TABCharacterExample

Example ‑ Wrong Assumption concerning the Tab Character, Solution

Write ?10,"TAB",?20,"Character",?40,"Example"

TAB Character Example

Example ‑ Assumption concerning the User’s Response Version I

Write !,"This is an important message to the user!"

Hang 10

Example ‑ Assumption concerning the User’s Response Version II

Write !,"This is an important message to the user!"

Read !,"Hit <Enter> to to acknowledge this message",X

Example ‑ Assumption concerning the User’s Response Version III

Write !,"This is an important message to the user!"

Read !,"Hit <Enter> to to acknowledge this message",X:120

Example ‑ Assumption concerning the User’s Response Version IV

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Set Tries=0  
Tryagain  
 Set Tries=Tries+1  
 Write !,"This is an important message to the user!"  
 Read !,"Hit <Enter> to to acknowledge this message",X:120  
 If '$Test {  
 If Tries>3 Write "at this point do some sort of error processing"  
 If Tries'>3 Goto Tryagain  
 }

Example ‑ Assumption concerning the Open Command, Solution

Set INFILE="INFILE.TXT"

Open INFILE:"R":10

If '$Test Write !,"Cannot open file: ",INFILE Quit

Example ‑ Assumptions concerning Parameter Passing

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal. RoutineA and RoutineB need to be in separate routines.

RoutineA ;

Set PARM1=1

Set PARM2=2

Do PROC^RoutineB(PARM1,PARM2)

Quit

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

RoutineB ;

PROC(PARM1,PARM2,PARM3)

Write !,PARM1

Write !,PARM2

Write !,PARM3

Quit

Example ‑ Undefined error from wrong number of Parameters

Do ^RoutineA

1

2

Write !,PARM3

^

<UNDEFINED>PROC+3^RoutineB \*PARM3

USER 3d1>

# Miscellaneous Topics

Example ‑ Setting Temporary Globals

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

START ;

Set VARA=1,VARB=2,VARC=3

Do STEP1

Do STEP2

Do STEP3

Quit

STEP1 ;

Set ^TMP($J,$H,"STEP1")="" ;global to mark this step

; . . . normal processing code . . .

; . . . normal processing code . . .

; . . . normal processing code . . .

Set ^TMP($J,$H,"STEP1","VARA")=VARA ;global to reveal the value of VARA

Set ^TMP($J,$H,"STEP1","VARB")=VARB ;global to reveal the value of VARB

Set ^TMP($J,$H,"STEP1","VARC")=VARC ;global to reveal the value of VARC

Quit

STEP2 ;

Set ^TMP($J,$H,"STEP2")="" ;global to mark this step

; . . . normal processing code . . .

; . . . normal processing code . . .

; . . . normal processing code . . .

Set ^TMP($J,$H,"STEP2","VARA")=VARA ;global to reveal the value of VARA

Set ^TMP($J,$H,"STEP2","VARB")=VARB ;global to reveal the value of VARB

Set ^TMP($J,$H,"STEP2","VARC")=VARC ;global to reveal the value of VARC

Quit

STEP3 ;

Set ^TMP($J,$H,"STEP3")="" ;global to mark this step

; . . . normal processing code . . .

; . . . normal processing code . . .

; . . . normal processing code . . .

Set ^TMP($J,$H,"STEP3","VARA")=VARA ;global to reveal the value of VARA

Set ^TMP($J,$H,"STEP3","VARB")=VARB ;global to reveal the value of VARB

Set ^TMP($J,$H,"STEP3","VARC")=VARC ;global to reveal the value of VARC

Quit

Example ‑ $Quit

If $Q=1 Quit 1 ;If $Q is 1, then we are inside a function and must

;return a value

If $Q=0 Quit ;If $Q is 0, then we are inside a subroutine and

;nothing is returned

Example ‑ $Quit Demonstrated

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Do Module() ;called as a SubRoutine, nothing is passed back

Write $$Module() ;called as a Function, 1 is passed back

1

Quit

Module()

;

; do processing

;

If $Q=1 Quit 1 ;If $Q is 1, then we are inside a function and must

;return a value

If $Q=0 Quit ;If $Q is 0, then we are inside a Subroutine and

;nothing is returned

Example ‑ Post-Conditional If Command

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Start ;

Set X=2

If X=1 Set Y="ABC" Do NextProc1 ;NextProc1 will not be executed

;if X is not 1

Set X=2

Set:X=1 Y="ABC" Do NextProc2 ;NextProc2 will always be executed

;regardless of the value of X

Quit

NextProc1 ;

Write !,"NextProc1"

Quit

NextProc2 ;

Write !,"NextProc2"

Quit

Example ‑ Other examples of Post Conditional commands

Write:X=1 "ABC" ;Write "ABC" if X=1

Do:X=1 Proc ;Do Proc if X=1

Read:X=1 Var ;Read Variable if X=1

Kill:X=1 Var ;Kill Variable if X=1

Quit:X=1 ;Quit if X=1

Example ‑ Indirection

Set TITLE="COS"

Set X="TITLE"

Write @X ;Write the variable that X contains, which is TITLE

COS

Write X

TITLE

Example ‑ Indirection Part 2

For X="CNT1","CNT2","CNT3" Set @X=0

Write

If you run the above from the Terminal, you should get the following output.

CNT1=0

CNT2=0

CNT3=0

X="CNT3"

Example ‑ Xecute command

Set TITLE="COS"

Set X="Write TITLE"

Xecute X

COS

Example ‑ $Data

Set X="" Write $D(X)

1

Write $D(^A(X))

<SUBSCRIPT>

Example ‑ $Data

Set ITEM="ABC"

Write $D(ITEM)

1

Example ‑ $Data revisited

Kill X

Set ITEM="ABC"

Write $D(ITEM,X) ;value of ITEM put into X

1

Write !,X

ABC

Example ‑ $Data revisited2

Kill ITEM

Write $D(ITEM,X)

0

Write !,X

<UNDEFINED>

Example ‑ Job Command to run a Routine: ^RTN

Job ^RTN

Example ‑ Job Command to run a Label: PROC^RTN

Job PROC^RTN

Example ‑ Job PROC^RTN(PARAM1,PARAM2)

Job PROC^RTN(PARAM1,PARAM2)

"There is no reason anyone would want a computer in their home."

-Ken Olson, president, chairman and founder of Digital Equipment Corporation, in 1977

# Miscellaneous Examples

Example ‑ Replace multiple spaces with a single space

Set X="ABC DEF"

For Quit:X'[" " Set X=$E(X,1,$F(X," ")-2)\_$E(X,$F(X," "),$L(X))

Write !,X

If you run the above code from the Terminal, you should get the following output.

ABC DEF

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set X="ABC DEF XYZ"

For Quit:X'[" " Set X=$E(X,1,$F(X," ")-2)\_$E(X,$F(X," "),$L(X))

Write !,X

If you run the above code from the Terminal, you should get the following output.

ABC DEF XYZ

Example ‑ Fill a line with words

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Set A="When in the Course of human events, it becomes necessary for one "  
 Set A=A\_"people to dissolve the political bands which have connected them "  
 Set A=A\_"with another, and to assume among the powers of the earth, the "  
 Set A=A\_"separate and equal station to which the Laws of Nature and of "  
 Set A=A\_"Nature's God entitle them, a decent respect to the opinions of "  
 Set A=A\_"mankind requires that they should declare the causes which impel "  
 Set A=A\_"them to the separation."  
 Set X=""  
  
 For I=1:1:$L(A," ") {  
   If $L(X)+$L($P(A," ",I))+1<81 {  
    If X="" {  
     Set X=X\_$P(A," ",I)  
    }  
    Else {  
     Set X=X\_" "\_$P(A," ",I)  
    }  
  }  
  Else {

Write !,X  
    Set X=$P(A," ",I)  
  }

 }

 If X'="" Write !,X

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

When in the Course of human events, it becomes necessary for one people to

dissolve the political bands which have connected them with another, and to

assume among the powers of the earth, the separate and equal station to which

the Laws of Nature and of Nature's God entitle them, a decent respect to the

opinions of mankind requires that they should declare the causes which impel

them to the separation.

Example ‑ Count Words or characters on a line or in a string variable

Set A="The slow lazy dog could not jump over his bed."

Write $L(A) ; number of characters in variable A

54

Write $L(A," ") ; number of words in variable A

11

Example ‑ Scan and replace text in a string variable

Set A="The slow lazy dog could not jump over his bed."

Set FROM="slow lazy"

Set TO="very fast"

Set $E(A,$F(A,FROM)-$L(FROM),$F(A,FROM))=TO\_" "

Write A

If you run the above code from the Terminal, you should get the following output.

The very fast dog could not jump over his bed.

Example ‑ Scan and replace text in a string variable using $Replace

Set A="The slow lazy dog could not jump over his bed."

Set FROM="slow lazy"

Set TO="very fast"

Set A=$Replace(A,FROM,TO)

Write A

If you run the above code from the Terminal, you should get the following output.

The very fast dog could not jump over his bed.

Example ‑ Find a value in a List multiple times.

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Set NUM=$LB("ONE","TWO","ONE","TWO","ONE")

Set X=0 Do {

Set X=$LF(NUM,"ONE",X) Q:X=0

Write !,X," ",$LI(NUM,X)

} While X’=""

If you run the above code from the Terminal, you should get the following output.

1 ONE

3 ONE

5 ONE

Example ‑ Create a line of characters

Write ! For I=1:1:30 Write "=" ;method one

==============================

Set $P(LINE,"=",31)="" ;method two

Write !,LINE

==============================

Example ‑ Find a variable string with spaces

Set X=" "

If X?1" " Write "Hit" ;pattern matching for one space

Hit

Set X=" "

If X?1.5" " Write "Hit" ;pattern matching for one to five space

Hit

Set X=" "

If X?." " Write "Hit" ;pattern matching for any number of spaces

Hit

Example ‑ Find the last piece in a string of pieces

Set X="FIRST^SECOND^THIRD^FOURTH"

Set LastPiece=$P(X,"^",$L(X,"^"))

Write !,LastPiece

FOURTH

Example ‑ Find the last piece in a list (created by $ListBuild

Set X=$LB("FIRST","SECOND","THIRD","FOURTH")

Set LastPiece=$LI(X,$LL(X))

Write !,LastPiece

FOURTH

Example ‑ Traversing a Global using Indirection and $Query

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Set ^Global="start"

Set ^Global(1)="Sub1=1"

Set ^Global(1,2)="Sub1=1,Sub2=2"

Set ^Global(2)="Sub1=2"

Set ^Global(2,3,4)="Sub1=2,Sub2=3,Sub3=4"

Set ^Global(3)="Sub1=3"

Set ^Global(4)="Sub1=4"

Set X="^Global"

Do {

Set X=$Q(@X) Q:X=""

Write X," = ",@X,!

} While X'=""

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

^Global(1) = Sub1=1

^Global(1,2) = Sub1=1,Sub2=2

^Global(2) = Sub1=2

^Global(2,3,4) = Sub1=2,Sub2=3,Sub3=4

^Global(3) = Sub1=3

^Global(4) = Sub1=4

Example ‑ Searching a Global for a string

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Set ^Global="start"

Set ^Global(1)="Sub1=1"

Set ^Global(1,2)="Sub1=1,Sub2=2"

Set ^Global(2)="Sub1=2"

Set ^Global(2,3,4)="Sub1=2,Sub2=3,Sub3=4"

Set ^Global(3)="Sub1=3"

Set ^Global(4)="Sub1=4"

Set X="^Global"

Set String="Sub2" ;set String to "Sub2"

Do {

Set X=$Q(@X) Q:X=""

If X[String!(@X[String) { ;does X contain "Sub2" or the contents of X

Write X," = ",@X,! ;contain "Sub2"

}

} While X'=""

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

^Global(1,2) = Sub1=1,Sub2=2

^Global(2,3,4) = Sub1=2,Sub2=3,Sub3=4

Example ‑ Set a list of variables to null

For X="VAR1","VAR2","VAR3" Set @X=""

Write;Write the variable just created

If you run the above code from the Terminal, you should get the following output.

VAR1=""

VAR2=""

VAR3=""

X="VAR3"

Example ‑ Set a number of variables to zero

For X="VAR1","VAR2","VAR3" Set @X=0

Write ;Write the variable just created

If you run the above code from the Terminal, you should get the following output.

VAR1=0

VAR2=0

VAR3=0

X="VAR3"

Example ‑ Set a number of variables to zero

Set COUNTERS="CNT1,CNT2,CNT3"

For I=1:1:$L(COUNTERS,",") Set @($P(COUNTERS,",",I))=0

Write;Write the counters just created

If you run the above code from the Terminal, you should get the following output.

CNT1=0

CNT2=0

CNT3=0

COUNTERS="CNT1,CNT2,CNT3"

I=3

Example ‑ Set a number of variables to zero, the easy way

Set (CNT1,CNT2,CNT3)=0

Write

If you run the above code from the Terminal, you should get the following output.

CNT1=0

CNT2=0

CNT3=0

Example ‑ Set a List of variables to zero

Set COUNTERS=$LB("CNT1","CNT2","CNT3")

For I=1:1:$LL(COUNTERS) Set @$LI(COUNTERS,I)=0

For I=1:1:3 Write !,@$LI(COUNTERS,I)

If you run the above code from the Terminal, you should get the following output.

0

0

0

Example ‑ Display a number of Pets

Set DOG="Rover"

Set CAT="Tiger"

Set FISH="Lamont"

For X="DOG","CAT","FISH" Write !,X,": ",@X

If you run the above code from the Terminal, you should get the following output.

DOG: Rover

CAT: Tiger

FISH: Lamont

Example ‑ Display a number of counters

Set CNT1=54

Set CNT2=65

Set CNT3=71

Set CNTRS="CNT1,CNT2,CNT3"

For I=1:1:$L(CNTRS,",") Write !,$P(CNTRS,",",I),": ",@($P(CNTRS,",",I))

If you run the above code from the Terminal, you should get the following output.

CNT1: 54

CNT2: 65

CNT3: 71

Example ‑ Display a list of counters

Set CNT1=54

Set CNT2=65

Set CNT3=71

Set COUNTERS=$LB("CNT1","CNT2","CNT3")

For I=1:1:$LL(COUNTERS) Write !,$LI(COUNTERS,I),": ",@($LI(COUNTERS,I))

If you run the above code from the Terminal, you should get the following output.

CNT1: 54

CNT2: 65

CNT3: 71

Example ‑ Display pieces in a List

Set PETS="Rover^Tiger^Lamont^Idiot"

For I=1:1:$L(PETS,"^") Write !,$P(PETS,"^",I)

If you run the above code from the Terminal, you should get the following output.

Rover

Tiger

Lamont

Idiot

Example ‑ De-piece a record

Set REC="Doe, John^Doe, Jane^Doe, Peter^Doe, Bambi"

Set LN="NAME1^NAME2^NAME3^NAME4"

For I=1:1:$L(LN,"^") Set @$P(LN,"^",I)=$P(REC,"^",I)

Write

If you run the above code from the Terminal, you should get the following output.

I=4

LN="NAME1^NAME2^NAME3^NAME4"

NAME1="Doe, John"

NAME2="Doe, Jane"

NAME3="Doe, Peter"

NAME4="Doe, Bambi"

REC="Doe, John^Doe, Jane^Doe, Peter^Doe, Bambi"

Example ‑ Example of returning a month's name

Set M=$LB("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")

Write !,$LI(M,5)

May

Write !,$LI(M,9)

Sep

*“Every man is a volumne, if you know how to read him.”*

*– William Ellery Channing*

# Date and Time System Supplied Functions

Example ‑ $HOROLOG or $H

Write !,$H

*61305,66339*

## 

Table ‑ Input Parameters for $ZDATETIME

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Parameter | Required/  Optional | Description |  |
| 1 | Date and Time | Required | Input date and time, in $HOROLOG format. |  |
| 2 | Date format | Optional | Format of the date to be returned. | See  Table 20‑2 for all possible Date formats. |
| 3 | Time format | Optional | Format of the time to be returned. | See  Table 20‑3 for possible Time formats. |
| 4 | Time Precision | Optional | Number of decimal places in time to be returned. |  |
| 5 | Month List | Optional | List of the month names. | See  Table 20‑4 for Date formats that can be used with Month List. |
| 6 | Year Option | Optional | Window to display the year in two-digits. | See  Table 20‑5 for Date formats that can be used with Year Option  See Table 20‑6 for Year Option formats. |
| 7 | 2 Digit Year Start | Optional | The start of the sliding window to display a two-digit year. |  |
| 8 | 2 Digit Year End | Optional | The end of the sliding window to display a two-digit year. |  |
| 9 | Minimum valid date | Optional | Lower limit of the range of valid dates. |  |
| 10 | Maximum valid date | Optional | The upper limit of the range of valid dates. |  |
| 11 | Error Option | Optional | This parameter suppresses error messages associated with invalid or out of range values. |  |

Example ‑ $ZDATETIME with One parameter

Write $ZDATETIME($H) ;display current date and time

*11/12/2008 11:47:45*

Write $ZDATETIME($H+5) ;display date 5 days from now

*11/17/2008*

Write $ZDATETIME($H-5) ;display date 5 days before now

*11/07/2008*

Write $ZDATETIME("61312,42310") ;display date and time from literal input

*11/12/2008 11:45:10*

Write $ZDATETIME("61312") ;display date from literal input, time

11/12/2008 ;is blank

Write $ZDATETIME(",42310") ;display time from literal input. Date of

*12/31/1840 11:45:10 ;blank is assumed zero, thus 12/31/1840*

Table ‑ Date Formats

|  |  |
| --- | --- |
| Date Format | Description |
| 1 | MM/DD/[YY]YY |
| 2 | DD Mmm [YY]YY |
| 3 | YYYY-MM-DD |
| 4 | DD/MM/[YY]YY |
| 5 | Mmm DD, YYYY |
| 6 | Mmm DD YYYY |
| 7 | Mmm DD [YY]YY |
| 8 | YYYYMMDD |
| 9 | Mmmmm D, YYYY |
| 10 | Day number for the week – 01=Sun, 02=Mon, 03=Tues, etc. |
| 11 | Abbreviated day name – Sun, Mon, Tues, etc. |
| 12 | Full day name – Sunday, Monday, Tuesday, etc. |

Example ‑ $ZDATETIME with Two parameters

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

For DateFormat=1:1:12 {

Write !,"Date Format: ",DateFormat," - ",$ZDATETIME($H,DateFormat)

}

Date Format: 1 - 11/12/2008 12:02:24

Date Format: 2 - 12 Nov 2008 12:02:24

Date Format: 3 - 2008-11-12 12:02:24

Date Format: 4 - 12/11/2008 12:02:24

Date Format: 5 - Nov 12, 2008 12:02:24

Date Format: 6 - Nov 12 2008 12:02:24

Date Format: 7 - Nov 12 2008 12:02:24

Date Format: 8 - 20081112 12:02:24

Date Format: 9 - November 12, 2008 12:02:24

Date Format: 10 - 3 12:02:24

Date Format: 11 - Wed 12:02:24

Date Format: 12 - Wednesday 12:02:24

Table ‑ Time Formats

|  |  |
| --- | --- |
| Time Format | Description |
| 1 | hh:mm:ss (24-hour clock) format | |
| 2 | hh:mm (24-hour clock) format | |
| 3 | hh:mm:ss[AM/PM] (12-hour clock) format | |
| 4 | hh:mm[AM/PM] (12-hour clock) format | |

Example ‑ $ZDATETIME with Three parameters

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Set DateFormat=1

For TimeFormat=1:1:4 {

Write !,"Time Format: ",TimeFormat," - "

Write $ZDATETIME($H,DateFormat,TimeFormat)

}

Time Format: 1 - 11/13/2008 12:12:40

Time Format: 2 - 11/13/2008 12:13

Time Format: 3 - 11/13/2008 12:12:40PM

Time Format: 4 - 11/13/2008 12:13PM

Example 20‑4 demonstrates the four time formats and their outputs.

Example ‑ $ZDATETIME with Four parameters – Time Precision

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

For Format=1:1:4 {

Write !,"Time Format: ",Format," - ",$ZDATETIME($H,,Format,2) ;2 decimal

}

Time Format: 1 - 11/13/2008 12:22:22.00 ;2 decimal precision

Time Format: 2 - 11/13/2008 12:22

Time Format: 3 - 11/13/2008 12:22:22.00PM ;2 decimal precision

Time Format: 4 - 11/13/2008 12:22PM

For Format=1:1:4 {

Write !,"Time Format: ",Format," - ",$ZDATETIME($H,,Format,4) ;4 decimals

}

Time Format: 1 - 11/13/2008 12:22:30.0000 ;4 decimal precision

Time Format: 2 - 11/13/2008 12:22

Time Format: 3 - 11/13/2008 12:22:30.0000PM;4 decimal precision

Time Format: 4 - 11/13/2008 12:22PM

Table ‑ Date Formats that can be used with MonthList

|  |  |
| --- | --- |
| Date Format | Description |
| 2 | DD Mmm [YY]YY |
| 5 | Mmm DD, YYYY |
| 6 | Mmm DD YYYY |
| 7 | Mmm DD [YY]YY |
| 9 | Mmmmm D, YYYY |

Example ‑ MonthList Delimiter

; Delimiter is a space

Set MonthList=" Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec"

; Delimiter is ^

Set MonthList="^Jan^Feb^Mar^Apr^May^Jun^Jul^Aug^Sep^Oct^Nov^Dec"

; Delimiter is ~

Set MonthList="~Jan~Feb~Mar~Apr~May~Jun~Jul~Aug~Sep~Oct~Nov~Dec"

Example ‑ $ZDATETIME with Five parameters - MonthList

;note: 1st character of the variable Months is the delimiter

Set Months=" Mon1 Mon2 Mon3 Mon4 Mon5 Mon6"

Set Months=Months\_" Mon7 Mon8 Mon9 Mon10 Mon11 Mon12" ;string of month names

Set StartDate=$ZDH("1/1/2005") ;$ZDH, which will be covered later

;converts a date into $H format

For Date=StartDate:30:StartDate+365 Write !,$ZDT(Date,2,,,Months)

01 Mon1 2005

31 Mon1 2005

02 Mon3 2005

01 Mon4 2005

01 Mon5 2005

31 Mon5 2005

30 Mon6 2005

30 Mon7 2005

29 Mon8 2005

28 Mon9 2005

28 Mon10 2005

27 Mon11 2005

27 Mon12 2005

Example ‑ $ZDATETIME with Five parameters - MonthList

Set Months=" January Febuary March April May June July"

Set Months=Months\_" August September October November December"

Set StartDate=$ZDH("1/1/2005") ;$ZDH, which will be covered later

;converts a date into $H format

For Date=StartDate:30:StartDate+365 Write !,$ZDT(Date,6,,,Months)

January 1 2005

January 31 2005

March 2 2005

April 1 2005

May 1 2005

May 31 2005

June 30 2005

July 30 2005

August 29 2005

September 28 2005

October 28 2005

November 27 2005

December 27 2005

Table ‑ Date Formats that can be used with Year Options

|  |  |
| --- | --- |
| Date Format | Description |
| 1 | MM/DD/[YY]YY |
| 2 | DD Mmm [YY]YY |
| 4 | DD/MM/[YY]YY |
| 7 | Mmm DD [YY]YY |

Table ‑ Year Option Formats

|  |  |
| --- | --- |
| Year Option | Description |
|  |  |
| 0 | Years 1900 through 1999 displayed with two-digit years, otherwise use 4 digit years. |
| 1 | 20th century dates displayed with two-digit years. |
| 2 | All years are two-digit years. |
| 3 | *Year Option* of 3 works with Parameters 6 and 7, start and end of two-digit year display. With this option, the start and end date are absolute years. |
| 4 | All years are four-digit years. |
| 5 | *Year Option* of 5 works with Parameters 6 and 7, start and end of two-digit year display. With this option, the start and end date are relative years. |
| 6 | All dates in the current century are two-digit years and all others are four-digit years. |

Example ‑ $ZDATETIME with Six parameters - Year Option of Zero

Set YearOpt=0 ; Years 1900 through 1999 are displayed with two-digit years,

; otherwise use 4 digit years.

Set Year(1)=$ZDH("1/1/1850") ;$ZDH, which will be covered later

Set Year(2)=$ZDH("1/1/1950") ;converts a date into $H format

Set Year(3)=$ZDH("1/1/2050")

For I=1:1:3 Set Date=Year(I) Write !,"Year ",I," – ",$ZDATETIME(Date,1,,,,YearOpt)

Year 1 – 01/01/1850 ;not years 1900 through 1999, four digit year

Year 2 – 01/01/50 ;years 1900 through 1999, two-digit year

Year 3 – 01/01/2050 ;not years 1900 through 1999, four digit year

Example ‑ $ZDATETIME with Six parameters - Year Option of One

Set YearOpt=1 ; 20th century dates are displayed with two-digit years,

; otherwise use 4 digit years.

Set Year(1)=$ZDH("1/1/1850") ;$ZDH, which will be covered later

Set Year(2)=$ZDH("1/1/1950") ;converts a date into $H format

Set Year(3)=$ZDH("1/1/2050")

For I=1:1:3 Set Date=Year(I) Write !,"Year ",I," – ",$ZDATETIME(Date,1,,,,YearOpt)

Year 1 – 01/01/1850 ;not the 20th century, four-digit year

Year 2 – 01/01/50 ;20th century, two-digit year

Year 3 – 01/01/2050 ;not the 20th century, four-digit year

Example ‑ $ZDATETIME with Six parameters - Year Option of Two

Set YearOpt=2 ; All years are two-digit years.

Set Year(1)=$ZDH("1/1/1850") ;$ZDH, which will be covered later

Set Year(2)=$ZDH("1/1/1950") ;converts a date into $H format

Set Year(3)=$ZDH("1/1/2050")

For I=1:1:3 Set Date=Year(I) Write !,"Year ",I," – ",$ZDATETIME(Date,1,,,,YearOpt)

Year 1 – 01/01/50

Year 2 – 01/01/50

Year 3 – 01/01/50

Example ‑ $ZDATETIME with Six parameters - Year Option of Four

Set YearOpt=4 ;All years are four-digit years.

Set Year(1)=$ZDH("1/1/1850") ;$ZDH, which will be covered later

Set Year(2)=$ZDH("1/1/1950") ;converts a date into $H format

Set Year(3)=$ZDH("1/1/2050")

For I=1:1:3 Set Date=Year(I) Write !,"Year ",I," – ",$ZDATETIME(Date,1,,,,YearOpt)

Year 1 – 01/01/1850

Year 2 – 01/01/1950

Year 3 – 01/01/2050

Example ‑ $ZDATETIME with Six parameters - Year Option of Six

Set YearOpt=6 ; All dates in the current century are two-digit years

; and all others are four-digit years.

Set Year(1)=$ZDH("1/1/1850") ;$ZDH, which will be covered later

Set Year(2)=$ZDH("1/1/1950") ;converts a date into $H format

Set Year(3)=$ZDH("1/1/2050")

For I=1:1:3 Set Date=Year(I) Write !,"Year ",I," – ",$ZDATETIME(Date,1,,,,YearOpt)

Year 1 – 01/01/1850 ;Not the current century, four-digit year

Year 2 – 01/01/1950 ;Not the current century, four-digit year

Year 3 – 01/01/50 ;Current century, two digit-year

Example ‑ $ZDATETIME with Eight parameters -Two Digit Year Start and End

Set YearOpt=3 ;YearOpt of 3, used with Start and End Date

Set Start=$ZDH("1/1/2002") ;Start Date of two-digit year

Set End=$ZDH("12/31/2005") ;End Date of two-digit year

;$ZDH, which will be covered later

;converts a date into $H format

For Date=($H-(10\*365)):365:+$H Write !,$ZDATETIME(Date,1,,,,YearOpt,Start,End)

12/30/1998

12/30/1999

12/29/2000

12/29/2001

12/29/02;year 2002

12/29/03;year 2003

12/28/04;year 2004

12/28/05;year 2005

12/28/2006

12/28/2007

12/27/2008

Example ‑ $ZDATETIME with Ten parameters – Minimum and Maximum Valid Date

Write $ZDATETIME($H-5,,,,,,,,$H-3) ; Minimum Valid Date of $H-3 is greater

**^** ; than $H-5

<VALUE OUT OF RANGE>

Set Date1=$ZDH("6/1/2008") ; Minimum Valid Date

Set Date2=$ZDH("5/31/2008") ; Date to be passed

; $ZDH, which will be covered later

; converts a date into $H format

Write $ZDATETIME(Date2,,,,,,,,Date1) ; Minimum Valid Date of 6/1/2008 is

^; greater than 5/31/2008

<VALUE OUT OF RANGE>

Write $ZDATETIME($H+4,,,,,,,,,$H) ; Maximum Valid Date of $H is less

^; than $H+4

<VALUE OUT OF RANGE>

Set Date1=$ZDH("6/1/2008") ; Maximum Valid Date

Set Date2=$ZDH("6/5/2008") ; Date to be passed

; $ZDH, which will be covered later

; converts a date into $H format

Write $ZDATETIME(Date2,,,,,,,,,Date1) ; Maximum Valid Date of 6/1/2008 is

^; less than 6/5/2008

<VALUE OUT OF RANGE>

Example ‑ Error Option demonstrated

Set ReturnValue=$ZDATETIME("61312,42310") ;Call with valid values

Write ReturnValue

"11/12/2008 11:45:10"

Set ReturnValue=$ZDATETIME("-61312,42310") ;Call with invalid values

Set ReturnValue=$ZDATETIME("-61312,42310")

^

<ILLEGAL VALUE>

Set ReturnValue=$ZDATETIME("-61312,42310",,,,,,,,,,1) ;Call supressing error

Write ReturnValue

1

Table ‑ Input Parameters for $ZDATETIMEH

|  |  |  |
| --- | --- | --- |
| Parameter | Required/  Optional | Description |
| Date and Time String | Required | Input date and time string |
| Date format | Optional | Input date format. |
| Time format | Optional | Input time format. |
| Time Precision | Optional | Number of decimals in the time. |

Example ‑ $ZDATETIMEH with One parameter

Write $ZDATETIMEH("11/12/2008 11:47:45") ;validate and convert to $H format

61312,42465

Write $ZDATETIMEH("11/12/2008") ;Time excluded, assume 0 (midnight)

61312,0

Write $ZDATETIMEH("13/12/2008 11:47:45") ;invalid month

^

<ILLEGAL VALUE>

Write $ZDATETIMEH("11/12/2008 11:61:45") ;invalid minutes

^

<ILLEGAL VALUE>

Table ‑ Date Formats

|  |  |
| --- | --- |
| Date Format | Description |
| 1 | MM/DD/[YY]YY |
| 2 | DD Mmm [YY]YY |
| 3 | YYYY-MM-DD |
| 4 | DD/MM/[YY]YY |
| 5 | Mmm D, YYYY |
| 6 | Mmm D YYYY |
| 7 | Mmm DD [YY]YY |
| 8 | YYYYMMDD |
| 9 | Mmmmm D, YYYY |

Example ‑ $ZDATETIMEH with Two parameters, date format 1

Date Format 1, MM/DD/YY[YY]

Set Date=$ZDATETIMEH("11/12/2008",1) ;Date Format 1, MM/DD/YYYY

Write Date

61312,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Set Date=$ZDATETIMEH("11/12/08",1) ;Date Format 1, MM/DD/YY

Write Date

24787,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 2

Date Format 2, DD Mmm YY[YY]

Set Date=$ZDATETIMEH("12 Nov 2008",2) ;Date Format 2, DD Mmm YYYY

Write Date

61312,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Set Date=$ZDATETIMEH("12 Nov 08",2) ;Date Format 2, DD Mmm YY

Write Date

24787,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 3

Date Format 3, YYYY-MM-DD

Set Date=$ZDATETIMEH("2008-11-12",3) ;Date Format 3, YYYY-MM-DD

Write Date

61312,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 4

Date Format 4, DD/MM/YY[YY]

Set Date=$ZDATETIMEH("05/12/2008",4) ;Date Format 4, DD/MM/YYYY

Write Date

61335,0

Write $ZDATETIME(Date)

12/05/2008 00:00:00

Set Date=$ZDATETIMEH("05/12/08",4) ;Date Format 4, DD/MM/YY

Write Date

24810,0

Write $ZDATETIME(Date)

12/05/08 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 5

Date Format 5, Mmm D, YYYY

Set Date=$ZDATETIMEH("Dec 5, 2008",5) ;Date Format 5, Mmm D, YYYY

Write Date

61335,0

Write $ZDATETIME(Date)

12/05/2008 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 6

Date Format 6, Mmm D YYYY

Set Date=$ZDATETIMEH("Dec 5 2008",6) ;Date Format 6, Mmm D YYYY

Write Date

61335,0

Write $ZDATETIME(Date)

12/05/2008 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 7

Date Format 7, Mmm DD YY[YY]

Set Date=$ZDATETIMEH("Nov 12 2008",7) ;Date Format 7, Mmm DD YY[YY]

Write Date

61312,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Set Date=$ZDATETIMEH("Nov 12 08",7) ;Date Format 7, Mmm DD YY

Write Date

24787,0

Write $ZDATETIME(Date)

11/12/08 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 8

Date Format 8, YYYYMMDD

Set Date=$ZDATETIMEH("20081112",8) ;Date Format 8, YYYYMMDD

Write Date

61312,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Example ‑ $ZDATETIMEH with Two parameters, date format 9

Date Format 9, Mmmmm DD, YYYY

Set Date=$ZDATETIMEH("November 12, 2008",9) ;Date Format 9, Mmmmm DD, YYYY

Write Date

61312,0

Write $ZDATETIME(Date)

11/12/2008 00:00:00

Table ‑ Time Formats

|  |  |
| --- | --- |
| Time Format | Description |
| 1 | hh:mm:ss (24-hour clock) format |
| 2 | hh:mm (24-hour clock) format |
| 3 | hh:mm:ss[AM/PM] (12-hour clock) format |
| 4 | hh:mm[AM/PM] (12-hour clock) format |

Example ‑ $ZDATETIMEH with Three parameters, time format 1

Time Format 1, hh:mm:ss (24-hour clock)

Set Date=$ZDATETIMEH("11/12/2008 10:22:30",1,1) ;hh:mm:ss (24-hour clock)

Write Date

61312,37350

Write $ZDATETIME(Date)

11/12/2008 10:22:30

Set Date=$ZDATETIMEH("11/12/2008 23:22:30",1,1) ;hh:mm:ss (24-hour clock)

Write Date

61312,84150

Write $ZDATETIME(Date)

11/12/2008 23:22:30

Example ‑ $ZDATETIMEH with Three parameters, time format 2

Time Format 2, hh:mm (24-hour clock)

Set Date=$ZDATETIMEH("11/12/2008 10:22",1,2) ; hh:mm:ss (24-hour clock)

Write Date

61312,37320

Write $ZDATETIME(Date)

11/12/2008 10:22:00

Set Date=$ZDATETIMEH("11/12/2008 23:22",1,2) ;hh:mm:ss (24-hour clock)

Write Date

61312,84120

Write $ZDATETIME(Date)

11/12/2008 23:22:00

Example ‑ $ZDATETIMEH with Three parameters, time format 3

Time Format 3, hh:mm:ss (12-hour clock)

Set Date=$ZDATETIMEH("11/12/2008 10:22:30",1,3) ;hh:mm:ss (12-hour clock)

Write Date

61312,37350

Write $ZDATETIME(Date)

11/12/2008 10:22:30

Set Date=$ZDATETIMEH("11/12/2008 10:22:30AM",1,3) ;hh:mm:ss (12-hour clock)

Write Date

61312,37350

Write $ZDATETIME(Date)

11/12/2008 10:22:30

Set Date=$ZDATETIMEH("11/12/2008 10:22:30PM",1,3) ;hh:mm:ss (12-hour clock)

Write Date

61312,80550

Write $ZDATETIME(Date)

11/12/2008 22:22:30

Example ‑ $ZDATETIMEH with Three parameters, time format 4

Time Format 4, hh:mm (12-hour clock)

Set Date=$ZDATETIMEH("11/12/2008 10:22",1,4) ;hh:mm (12-hour clock)

Write Date

61312,37320

Write $ZDATETIME(Date)

11/12/2008 10:22:00

Set Date=$ZDATETIMEH("11/12/2008 10:22AM",1,4) ;hh:mm (12-hour clock)

Write Date

61312,37320

Write $ZDATETIME(Date)

11/12/2008 10:22:00

Set Date=$ZDATETIMEH("11/12/2008 10:22PM",1,4) ;hh:mm (12-hour clock)

Write Date

61312,80520

Write $ZDATETIME(Date)

11/12/2008 22:22:00

Example ‑ $ZDATETIMEH with Four parameters

Time Format 1, hh:mm:ss.dddd

with Time Precision

Set Date=$ZDATETIMEH("11/12/2008 10:22:30.1234",1,1,4) ;hh:mm:ss.dddd

Write Date

61312,37350.1234

Write $ZDATETIME(Date,1,1,4)

11/12/2008 10:22:30.1234

Example ‑ $ZTIMESTAMP

Write $ZTIMESTAMP

61306,14437.019298

ddddd,ttttt.ffffff

Example ‑ $ZDATE and $ZDATEH

Write $ZDATE($H)

11/05/2008

Write $ZDATE($H-2)

11/03/2008

Write $ZDATE($H+5)

11/10/2008

Write $ZDATEH("11/05/2008")

61305

Example ‑ $ZTIME and $ZTIMEH

Write $ZTIME($P($H,",",2))

13:57:28

Write $ZTIMEH("13:57:28")

50284;number of seconds since midnight

Example ‑ Elapsed Time

This code cannot be run from the Terminal, it needs to be put in a Routine and then run the Routine.

Set Start=$P($H,",",2)-200

Set Now=$P($H,",",2)

Set Delta=Now-Start

If Delta<10 Set Delta="0"\_Delta

If Delta<60 Set Elapsed="00:00:"\_Delta

If Delta>59 {

Set Min=Delta\60,Sec=Delta#60

If Sec<10 Set Sec="0"\_Sec

If Min<10 Set Min="0"\_Min

If Min<60 Set Elapsed="00:"\_$E(Min,1,2)\_":"\_Sec

If Min>59 {

Set Hour=Min\60,Min=Min#60

If Sec<10 Set Sec="0"\_Sec

If Min<10 Set Min="0"\_Min

If Hour<10 Set Hour="0"\_Hour

Set Elapsed=Hour\_":"\_$E(Min,1,2)\_":"\_Sec

}

}

Write Elapsed

00:03:20

Example ‑ DeltaDate Routine

DeltaDate(HDATE1,HDATE2)

;------------------------------------------------------------------

; Examples of calling this routine:

; Write $$^DeltaDate(HDATE1,HDATE2)

; Set DIFF=$$^DeltaDate(HDATE1,HDATE2)

; The HDATE1 and HDATE2 parameters are in $H date and time format.

;------------------------------------------------------------------

New (HDATE1,HDATE2)

Set HDATE1=$G(HDATE1)

Set HDATE2=$G(HDATE2)

Set Day1=$P(HDATE1,",",1)

Set Day2=$P(HDATE2,",",1)

Set Time1=$P(HDATE1,",",2)

Set Time2=$P(HDATE2,",",2)

Set DeltaSec=(Day2-Day1)\*24\*60\*60+(Time2-Time1)

If DeltaSec<0 Set DeltaSec=DeltaSec\*-1

Set DeltaDay=0 If DeltaSec>86399 { ;86400 seconds in a day

Set DeltaDay=DeltaSec\86400

Set DeltaSec=DeltaSec-(DeltaDay\*86400)

}

Set DeltaHour=0 If DeltaSec>3600 {

Set DeltaHour=DeltaSec\3600

Set DeltaSec=DeltaSec-(DeltaHour\*3600)

}

Set DeltaMin=0 If DeltaSec>60 {

Set DeltaMin=DeltaSec\60

Set DeltaSec=DeltaSec-(DeltaMin\*60)

}

If $L(DeltaSec)=1 Set DeltaSec="0"\_DeltaSec

If $L(DeltaMin)=1 Set DeltaMin="0"\_DeltaMin

If $L(DeltaHour)=1 Set DeltaHour="0"\_DeltaHour

Q DeltaDay\_" - "\_DeltaHour\_":"\_DeltaMin\_"."\_DeltaSec

Example ‑ Calling DeltaDate Routine

Set HDATE1=$ZDATEH("12/30/1995",5) ;MM/DD/YYYY format

Set HDATE2=$ZDATEH("12/31/1995",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("12/30/95",5) ;MM/DD/YY format

Set HDATE2=$ZDATEH("12/31/95",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("30 Dec 1995",5) ;DD Mmm YYYY format

Set HDATE2=$ZDATEH("31 Dec 1995",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("1995-12-30",5) ;YYYY-MM-DD format

Set HDATE2=$ZDATEH("1995-12-31",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("31/12/95",4) ;DD/MM/YY format

Set HDATE2=$ZDATEH("30/12/95",4)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("31/12/1995",4) ;DD/MM/YYYY format

Set HDATE2=$ZDATEH("30/12/1995",4)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("Dec 30 1995",5) ;Mmm DD YYYY format

Set HDATE2=$ZDATEH("Dec 31 1995",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("19951230",5) ;YYYYMMDD format

Set HDATE2=$ZDATEH("19951231",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("951230",5) ;YYMMDD format

Set HDATE2=$ZDATEH("951231",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$ZDATEH("December 31, 1995",5) ;Mmmmmm DD, YYYY format

Set HDATE2=$ZDATEH("December 30, 1995",5)

Write !,$$^DeltaDate(HDATE1,HDATE2)

1 - 00:00 ;1 day difference

Set HDATE1=$H

Set HDATE2=$H

Set $P(HDATE1,",",2)=$ZTIMEH("14:05",2) ;HH:MM format

Set $P(HDATE2,",",2)=$ZTIMEH("14:10",2)

Write !,$$^DeltaDate(HDATE1,HDATE2)

0 - 00:05 ;five minute difference

Set HDATE1=$H

Set HDATE2=$H

Set $P(HDATE1,",",2)=$ZTIMEH("04:05AM",4) ;HH:MMAM/PM format

Set $P(HDATE2,",",2)=$ZTIMEH("04:05PM",4)

Write !,$$^DeltaDate(HDATE1,HDATE2)

0 - 12:00 ;12 hours difference

Set HDATE1=$H

Set HDATE2=$H

Set $P(HDATE1,",",2)=$ZTIMEH("04:05 AM",4) ;HH:MM AM/PM format

Set $P(HDATE2,",",2)=$ZTIMEH("04:05 PM",4)

Write !,$$^DeltaDate(HDATE1,HDATE2)

0 - 12:00 ;12 hour difference

Set HDATE1=$H

Set HDATE2=$H

Set $P(HDATE1,",",2)=$ZTIMEH("04:05:35 AM",4) ;HH:MM:SS AM/PM format

Set $P(HDATE2,",",2)=$ZTIMEH("04:05:40 PM",4)

Write !,$$^DeltaDate(HDATE1,HDATE2)

0 - 12:00.05 ;12 hour and 5 sec difference

# Object Technology Introduction

“There are persons who constantly clamor. They complain of oppression, speculation, and pernicious influence of wealth. They cry out loudly against all banks and corporations, and a means by which small capitalists become united in order to produce important and beneficial results. They carry on mad hostility against all established institutions. They would choke the fountain of human civilization.”

* Daniel Webster

Illustration ‑ Class Hierarchy, Inheritance and Polymorphism

Class: All Vehicles

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

As we go down the Class Hierarchy, Inheritance comes into play. Both Data and Methods from the upper level classes are available to the downstream classes. Polymorphism also comes into effect; the downstream classes modify Inherited Methods to fit their own needs.

Class: All Motorcycles

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

Class: All Cars

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

Class: All Trucks

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

Class: Fords

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

Class: Chevrolets

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

Class: Lumina

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

Class: Corvettes

Method: Steps for changing the oil

Method: Steps for changing a tire

Method: Steps for changing spark plugs

Etc

Table ‑ simple table with rows and columns

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table ‑ A Class (Table) of People

|  |  |  |  |
| --- | --- | --- | --- |
| Person Name | Person's Date of Birth | Person's Sex | Person's Address |
| Ben Dover | 5/5/1970 | M | 123 Main St. Somewhere |
| Ilene Dover | 10/2/1972 | F | 456 Simple Rd Anywhere |
| Jack Snow | 3/3/1956 | M | 789 First St. Sometown |

Table ‑ A Class (Table) of People with Object Ids

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object ID | Person Name | Person's Date of Birth | Person's Sex | Person's Address |
| 1 | Ben Dover | 5/5/1970 | M | 123 Main St. Somewhere |
| 2 | Ilene Dover | 10/2/1972 | F | 456 Simple Rd Anywhere |
| 3 | Jack Snow | 3/3/1956 | M | 789 First St. Sometown |

*“A proud man is always looking down on things and people; and, of course, as long as you are looking down, you cannot see something that is above you.”   
―* [C.S. Lewis](http://www.goodreads.com/author/show/1069006.C_S_Lewis)*,* [Mere Christianity](http://www.goodreads.com/work/quotes/801500)

# Object Class & Properties

Example ‑ Class and Object Definition for MyPackage.Actor

Class MyPackage.Actor Extends %Persistent  
 {

Property Name As %String;

}

Table ‑ Object Properties

|  |  |  |
| --- | --- | --- |
| Object Property | Data Type | Name we chose |
| Data Type | %String | Name |

Example ‑ Populating the "Name" Object Property

The "Name" Property is defined as a Data Type Class

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

 For Name="John Wayne","Jodie Foster" D CreateObject(Name)  
 For Name="Clint Eastwood","Julie Andrews" D CreateObject(Name)  
 For Name="Johnny Depp","Carol Burnett" D CreateObject(Name)  
 For Name="Will Smith","Ann Margaret" D CreateObject(Name)  
 For Name="Dean Martin","Ally Sheedy" D CreateObject(Name)  
 For Name="Humphrey Bogart","Katharine Hepburn" D CreateObject(Name)

Quit  
  
CreateObject(Name) [] Public {  
  Set ActorOref=##class(MyPackage.Actor).%New() ;create a new object   
  Set ActorOref.Name=Name ;populate the object property with a name   
  Do ActorOref.%Save() ;save the object  
  Set NewId=ActorOref.%Id() ;obtain the newly assigned Id

Write !,"Id: ",NewId ;write the Id of new Object  
  Write " – ",ActorOref.Name ;write the name of the new Object Property  
}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 – John Wayne

Id: 2 – Jodie Foster

Id: 3 – Clint Eastwood

Id: 4 – Julie Andrews

Id: 5 – Johnny Depp

Id: 6 – Carol Burnett

Id: 7 – Will Smith

Id: 8 – Ann Margaret

Id: 9 – Dean Martin

Id: 10 – Ally Sheedy

Id: 11 – Humphrey Bogart

Id: 12 – Katharine Hepburn

Example ‑ System Dump command for Orefs

Set ActorOref=##class(MyPackage.Actor).%OpenId(1)

Do $system.OBJ.Dump(ActorOref)

+----------------- general information ---------------

| oref value: 1

| class name: MyPackage.Actor

| reference count: 1

+----------------- attribute values ------------------

| %Concurrency = 1 <Set>

| Name = "John Wayne"

Example ‑ Test whether an Oref is valid

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;valid

If $IsObject(ActorOref) Write "Object is valid."

Object is valid.

If '$IsObject(ActorOref) Write "Object is not valid."

Set ActorOref=##class(MyPackage.Actor).%OpenId(1000) ;invalid

If $IsObject(ActorOref) Write "Object is valid."

If '$IsObject(ActorOref) Write "Object is not valid."

Object is not valid.

Example ‑ Global generated from Class MyPackage.Actor

ZW ^MyPackage.ActorD

^MyPackage.ActorD=12

^MyPackage.ActorD(1)=$lb("","John Wayne")

^MyPackage.ActorD(2)=$lb("","Jodie Foster")

^MyPackage.ActorD(3)=$lb("","Clint Eastwood")

^MyPackage.ActorD(4)=$lb("","Julie Andrews")

^MyPackage.ActorD(5)=$lb("","Johnny Depp")

^MyPackage.ActorD(6)=$lb("","Carol Burnett")

^MyPackage.ActorD(7)=$lb("","Will Smith")

^MyPackage.ActorD(8)=$lb("","Ann Margaret")

^MyPackage.ActorD(9)=$lb("","Dean Martin")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn")

Example ‑ Actor Class Redefinition - Index the "Name" Property

Class MyPackage.Actor Extends %Persistent  
 {

Property Name As %String [ Required ];  
  
 Index NameIndex On Name;  
 }

Example ‑ Index the Name Property in the MyPackage.Actor Class

Write ##class(MyPackage.Actor).%BuildIndices()

1

Example ‑ Example of the Index Global after indexing

ZW ^MyPackage.ActorI

^MyPackage.ActorI("NameIndex"," ALLY SHEEDY",10)=""

^MyPackage.ActorI("NameIndex"," ANN MARGARET",8)=""

^MyPackage.ActorI("NameIndex"," CAROL BURNETT",6)=""

^MyPackage.ActorI("NameIndex"," CLINT EASTWOOD",3)=""

^MyPackage.ActorI("NameIndex"," DEAN MARTIN",9)=""

^MyPackage.ActorI("NameIndex"," HUMPHREY BOGART",11)=""

^MyPackage.ActorI("NameIndex"," JODIE FOSTER",2)=""

^MyPackage.ActorI("NameIndex"," JOHN WAYNE",1)=""

^MyPackage.ActorI("NameIndex"," JOHNNY DEPP",5)=""

^MyPackage.ActorI("NameIndex"," JULIE ANDREWS",4)=""

^MyPackage.ActorI("NameIndex"," KATHARINE HEPBURN",12)=""

^MyPackage.ActorI("NameIndex"," WILL SMITH",7)=""

Example ‑ %ExistsId Method – see if an Object Id Exists

The "Name" Property is defined as a Data Type Class

Write ##class(MyPackage.Actor).%ExistsId(1) ;see if an Object Id Exists

1

Write ##class(MyPackage.Actor).%ExistsId(101) ;see if an Object Id Exists

0

Example ‑ Add another Actor to the MyPackage.Actor Class

The "Name" Property is defined as a Data Type Class

Set ActorOref=##class(MyPackage.Actor).%New() ;create a new object   
 Set ActorOref.Name="Jack Nicholson" ;populate with Jack Nicholson   
 Do ActorOref.%Save() ;save the object  
 Set NewId=ActorOref.%Id() ;obtain the newly assigned Id

Write !,"Id: ",NewId ;write ID of new Object  
 Write " – ",ActorOref.Name ;write the name of the new Object

If you run the above code from the Terminal, you should get the following output.

Id: 13 – Jack Nicholson

Example ‑ Delete an Actor from the MyPackage.Actor Class

The "Name" Property is defined as a Data Type Class

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

New SearchName,id,name

Set SearchName="Jack Nicholson" ;Name to search for

&sql(SELECT %Id,Name INTO :id, :name  
    FROM MyPackage.Actor  
    WHERE name = :SearchName)

If '$Data(id) {  
 Write "Object Not Found"  
 }  
 Else {  
 Set ReturnCode = ##class(MyPackage.Actor).%DeleteId(id) ;Delete Id  
 If ReturnCode=1 {   
 Write "Object Deleted."  
 }  
 Else {  
 Write "Object Not Deleted."  
 }

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Object Deleted.

Table ‑ Object Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Object Property | Data Type | Reference  Object Class  and Name | Name we chose |
| Name of  Data Type Class | %String |  | Name  (Name of Actor) |
| Reference to Persistent Objects |  | Persistent Object  MyPackage.Accountants | MyAccountant  (Accountants used by Actor) |

Table ‑ Primary Object Property Links to a Referenced Object Property

|  |  |  |
| --- | --- | --- |
| Primary Object Property | >> Link>> | Reference Object Property |
| Class: MyPackage.Actor   |  |  | | --- | --- | | Object Properties |  | | Name | Literal | | MyAccountant | Reference Property | | >>Link>> | Class: MyPackage.Accountants   |  |  | | --- | --- | | Object Properties |  | | AccountantName | Literal | |

Example ‑ Class Definition – Creating the Accountants Class

Class MyPackage.Accountants Extends %Persistent  
{  
  
Property AccountantName As %String [ Required ];  
  
Index AccountantNameIndex On AccountantName;  
  
}

Example ‑ Populate the AccountantName Property in the MyPackage.Accountants Class

The "AccountantName" Property is defined as a Data Type Class

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

  Do CreateObject("Fine Accountant")

Do CreateObject("Fair Accountant")

Do CreateObject("Really Bad Accountant")

Do CreateObject("Down Right #%&$ Accountant")

Quit  
  
 CreateObject(Name) [] Public {  
   Set AccountantOref=##class(MyPackage.Accountants).%New() ;create new object   
   Set AccountantOref.AccountantName=Name ;populate the object with a name   
   Do AccountantOref.%Save() ;save the object

   Set NewId=AccountantOref.%Id() ;obtain the newly assigned Id

Write !,"Id: ",NewId ;write the ID of new Object

   Write " – ",AccountantOref.AccountantName ;write the name of the new Object  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 - Fine Accountant

Id: 2 - Fair Accountant

Id: 3 - Really Bad Accountant

Id: 4 - Down Right #%&$ Accountant

Example ‑, Actor Class Redefinition – add Reference Property that points to the Accountants Class

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;

}

Example ‑ Modify Reference Property – associate Object Name MyAccountant with Class Accountants

The MyAccountant Property is defined as a Reference to a Persistent Object

Set ActorOref=##class(MyPackage.Actor).%OpenId(3) ;bring object Clint

;Eastwood into memory

Set AccountantOref=##class(MyPackage.Accountants).%OpenId(2) ;bring object

;Fair Accountant into memory

Set ActorOref.MyAccountant = AccountantOref ;associate MyAccountant

;(Fair Accountant) with Clint Eastwood

Do ActorOref.%Save() ;save the data

Write ActorOref.Name

Clint Eastwood

Write ActorOref.MyAccountant.AccountantName ;association made

Fair Accountant

Example ‑ Globals generated from Classes MyPackage.Actor and MyPackage.Accountants

ZW ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne")

^MyPackage.ActorD(2)=$lb("","Jodie Foster")

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2")

^MyPackage.ActorD(4)=$lb("","Julie Andrews")

^MyPackage.ActorD(5)=$lb("","Johnny Depp")

^MyPackage.ActorD(6)=$lb("","Carol Burnett")

^MyPackage.ActorD(7)=$lb("","Will Smith")

^MyPackage.ActorD(8)=$lb("","Ann Margaret")

^MyPackage.ActorD(9)=$lb("","Dean Martin")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn")

ZW ^MyPackage.AccountantsD

^MyPackage.AccountantsD(1)=$lb("","Fine Accountant")

^MyPackage.AccountantsD(2)=$lb("","Fair Accountant")

^MyPackage.AccountantsD(3)=$lb("","Really Bad Accountant")

^MyPackage.AccountantsD(4)=$lb("","Down Right #%&$ Accountant")

Example ‑ Modify Reference Property – associate the Fair Accountant with two more Actors

The MyAccountant Property is defined as a Reference to a Persistent Object

Set ActorOref1=##class(MyPackage.Actor).%OpenId(10) ;bring object Ally

;Sheedy into memory

Set AccountantOref=##class(MyPackage.Accountants).%OpenId(2) ;bring object the

;Fair Accountant into memory

Set ActorOref1.MyAccountant = AccountantOref ;associate MyAccountant (Fair

;Accountant) with Ally Sheedy

Do ActorOref1.%Save() ;save the data

Write ActorOref1.Name

Ally Sheedy

Write ActorOref1.MyAccountant.AccountantName ;association made

Fair Accountant

**- - - - - - - - - - - - - - - - - - -**

Set ActorOref2=##class(MyPackage.Actor).%OpenId(12) ;bring object Katharine

;Hepburn into memory

Set AccountantOref=##class(MyPackage.Accountants).%OpenId(2) ;bring object

;Fair Accountant into memory

Set ActorOref2.MyAccountant = AccountantOref ;associate MyAccountant (Fair

;Accountant) with Katharine Hepburn

Do ActorOref2.%Save() ;save the data

Write ActorOref2.Name

Katharine Hepburn

Write ActorOref2.MyAccountant.AccountantName ;association made

Fair Accountant

Example ‑ Modify Reference Property – Disassociate MyAccountant with Accountants

The MyAccountant Property is defined as a Reference to a Persistent Object

Set ActorOref=##class(MyPackage.Actor).%OpenId(10) ;bring object Ally

;Sheedy into memory

Set ActorOref.MyAccountant = "" ;disassociate MyAccountant from Accountants

Do ActorOref.%Save()

Write ActorOref.Name

Ally Sheedy

Write ActorOref.MyAccountant.AccountantName ;association broken

<>

Table ‑ Object Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Object Property | Data Type | Reference Object Class  and Name | Name we chose |
| Name of Data Type Class | %String |  | Name  (Name of Actor) |
| Reference to Persistent Objects |  | Persistent Object  MyPackage.Accountants | MyAccountant  (Accountant used by the Actor) |
| Reference to Embedded  Objects |  | Embedded or Serial Object  MyPackage.Address | MyHome  (Address of the Actor) |

Example ‑ Class Definition – Creating an Address Class of type %Serial

Class MyPackage.Address Extends %SerialObject  
{  
  
Property Street As %String(MAXLEN = 80);  
  
Property City As %String(MAXLEN = 30);  
  
Property State As %String(MINLEN = 2);

Property Zip As %String(MAXLEN = 10);  
  
}

Example ‑ Actor Class Redefinition – add Embedded Object Property "MyHome" that points to the Address Class

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;

Property MyHome As Address;

}

Example ‑ Modify or add Data to the MyHome Object Property

The MyHome Property is defined as a Reference to an Embedded Class

Set ActorOref=##class(MyPackage.Actor).%OpenId(4) ;bring object Julie Andrews

;into memory

Set ActorOref.MyHome.City = "Marlboro"

Set ActorOref.MyHome.State = "MA"

Set ActorOref.MyHome.Street = "123 Main St."

Set ActorOref.MyHome.Zip="01752"

Do ActorOref.%Save()

Example ‑ Global generated from Class MyPackage.Actor with Embedded Class MyPackage.Address

ZW ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne")

^MyPackage.ActorD(2)=$lb("","Jodie Foster")

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.","Marlboro","MA","01752"))

^MyPackage.ActorD(5)=$lb("","Johnny Depp")

^MyPackage.ActorD(6)=$lb("","Carol Burnett")

^MyPackage.ActorD(7)=$lb("","Will Smith")

^MyPackage.ActorD(8)=$lb("","Ann Margaret")

^MyPackage.ActorD(9)=$lb("","Dean Martin")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2")

# Embedded and Dynamic SQL

Table ‑ Tables, Columns, Rows and Cells.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table ‑ Table of Actors

|  |  |
| --- | --- |
| ID | Name |
| 1 | John Wayne |
| 2 | Jodie Foster |
| 3 | Client Eastwood |
| 4 | Julie Andrews |
| 5 | Johnny Depp |
| 6 | Carol Burnett |
| 7 | Will Smith |
| 8 | Ann Margaret |
| 9 | Dean Martin |
| 10 | Ally Sheedy |
| 11 | Humphrey Bogart |
| 12 | Katharine Hepburn |

Example ‑ Embedded SQL Routine to Display the Actors Class

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

SQLROUTINE ;Start of routine to hold Embedded SQL

New Id, Name ;Host variables, pass variables into Embedded SQL

;"&sql" - the SQL directive, start of Embedded SQL  
 &sql(  
 Declare MyCursor CURSOR FOR  
  SELECT ID, Name  
        INTO :Id, :Name  
       FROM MyPackage.Actor  
        ORDER BY Name)

  &sql(OPEN MyCursor) ;Open Cursor to start at top of data  
  &sql(FETCH MyCursor) ;Does initial Fetch of data

  While (SQLCODE = 0) { ;loop as long as we have a good SQLCODE  
   Write !,?5,"Id: ",Id ;good return code, display data  
  Write ?15,"Name: ",Name

&sql(FETCH MyCursor) ;do subsequent fetches  
   }

   &sql(CLOSE MyCursor) ;need to close cursor

Example ‑ Run Embedded SQL Routine to access the Actors Class

Do ^SQLROUTINE

Id: 10 Name: Ally Sheedy

Id: 8 Name: Ann Margaret

Id: 6 Name: Carol Burnett

Id: 3 Name: Clint Eastwood

Id: 9 Name: Dean Martin

Id: 11 Name: Humphrey Bogart

Id: 2 Name: Jodie Foster

Id: 1 Name: John Wayne

Id: 5 Name: Johnny Depp

Id: 4 Name: Julie Andrews

Id: 12 Name: Katharine Hepburn

Id: 7 Name: Will Smith

Example ‑ Embedded SQL Routine to Display the Actors Class with a Where clause

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

SQLROUTINE ;Start of routine to hold Embedded SQL

New Id, Name ;Host variables, pass variables into Embedded SQL

&sql(  
 Declare MyCursor CURSOR FOR  
  SELECT ID, Name  
        INTO :Id, :Name  
       FROM MyPackage.Actor

    WHERE Name = 'John Wayne'  
        ORDER BY Name)

  &sql(OPEN MyCursor) ;Open Cursor to start at top of data  
  &sql(FETCH MyCursor) ;Does initial Fetch of data

  While (SQLCODE = 0) { ;loop as long as we have a good SQLCODE  
   Write !,?5,"Id: ",Id ;good return code, display data  
  Write ?15,"Name: ",Name

&sql(FETCH MyCursor) ;do subsequent fetches  
   }

   &sql(CLOSE MyCursor) ;need to close cursor

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Id: 1 Name: John Wayne

Example ‑ Actor Class Redefinition

Class MyPackage.Actor Extends %Persistent  
 {

Property Name As %String [ Required ];  
  
 Index NameIndex On Name;  
  
 Property MyAccountant As Accountants;  
  
 Property MyHome As Address;  
  
 Property FavoriteColor As %String;

}

Example ‑ Embedded SQL Routine to Populate the Actor Class with Favorite Color

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

SQLROUTINE ;Start of routine to hold Embedded SQL

Do AddFavoriteColor("John Wayne","Blue")

Do AddFavoriteColor("Jodie Foster","Green")

Do AddFavoriteColor("Clint Eastwood","Cyan")

Do AddFavoriteColor("Julie Andrews","Brown")

Do AddFavoriteColor("Johnny Depp","Tan")

Do AddFavoriteColor("Carol Burnett","Red")   
 Do AddFavoriteColor("Will Smith","Navy")

Do AddFavoriteColor("Ann Margaret","Yellow")   
 Do AddFavoriteColor("Dean Martin","Green")

Do AddFavoriteColor("Ally Sheedy","Black")  
 Do AddFavoriteColor("Humphrey Bogart","Brown")

Do AddFavoriteColor("Katharine Hepburn","Blue")

Quit

;  
AddFavoriteColor(Actor,Color)  
 Write !,"Inserting Color: ",Color, " for Actor: ",Actor  
   
 &sql(UPDATE MyPackage.Actor (FavoriteColor)

VALUES (:Color)

WHERE Name=:Actor)

If SQLCODE'=0 {  
 Write !,"Error inserting Color into Actors"  
    Write "SQLCODE=",SQLCODE," : ",%msg  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Inserting Color: Blue for Actor: John Wayne

Inserting Color: Green for Actor: Jodie Foster

Inserting Color: Cyan for Actor: Clint Eastwood

Inserting Color: Brown for Actor: Julie Andrews

Inserting Color: Tan for Actor: Johnny Depp

Inserting Color: Red for Actor: Carol Burnett

Inserting Color: Navy for Actor: Will Smith

Inserting Color: Yellow for Actor: Ann Margaret

Inserting Color: Green for Actor: Dean Martin

Inserting Color: Black for Actor: Ally Sheedy

Inserting Color: Brown for Actor: Humphrey Bogart

Inserting Color: Blue for Actor: Katharine Hepburn

Example ‑ Global generated from Class MyPackage.Actor

ZW ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne","",$lb("","","",""), "Blue")

^MyPackage.ActorD(2)=$lb("","Jodie Foster","",$lb("","","",""), "Green")

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2",$lb("","","",""), "Cyan")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.","Marlboro","MA",

"01752"), "Brown")

^MyPackage.ActorD(5)=$lb("","Johnny Depp","",$lb("","","",""), "Tan")

^MyPackage.ActorD(6)=$lb("","Carol Burnett","",$lb("","","",""), "Red")

^MyPackage.ActorD(7)=$lb("","Will Smith","",$lb("","","",""), "Navy")

^MyPackage.ActorD(8)=$lb("","Ann Margaret","",$lb("","","",""), "Yellow")

^MyPackage.ActorD(9)=$lb("","Dean Martin","",$lb("","","",""), "Green")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","",$lb("","","",""), "Black")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart","",$lb("","","",""), "Brown")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2",$lb("","","",""), "Blue")

Example ‑ Embedded SQL Routine to Display the Actors Class "Reference to Persistent Object" Property

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

SQLROUTINE ;Start of routine to hold Embedded SQL

New Id, Name, MyAccountantName ;Host variables, pass variables into Embedded SQL  
  
 &sql(  
 Declare MyCursor CURSOR FOR  
  SELECT ID, Name, MyAccountant->AccountantName   
        INTO :Id, :Name, :MyAccountantName  
       FROM MyPackage.Actor  
        ORDER BY MyPackage.Actor.Name)  
  
  &sql(OPEN MyCursor) ;Open Cursor to start at top of data  
  &sql(FETCH MyCursor) ;Does initial Fetch of data  
  
  While (SQLCODE = 0) { ;loop as long as we have a good SQLCODE  
   Write !,?5,"Id: ",Id ;good return code, display data  
  Write ?15,"Name: ",Name  
  If MyAccountantName'="" {  
   Write ?40,"Accountant Name : ",MyAccountantName  
  }  
  &sql(FETCH MyCursor) ;do subsequent fetches  
   }  
     
 &sql(CLOSE MyCursor) ;need to close cursor

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Id: 10 Name: Ally Sheedy

Id: 8 Name: Ann Margaret

Id: 6 Name: Carol Burnett

Id: 3 Name: Clint Eastwood Accountant Name : Fair Accountant

Id: 9 Name: Dean Martin

Id: 11 Name: Humphrey Bogart

Id: 2 Name: Jodie Foster

Id: 1 Name: John Wayne

Id: 5 Name: Johnny Depp

Id: 4 Name: Julie Andrews

Id: 12 Name: Katharine Hepburn Accountant Name : Fair Accountant

Id: 7 Name: Will Smith

Example ‑ Embedded SQL Routine to Display the Actors Class "Reference to Embedded Class" Property, MyHome – MyPackage.Address Class

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

SQLROUTINE ;Start of routine to hold Embedded SQL  
  
 New Id, Name, Street, City, State, Zip ;Host variables, passed to Embedded SQL

&sql(   
 Declare MyCursor CURSOR FOR   
  SELECT ID, Name, MyHome\_Street, MyHome\_City, MyHome\_State, MyHome\_Zip  
        INTO :Id, :Name, :Street, :City, :State, :Zip  
       FROM MyPackage.Actor  
        ORDER BY MyPackage.Actor.Name)  
  
  &sql(OPEN MyCursor) ;Open Cursor to start at top of data  
  &sql(FETCH MyCursor) ;Does initial Fetch of data  
  
  While (SQLCODE = 0) { ;loop as long as we have a good SQLCODE  
   Write !,?5,"Id: ",Id ;good return code, display data  
  Write ?15,"Name: ",Name  
  If Street'="" Write ?40,"Street : ",Street  
  If City'=""   Write !,?40,"City : ",City  
  If State'=""  Write !,?40,"State : ",State  
  If Zip'=""    Write !,?40,"Zip : ",Zip

  &sql(FETCH MyCursor) ;do subsequent fetches  
   }  
  
   &sql(CLOSE MyCursor) ;need to close cursor

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Id: 10 Name: Ally Sheedy

Id: 8 Name: Ann Margaret

Id: 6 Name: Carol Burnett

Id: 3 Name: Clint Eastwood

Id: 9 Name: Dean Martin

Id: 11 Name: Humphrey Bogart

Id: 2 Name: Jodie Foster

Id: 1 Name: John Wayne

Id: 5 Name: Johnny Depp

Id: 4 Name: Julie Andrews Street : 123 Main St.

City : Marlboro

State : MA

Zip : 01752

Id: 12 Name: Katharine Hepburn

Id: 7 Name: Will Smith

Example ‑ Dynamic SQL Routine to Display the Actors Class

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

 SQLROUTINE ;Start of routine to hold Dynamic SQL

   Set MyQuery="SELECT \* FROM MyPackage.Actor" ;Define the Query  
  
   Set ActorOref = ##class(%SQL.Statement).%New() ;New Instance of %SQL.Statement   
  
   Set Status = ActorOref.%Prepare(MyQuery) ;Prepare the Query  
  
   Set ResultSet = ActorOref.%Execute() ;Execute the Query  
  
   While ResultSet.%Next() { ;Return each row of the Query  
   Write !,?5,"Id: ",ResultSet.Id  ;good return code, display data  
    Write ?15,ResultSet.Name  
    Write ?40,ResultSet.FavoriteColor  
   }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Id: 1 John Wayne Blue

Id: 2 Jodie Foster Green

Id: 3 Clint Eastwood Cyan

Id: 4 Julie Andrews Brown

Id: 5 Johnny Depp Tan

Id: 6 Carol Burnett Red

Id: 7 Will Smith Navy

Id: 8 Ann Margaret Yellow

Id: 9 Dean Martin Green

Id: 10 Ally Sheedy Black

Id: 11 Humphrey Bogart Brown

Id: 12 Katharine Hepburn Blue

Example ‑ Dynamic SQL Routine to Display the Actors Class

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

 SQLROUTINE ;Start of routine to hold Dynamic SQL

Set MyQuery="SELECT \* FROM MyPackage.Actors" ;Define the Query

  Set ActorOref = ##class(%SQL.Statement).%New() ;New Instance of %SQL.Statement

  Set Status = ActorOref.%Prepare(MyQuery) ;Prepare the Query

If +Status'=1 {

Write !,"Invalid Status returned from the %Prepare Call."

Write !,"Status: ",Status

Quit 0

}

  Set ResultSet = ActorOref.%Execute() ;Execute the Query

   While ResultSet.%Next() { ;Return each row of the Query  
   Write !,?5,"Id: ",ResultSet.Id  ;good return code, display data  
    Write ?15,ResultSet.Name  
    Write ?40,ResultSet.FavoriteColor  
   }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Invalid Status returned from the %Prepare Call.

Status: 0

¤â% Table 'MYPACKAGE.Actor' not found

Example ‑ Dynamic SQL Routine to Display the Actors Class using %Display

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

 SQLROUTINE ;Start of routine to hold Dynamic SQL

Set MyQuery="SELECT \* FROM MyPackage.Actor" ;Define the Query

  Set ActorOref = ##class(%SQL.Statement).%New() ;New Instance of %SQL.Statement

  Set Status = ActorOref.%Prepare(MyQuery) ;Prepare the Query

  Set ResultSet = ActorOref.%Execute() ;Execute the Query

   Do ResultSet.%Display() ;Display

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

ID FavoriteColor MyAccountant Name MyHome\_City

MyHome\_StateMyHome\_Street MyHome\_Zip

1 Blue John Wayne

2 Green Jodie Foster

3 Cyan 2 Clint Eastwood

4 Brown Julie Andrews Marlboro MA 123 Main St.01752

5 Tan Johnny Depp

6 Red Carol Burnett

7 Navy Will Smith

8 Yellow Ann Margaret

9 Green Dean Martin

10 Black Ally Sheedy

11 Brown Humphrey Bogart

12 Blue 2 Katharine Hepburn

12 Rows(s) Affected

Example ‑ Dynamic SQL Routine to Display the Actors Class "Reference to Persistent Object" Property

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

 SQLROUTINE ;Start of routine to hold Dynamic SQL

Set MyQuery="SELECT Id, Name, FavoriteColor, MyAccountant->AccountantName "  
 Set MyQuery=MyQuery\_" FROM MyPackage.Actor" ;Define the Query  
  
 Set ActorOref = ##class(%SQL.Statemenew Instance of %SQL.Statement  
  
 Set Status = ActorOref.%Prepare(MyQuery) ;Prepare the Query  
  
 If Status=1 {  
     Set ResultSet = ActorOref.%Execute() ;Execute the Query  
        While ResultSet.%Next() {         ;Return each row of the Query  
        Write !,?5,"Id: ",ResultSet.Id  
         Write ?15,ResultSet.Name  
         Write ?35,ResultSet.FavoriteColor  
         If ResultSet.AccountantName'="" {  
      Write ?45,"Accountant Name : ",ResultSet.AccountantName  
  }  
     }  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Id: 1 John Wayne Blue

Id: 2 Jodie Foster Green

Id: 3 Clint Eastwood Cyan Accountant Name : Fair Accountant

Id: 4 Julie Andrews Brown

Id: 5 Johnny Depp Tan

Id: 6 Carol Burnett Red

Id: 7 Will Smith Navy

Id: 8 Ann Margaret Yellow

Id: 9 Dean Martin Green

Id: 10 Ally Sheedy Black

Id: 11 Humphrey Bogart Brown

Id: 12 Katharine Hepburn Blue Accountant Name : Fair Accountant

Example ‑ Dynamic SQL Routine to Display the Actors Class "Reference to Embedded Class" Property

To run this code you must put the code in a routine (SQLROUTINE), save the routine, and then run the routine from the terminal.

 SQLROUTINE ;Start of routine to hold Dynamic SQL

Set MyQuery="SELECT ID, Name, MyHome\_Street, MyHome\_City, MyHome\_State, MyHome\_Zip"  
   Set MyQuery=MyQuery\_" FROM MyPackage.Actor" ;Define the Query  
  
   Set ActorOref = ##class(%SQL.Statement).%New() ;New Instance of %SQL.Statement   
  
   Set Status = ActorOref.%Prepare(MyQuery) ;Prepare the Query  
  
   Set ResultSet = ActorOref.%Execute() ;Execute the Query  
  
   While ResultSet.%Next() { ;Return each row of the Query  
       Write !,?5,"Id: ",ResultSet.Id  
       Write ?15,"Name: ",ResultSet.Name  
       If ResultSet.MyHomeStreet'="" Write ?35,"Street : ",ResultSet.MyHomeStreet  
     If ResultSet.MyHomeCity'=""   Write !,?35,"City : ",ResultSet.MyHomeCity  
     If ResultSet.MyHomeState'=""  Write !,?35,"State : ",ResultSet.MyHomeState  
     If ResultSet.MyHomeZip'=""    Write !,?35,"Zip : ",ResultSet.MyHomeZip  
   }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Do ^SQLROUTINE

Id: 1 Name: John Wayne

Id: 2 Name: Jodie Foster

Id: 3 Name: Clint Eastwood

Id: 4 Name: Julie Andrews Street : 123 Main St.

City : Marlboro

State : MA

Zip : 01752

Id: 5 Name: Johnny Depp

Id: 6 Name: Carol Burnett

Id: 7 Name: Will Smith

Id: 8 Name: Ann Margaret

Id: 9 Name: Dean Martin

Id: 10 Name: Ally Sheedy

Id: 11 Name: Humphrey Bogart

Id: 12 Name: Katharine Hepburn

# Object Property Datatypes

Table ‑ Common Datatypes

|  |  |  |
| --- | --- | --- |
| Datatypes | Source | Description |
| %String | %Library.String | The %String *Datatype*  class represents a string of characters. |
| %Name | %Library.Name | The %Name *Datatype* class represents a string containing a name in the format:"lastname,firstname". |
| %Numeric | %Library.Numeric | The %Numeric *Datatype* class represents a real number. |
| %Integer | %Library.Integer | The %Integer *Datatype* class represents an integer value. |
| %Date | %Library.Date | The %Date *Datatype* class represents a date. The value of the %Date is in $H format. |
| %Time | %Library.Time | The %Time *Datatype* is the number of seconds past midnight. |
| %Status | %Library.Status | The %Status *Datatype* represents a status code. |

Example ‑ Define User.Datatypes1 Class with DataItem1 Property

Class User.Datatypes1 Extends %Persistent  
{  
  
Property DataItem1 As %String [ Required ];

}

Example ‑ Property Data Validation – Required - Demonstration

Set oref=##class(User.Datatypes1).%New() ;create a new oref or Object Reference

Set oref.DataItem1="" ;DataItem1 set to null

Set status=oref.%Save() ;attempt to save

If status’=1 Do $system.OBJ.DisplayError(status) ;save not successful

ERROR #5659: Property 'User.Datatypes1::DataItem1(1@User.Datatypes1,ID=)'

Required

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set oref.DataItem1="somevalue" ;DataItem1 set to somevalue

Set status=oref.%Save() ;attempt to save

If status’=1 Do $system.OBJ.DisplayError(status)

<>;save successful

Write status

1;save successful

Example ‑ Define Properties with Data Types, Min and Max Length

Class User.Datatypes1 Extends %Persistent  
{  
  
Property DataItem1 As %String [ Required ];

Property DataItem2 As %String(MAXLEN = 10, MINLEN = 5);

}

Example ‑ Property Data Validation – Min and Max Length - Demonstration

Set oref=##class(User.Datatypes1).%New() ;create a new oref or Object Reference

Set oref.DataItem1="SomeValue" ;DataItem1 is required

Set oref.DataItem2="abc" ;DataItem2 set to less than min length

Set status=oref.%Save() ;attempt to save

If status’=1 Do $system.OBJ.DisplayError(status) ;save not successful

ERROR #7202: Datatype value 'abc' length less than MINLEN allowed of 5

> ERROR #5802: Datatype validation failed on property 'User.Datatypes1:DataItem2'

, with value equal to "abc"

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set oref.DataItem2="abcde" ;DataItem2 set to the specified minimum length

Set status=oref.%Save() ;attempt to save

If status’=1 Do $system.OBJ.DisplayError(status)

<>;save successful

Write status

1;save successful

Example ‑ Property Data Validation – Min and Max Length - IsValid

Set oref=##class(User.Datatypes1).%New() ;create a new oref or Object Reference

Set oref.DataItem1="SomeValue" ;DataItem1 is required

Set oref.DataItem2="abc" ;DataItem2 set to less than min length

Set status=##class(User.Datatypes1).DataItem2IsValid(oref.DataItem2)

If status’=1 Do $system.OBJ.DisplayError(status) ;validation not successful

ERROR #7202: Datatype value 'abc' length less than MINLEN allowed of 5

Example ‑ Define Properties with Data Types, Pattern

Class User.Datatypes1 Extends %Persistent  
{  
  
Property DataItem1 As %String [ Required ];

Property DataItem2 As %String(MAXLEN = 10, MINLEN = 5);

Property SSN As %String(PATTERN = "3N1""-""2N1""-""4N");

}

Example ‑ Property Data Validation – Pattern - Demonstration

Set oref=##class(User.Datatypes1).%New() ;create a new oref or Object Reference

Set oref.DataItem1="SomeValue" ;DataItem1 is required

Set oref.DataItem2="abcde" ;DataItem2 set to the min length

Set oref.SSN="123-45-66" ;SSN set to wrong pattern

Set status=##class(User.Datatypes1).SSNIsValid(oref.SSN)

If status’=1 Do $system.OBJ.DisplayError(status) ;validation not successful

ERROR #7209: Datatype value '123-45-66' does not match PATTERN '3N1"-"2N1"-"4N'

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set oref.SSN="123-45-6677" ;SSN set to correct pattern

Set status=##class(User.Datatypes1).SSNIsValid(oref.SSN)

If status’=1 Do $system.OBJ.DisplayError(status) ;validation successful

<>

Write status

1

Example ‑ Define Properties with Data Types, Valuelist

Class User.Datatypes1 Extends %Persistent  
{  
  
Property DataItem1 As %String [ Required ];

Property DataItem2 As %String(MAXLEN = 10, MINLEN = 5);

Property SSN As %String(PATTERN = "3N1""-""2N1""-""4N");

Property StatusCode As %String(VALUELIST = "-Success-Fail-Pend");

}

Example ‑ Property Data Validation – Value - Demonstration

Set oref=##class(User.Datatypes1).%New() ;create a new oref or Object Reference

Set oref.DataItem1="SomeValue" ;DataItem1 is required

Set oref.DataItem2="abcde" ;DataItem2 set to the min length

Set oref.SSN="123-45-6677" ;SSN set to correct pattern

Set oref.StatusCode="WrongValue" ;Set StatusCode to a wrong value

Set status=##class(User.Datatypes1).StatusCodeIsValid(oref.StatusCode)

If status’=1 Do $system.OBJ.DisplayError(status) ;validation not successful

ERROR #7205: Datatype value 'WrongValue' not in VALUELIST '-Success-Fail-Pend'

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set oref.StatusCode="Success" ;Set StatusCode to a correct value

Set status=##class(User.Datatypes1).StatusCodeIsValid(oref.StatusCode)

If status’=1 Do $system.OBJ.DisplayError(status) ;validation successful

<>

Write status

1

Example ‑ Define a Custom Datatype Class for Name

Class User.NameDatatype Extends %Persistent  
{  
  
Property FirstName As %String;

Property MiddleInitial As %String;

Property LastName As %String;

}

Example ‑ Add Name Property to User.Datatypes1

Class User.Datatypes1 Extends %Persistent  
{  
  
Property DataItem1 As %String [ Required ];

Property DataItem2 As %String(MAXLEN = 10, MINLEN = 5);

Property SSN As %String(PATTERN = "3N1""-""2N1""-""4N");

Property StatusCode As %String(VALUELIST = "-Success-Fail-Pend");

Property Name As User.NameDatatype;

}

Example ‑ Property Data Validation – Custom Datatype - Demonstration

Set oref=##class(User.Datatypes1).%New() ;create a new Object Reference

Set oref.DataItem1="SomeValue" ;DataItem1 is required

Set oref.DataItem2="abcde" ;DataItem2 set to the min length

Set oref.SSN="123-45-6677" ;SSN set to correct pattern

Set oref.StatusCode="Success" ;Set StatusCode to a correct value

Set NameOref=##class(User.NameDatatype).%New() ;create a new oref for

;the NameDatatype Class

Set NameOref.FirstName="Ben" ;Populate the Name Class

Set NameOref.MiddleInitial="T"

Set NameOref.LastName="Dover"

Set oref.Name=NameOref ;Point the Name Property to

;the Customer Name Datatype

Write oref.%Save()

1

Write oref.Name.FirstName ;Reference the Name

Ben

Write oref.Name.MiddleInitial

T

Write oref.Name.LastName

Dover

# Class & Object Properties - Collection List of Data Types

Table ‑ A Collection List of Shirts

|  |  |
| --- | --- |
| Slot Number | Value |
| 1 | RedShirt |
| 2 | WhiteShirt |
| 3 | BlueShirt |

Table ‑ A Collection Array of Hats

|  |  |
| --- | --- |
| Key | Value |
| 1 | Bowler |
| 2 | Straw |
| Mesh | FruitMesh |
| Top | TopHat |

Table ‑ Table Collection Methods

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Chapter 25 | Chapter 26 | Chapter 27 | Chapter 28 | Chapter 29 | Chapter 30 |  |
| Actor used | John Wayne | Jodi Foster | Johnny Depp | Carol Burnett | Dean Martin | Ann Margaret |  |
| Associated data | MyShirts | MyHats | MyContacts | MyClients | MyRentals | MyPets |  |
|  | Data Types | | References to Persistent Objects | | References to Embedded Objects | |  |
| Methods | Collection  Lists of  Data  Types | Collection  Arrays of  Data  Types | Collection  Lists of References  To Persistent Objects | Collection  Arrays of  References  To Persistent  Objects | Collection  Lists of  References  To  Embedded  Objects | Collection  Arrays of  References  To  Embedded  Objects | Description |
| Clear() | Example  25 – 2 | Example  26 – 2 | Example  27 – 4 | Example  28 – 4 | Example  29 – 3 | Example  30 – 3 | Clears or Deletes the Collection of Elements |
| Count() | Example  25 – 2 | Example  26 – 2 | Example  27 – 4 | Example  28 – 4 | Example 29– 3 | Example  30 – 3 | Returns the Number of Elements in the Collection |
|  |  |  |  |  |  |  |  |
| GetAt(Slot) | Example  25 – 5 |  |  |  |  |  | Returns the Element associated with a Slot |
| GetAt(Key) |  | Example  26 – 6 | Example  27 – 7 | Example  28 – 8 | Example  29 – 6 | Example  30 – 7 | Returns the Element associated with a Key |
|  |  |  |  |  |  |  |  |
| Find(Element, Slot) | Example  25 – 6 |  |  |  |  |  | Finds the Element starting at the Slot |
| Find(Element, Key) |  | Example  26 – 7 |  |  |  |  | Finds the Element starting at the Key |
| Find(String) |  |  | Example  27 – 8 | Example  28 – 9 | Example  29 – 7 | Example  30 – 8 | Code to emulate a Find. Finds the associated Key for a String |
| IsDefined(Key) |  | Example  26 – 5 |  | Example  28 – 7 |  | Example  30 – 6 | Returns a 1 if the Key is defined otherwise 0 |
|  |  |  |  |  |  |  |  |
| Next(Slot) | Example  25 – 7 |  |  |  |  |  | Returns the next Slot position |
| Next(Key) |  | Example 26 – 9 | Example  27 – 9 | Example  28 – 10 | Example  29 –8 | Example  30 –9 | Returns the Element for the next Key |
| Previous(Slot) | Example  25 – 8 |  |  |  |  |  | Returns the previous Slot position |
| Previous(Key) |  | Example 26 – 10 | Example  27 – 10 | Example  28 – 11 | Example  29 – 9 | Example  30 –10 | Returns the Element for the previous Key |
|  |  |  |  |  |  |  |  |
| GetNext(.Slot)  (slot passed by reference) | Example  25 – 9 |  |  |  |  |  | Returns the Element for the next Slot |
| GetNext(.Key)  (key passed by reference) |  | Example  26 – 11 | Example  27 – 11 | Example  28 – 12 | Example  29 – 10 | Example  30 –11 | Returns the Element for the next Key |
| GetPrevious(.Slot)  (slot passed by reference) | Example  25 – 10 |  |  |  |  |  | Returns the Element for the previous Slot |
| GetPrevious(.Key)  (Key passed by reference) |  | Example  26 – 12 | Example  27 – 12 | Example  28 – 13 | Example  29 – 11 | Example  30 –12 | Returns the Element for the previous Key |
|  |  |  |  |  |  |  |  |
| Insert(Element) | Example  25 – 3 |  |  |  |  |  | Inserts an Element at the end of the collection |
| Insert(Oref) |  |  | Example  27 – 5 |  | Example  29 – 4 |  | Inserts an Oref at the end of the collection |
| InsertAt(Element,Slot) | Example  25 – 11 |  |  |  |  |  | Inserts an Element into a Collection at a specified Slot |
| InsertAt(Oref,Key) |  |  | Example  27 – 13 |  | Example  29 – 12 |  | Insert an Oref into a Collection at a specific Key |
|  |  |  |  |  |  |  |  |
| InsertOrdered(Value) | Example  25 – 13 |  |  |  |  |  | Inserts an Element into a collection |
|  |  |  |  |  |  |  |  |
| SetAt(Element, Slot) | Example  25 – 12 |  |  |  |  |  | Sets an Element at the specified Slot |
| SetAt(Element, Key) |  | Example 26 – 3  Example  26 – 8 | Example  27 – 14 |  |  |  | Set or Replace an Element at a specific Key |
| SetAt(Oref,Key) |  |  |  | Example  28 – 5  Example  28 – 14 | Example  29 – 4 | Example  30 – 4  Example  30 – 13 | Set an Oref at the specific Key |
| RemoveAt(Slot) | Example  25 – 14 |  |  |  |  |  | Remove the Element at the specified Slot position |
| RemoveAt(Key) |  | Example 26 – 13 | Example  27 – 15 | Example  28 – 15 | Example  29 – 14 | Example  30 – 14 | Remove the Element associated with a Key |
|  |  |  |  |  |  |  |  |
| Select data with Embedded SQL | Example  25 – 15 | Example  26 – 14 | Example  27 - 16 | Example  28 – 16 | Example  29 – 15 | Example  30 - 15 |  |
| Display data with Embedded SQL | Example  25 – 15 | Example  26 – 14 | Example  27 - 16 | Example  28 – 16 | Example  29 – 15 | Example  30 - 15 |  |
| Select data with Dynamic SQL | Example  25 – 16 | Example  26 - 15 | Example  27 - 17 | Example  28 - 17 | Example  29 - 16 | Example  30 - 16 |  |
| Display data with Dynamic SQL | Example  25 – 16 | Example  26 - 15 | Example  27 - 17 | Example  28 - 17 | Example  29 - 16 | Example  30 - 16 |  |

Table ‑ Object Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Object Property | Data Type | Name we chose | Special Considerations |
| Data Types | %String | Name  (Name of Actor) |  |
| Collection List of Data Types | %String | MyShirts  (Shirts belonging to an Actor) | Total length cannot exceed 32k |
| Collection Array of Data Types | %String | MyHats  (Hats belonging to an Actor) | No length limit  In SQL must be handled as a Child table  Must specify a Key when accessing |

### 

Example ‑ Actor Class Redefinition - Include the Collection List Property: MyShirts

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;  
  
Property MyHome As Address;  
  
Property FavoriteColor As %String;  
  
Property MyShirts As list Of %String;  
  
}

Example ‑ Clear and Count Methods - Deletes all elements and show the count

The MyShirts Property is defined as a Collection List of Data Types

Clear Method - Clears or Deletes the Collection of Elements

Count Method - Returns the Number of Elements in the Collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John

;Wayne into memory

Do ActorOref.MyShirts.Clear() ;clear list of shirts  
 Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 0

1

Example ‑ Insert Method – Insert three shirts into the Collection

The MyShirts Property is defined as a Collection List of Data Types

Insert Method – Inserts an Element at the end of the collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Do ActorOref.MyShirts.Insert("BlueShirt") ;insert RedShirt  
 Do ActorOref.MyShirts.Insert("RedShirt") ;insert WhiteShirt  
 Do ActorOref.MyShirts.Insert("WhiteShirt") ;insert BlueShirt

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 3

1

Example ‑ Global generated from Class MyPackage.Actor

zw ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne","",$lb("","","",""),"Blue",$lb("BlueShirt","RedShirt","WhiteShirt"))

^MyPackage.ActorD(2)=$lb("","Jodie Foster","",$lb("","","",""),"Green")

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2",$lb("","","",""),"Cyan")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.","Marlboro",

"MA","01752"),"Brown")

^MyPackage.ActorD(5)=$lb("","Johnny Depp","",$lb("","","",""),"Tan")

^MyPackage.ActorD(6)=$lb("","Carol Burnett","",$lb("","","",""),"Red")

^MyPackage.ActorD(7)=$lb("","Will Smith","",$lb("","","",""),"Navy")

^MyPackage.ActorD(8)=$lb("","Ann Margaret","",$lb("","","",""),"Yellow")

^MyPackage.ActorD(9)=$lb("","Dean Martin","",$lb("","","",""),"Green")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","",$lb("","","",""),"Black")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart","",$lb("","","",""),"Brown")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2",$lb("","","",""),"Blue")

Example ‑ GetAt Method – Display the Collection List

The MyShirts Property is defined as a Collection List of Data Types

GetAt Method - Returns the Element associated with a Slot

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

For Slot=1:1:ActorOref.MyShirts.Count() { ;Display each element  
 Write !,"Slot: ",Slot ;of the Collection List  
 Write " - ",ActorOref.MyShirts.GetAt(Slot) ;Property: MyShirts  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Slot: 1 - BlueShirt

Slot: 2 - RedShirt

Slot: 3 - WhiteShirt

Example ‑ Find a shirt in the Collection List starting at a Slot number

The MyShirts Property is defined as a Collection List of Data Types

Find Method - Finds the Element starting at the Slot

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Set Slot=ActorOref.MyShirts.Find("RedShirt",0) ;find RedShirt starting

;at Slot 0 or the

;beginning of the collection

Write !,"Slot: ",Slot," - ",ActorOref.MyShirts.GetAt(Slot) ;display RedShirt

Slot: 2 - RedShirt

Set Slot=ActorOref.MyShirts.Find("WhiteShirt",0) ;find WhiteShirt starting

;at Slot 0 or the

;beginning of the collection

Write !,"Slot: ",Slot," - ",ActorOref.MyShirts.GetAt(Slot) ;display WhiteShirt

Slot: 3 – WhiteShirt

Set Slot=ActorOref.MyShirts.Find("BlueShirt",2) ;find BlueShirt starting at

;Slot 2, since BlueShirt is in

;Slot 1, nothing is found.

Write !,"Slot: ",Slot

Slot:

Table ‑ Traversing Methods

|  |  |  |
| --- | --- | --- |
| Method |  | How Key or Slot is passed |
| Next | Traversing Forward | Key or Slot passed by Value |
| Previous | Traversing Backward | Key or Slot Passed by Value |
| GetNext | Traversing Forward | Key or Slot Passed by Reference |
| GetPrevious | Traversing Backward | Key or Slot Passed by Reference |

Example ‑ Next Method – Returns the Element for the next Slot

The MyShirts Property is defined as a Collection List of Data Types

Next Method - Returns the next Slot position

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne   
 ;into memory

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

Set Slot = "" Do { ;start with slot null

Set Slot=ActorOref.MyShirts.Next(Slot) ;get the next slot   
   If Slot'="" {  
 Write !,"Slot: ",Slot ;display the slot number  
 Write " - ",ActorOref.MyShirts.GetAt(Slot) ;display the shirt

   }   
 } While (Slot '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Slot: 1 - BlueShirt

Slot: 2 - RedShirt

Slot: 3 – WhiteShirt

Example ‑ Previous Method – Returns the Element for the previous Slot

The MyShirts Property is defined as a Collection List of Data Types

Previous Method - Returns the previous Slot position

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne   
 ;into memory

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

Set Slot = "" Do { ;start with slot null

Set Slot=ActorOref.MyShirts.Previous(Slot) ;get the previous slot  
   If Slot'="" {  
 Write !,"Slot: ",Slot ;display the slot number  
     Write " - ",ActorOref.MyShirts.GetAt(Slot) ;display the shirt  
   }   
 } While (Slot '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Slot: 3 - WhiteShirt

Slot: 2 - RedShirt

Slot: 1 – BlueShirt

Example ‑ GetNext Method – Returns the Element for the next Slot

The MyShirts Property is defined as a Collection List of Data Types

GetNext Method - Returns the Element for the next Slot

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

Set Slot = "" Do { ;start with beginning slot  
 Set Shirt=ActorOref.MyShirts.GetNext(.Slot) ;get next slot  
 If Slot'="" {

Write !,"Slot: ",Slot ;display the slot

Write " ",Shirt ;display the shirt

}  
 } While (Slot '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Slot: 1 - BlueShirt

Slot: 2 - RedShirt

Slot: 3 – WhiteShirt

The MyShirts Property is defined as a Collection List of Data Types

GetPrevious Method - Returns the Element for the previous Slot

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

Set Slot = "" Do { ;start with beginning slot  
 Set Shirt=ActorOref.MyShirts.GetPrevious(.Slot) ;get previous slot  
 If Slot'="" {

Write !,"Slot: ",Slot ;display the slot

Write " ",Shirt ;display the shirt

}  
 } While (Slot '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Slot: 3 - WhiteShirt

Slot: 2 - RedShirt

Slot: 1 – BlueShirt

Example ‑ InsertAt Method – Insert a shirt between the 1st and 2nd shirts

The MyShirts Property is defined as a Collection List of Data Types

InsertAt Method - Inserts an Element into a Collection at a specified Slot

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Do ActorOref.MyShirts.InsertAt("PurpleShirt",2) ;Insert PurpleShirt into the

; 2rd Slot pushing the other

;shirts out

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

For Slot=1:1:ActorOref.MyShirts.Count() { ;Display each element  
 Write !,"Slot: ",Slot ;of Collection List  
 Write " - ",ActorOref.MyShirts.GetAt(Slot) ;Property: MyShirts  
 }

Write !,ActorOref.%Save() ;Save the Object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Slot: 1 - BlueShirt

Slot: 2 - PurpleShirt

Slot: 3 - RedShirt

Slot: 4 – WhiteShirt

1

Example ‑ SetAt Method – Replace a specific shirt

The MyShirts Property is defined as a Collection List of Data Types

SetAt Method - Sets an Element at the specified Slot

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Do ActorOref.MyShirts.SetAt("CyanShirt",2) ;Change the value in Slot 2

;to CyanShirt

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

For Slot=1:1:ActorOref.MyShirts.Count() { ;Display each element  
 Write !,"Slot: ",Slot ;of Collection List  
 Write " - ",ActorOref.MyShirts.GetAt(Slot) ;Property: MyShirts  
 }

Write !,ActorOref.%Save() ;Save the Object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Slot: 1 - BlueShirt

Slot: 2 - CyanShirt

Slot: 3 - RedShirt

Slot: 4 – WhiteShirt

1

Example ‑ InsertOrdered Method – Add a shirt to the collection

The MyShirts Property is defined as a Collection List of Data Types

InsertOrdered Method - Inserts an Element into the collection

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Do ActorOref.MyShirts.InsertOrdered("PinkShirt") ;Insert PinkShirt into

;the Collections

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

For Slot=1:1:ActorOref.MyShirts.Count() { ;Display each element  
 Write !,"Slot: ",Slot ;of Collection List  
 Write " - ",ActorOref.MyShirts.GetAt(Slot) ;Property: MyShirts  
 }

Write !,ActorOref.%Save() ;Save the Object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 5

Slot: 1 - BlueShirt

Slot: 2 - CyanShirt

Slot: 3 - PinkShirt

Slot: 4 - RedShirt

Slot: 5 – WhiteShirt

1

Example ‑ RemoveAt Method – Remove an Element at a Slot

The MyShirts Property is defined as a Collection List of Data Types

RemoveAt Method - Remove the Element at the specified Slot position

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(1) ;bring object John Wayne

;into memory

Do ActorOref.MyShirts.RemoveAt(3) ;Remove Shirt at Slot 3

Write !,"Count: ",ActorOref.MyShirts.Count() ;count of elements

For I=1:1:ActorOref.MyShirts.Count() {  
 Write !,"Slot: ",I  
 Write " - ",ActorOref.MyShirts.GetAt(I)  
 }

Write !,ActorOref.%Save() ;Save the Object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Slot: 1 - BlueShirt

Slot: 2 - CyanShirt

Slot: 3 - RedShirt

Slot: 4 – WhiteShirt

1

Example ‑ Select MyShirts Data using Embedded SQL

The MyShirts Property is defined as a Collection List of Data Types

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

New name, myshirts

&sql(Declare MyCursor CURSOR FOR  
  SELECT Name, MyShirts  
  INTO :name, :myshirts  
  FROM MyPackage.Actor  
  WHERE FOR SOME %ELEMENT(MyShirts) (%Value='BlueShirt')  
  ORDER BY Name)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !,"Name: ",name  
 If myshirts'="" {  
 For I=1:1:$LL(myshirts) {  
 Write !,?20,$LI(myshirts,I)  
   }  
  }

&sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Name: John Wayne

BlueShirt

CyanShirt

RedShirt

WhiteShirt

Example ‑ Display MyShirts Data using Embedded SQL

The MyShirts Property is defined as a Collection List of Data Types

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

 New name,myshirts   
 &sql(Declare MyCursor CURSOR FOR  
  SELECT Name, MyShirts  
  INTO :name, :myshirts  
  FROM MyPackage.Actor  
  ORDER BY Name)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !,"Name: ",name  
 If myshirts'="" {  
 For I=1:1:$LL(myshirts) {  
 Write !,?20,$LI(myshirts,I)  
   }  
  }

&sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Name: Ally Sheedy

Name: Ann Margaret

Name: Carol Burnett

Name: Clint Eastwood

Name: Dean Martin

Name: Humphrey Bogart

Name: Jodie Foster

Name: John Wayne

BlueShirt

CyanShirt

RedShirt

WhiteShirt

Name: Johnny Depp

Name: Julie Andrews

Name: Katharine Hepburn

Name: Will Smith

Example ‑ Select MyShirts Data using Dynamic SQL

The MyShirts Property is defined as a Collection List of Data Types

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

 Set MyQuery="SELECT Name,MyShirts FROM MyPackage.Actor" ;Define the Query

 Set MyQuery=MyQuery\_" WHERE FOR SOME %ELEMENT(MyShirts) (%Value='BlueShirt')"  
   
 Set ResultSet=##class(%ResultSet).%New("%DynamicQuery:SQL")  
 ;Create a new Instance of %ResultSet  
  
 Set Status=ResultSet.Prepare(MyQuery) ;Prepare the Query  
 If Status'=1 Write "Status'=1, return from Prepare Statement" Quit  
  
 Set Status=ResultSet.Execute() ;Execute the Query  
 If Status'=1 Write "Status'=1, return from Execute Statement" Quit  
  
 While ResultSet.Next() { ;Process the Query results  
     Write !,ResultSet.Data("Name")  
     Set myshirts=ResultSet.Data("MyShirts")  
     If myshirts'="" {  
   For I=1:1:$LL(myshirts) {  
 Write !,?15,$LI(myshirts,I)  
   }  
     }  
  }

 Set SC=ResultSet.Close() ;Close the Query

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

John Wayne

BlueShirt

CyanShirt

RedShirt

WhiteShirt

Example ‑ Display MyShirts Data using Dynamic SQL

The MyShirts Property is defined as a Collection List of Data Types

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

 Set MyQuery="SELECT Name,MyShirts FROM MyPackage.Actor" ;Define the Query  
   
 Set ResultSet=##class(%ResultSet).%New("%DynamicQuery:SQL")  
 ;Create a new Instance of %ResultSet  
  
 Set Status=ResultSet.Prepare(MyQuery) ;Prepare the Query  
 If Status'=1 Write "Status'=1, return from Prepare Statement" Quit  
  
 Set Status=ResultSet.Execute() ;Execute the Query  
 If Status'=1 Write "Status'=1, return from Execute Statement" Quit  
  
 While ResultSet.Next() { ;Process the Query results  
  Write !,ResultSet.Data("Name")  
     Set myshirts=ResultSet.Data("MyShirts")  
     If myshirts'="" {  
   For I=1:1:$LL(myshirts) {  
 Write !,?15,$LI(myshirts,I)  
   }  
     }  
  }

 Set SC=ResultSet.Close() ;Close the Query

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

John Wayne

BlueShirt

CyanShirt

RedShirt

WhiteShirt

Jodie Foster

Clint Eastwood

Julie Andrews

Johnny Depp

Carol Burnett

Will Smith

Ann Margaret

Dean Martin

Ally Sheedy

Humphrey Bogart

Katharine Hepburn

# Class & Object Properties - Collection Array of Data Types

Table ‑ A Collection List of Shirts

|  |  |
| --- | --- |
| Slot Number | Value |
| 1 | RedShirt |
| 2 | WhiteShirt |
| 3 | BlueShirt |

Table ‑ A Collection Array of Hats

|  |  |
| --- | --- |
| Key | Value |
| 1 | Bowler |
| 2 | Straw |
| Mesh | FruitMesh |
| Top | TopHat |

Table ‑ Object Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Object Property | Data Type | Name we chose | Special Considerations |
| Data Types | %String | Name  (Name of Actor) |  |
| Collection List of Data Types | %String | MyShirts  (Shirts belonging to an Actor) | Total length cannot exceed 32k |
| Collection Array of Data Types | %String | MyHats  (Hats belonging to an Actor) | No length limit  In SQL must be handled as a Child table  Must specify a Key when accessing |

Example ‑ Actor Class Redefinition - Include the Collection Array Property: MyHats

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;  
  
Property MyHome As Address;  
  
Property FavoriteColor As %String;  
  
Property MyShirts As list Of %String;  
  
Property MyHats As array Of %String;  
  
}

Example ‑ Clear and Count Methods - Deletes all elements and show the count

The MyHats Property is defined as a Collection Array of Data Types

Clear Method – Clears or Deletes the Collection of Elements

Count Method - Returns the Number of Elements in the Collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Do ActorOref.MyHats.Clear() ;clear list of hats  
 Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 0

1

Example ‑ SetAt Method – Add Four Hats to the Collection

The MyHats Property is defined as a Collection Array of Data Types

SetAt Method - Set or Replace an Element at the specific Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Do ActorOref.MyHats.SetAt("Bowler",1) ;keyed by numeric 1  
 Do ActorOref.MyHats.SetAt("Straw",2) ;keyed by numeric 2  
 Do ActorOref.MyHats.SetAt("FruitMesh","Mesh") ;keyed by alpha "Mesh"

Do ActorOref.MyHats.SetAt("TopHat","Top") ;keyed by alpha "Top"

Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 4

1

Example ‑ Global generated from Class MyPackage.Actor

zw ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne","",$lb("","","",""),"Blue",$lb("BlueShirt","CyanShirt","RedShirt",""WhiteShirt"))

^MyPackage.ActorD(2)=$lb("","Jodie Foster","",$lb("","","",""),"Green","")

^MyPackage.ActorD(2,"MyHats",1)="Bowler"

^MyPackage.ActorD(2,"MyHats",2)="Straw"

^MyPackage.ActorD(2,"MyHats","Mesh")="FruitMesh"

^MyPackage.ActorD(2,"MyHats","Top")="TopHat"

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2",$lb("","","",""),"Cyan")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.","Marlboro",

"MA","01752"),"Brown")

^MyPackage.ActorD(5)=$lb("","Johnny Depp","",$lb("","","",""),"Tan")

^MyPackage.ActorD(6)=$lb("","Carol Burnett","",$lb("","","",""),"Red")

^MyPackage.ActorD(7)=$lb("","Will Smith","",$lb("","","",""),"Navy")

^MyPackage.ActorD(8)=$lb("","Ann Margaret","",$lb("","","",""),"Yellow")

^MyPackage.ActorD(9)=$lb("","Dean Martin","",$lb("","","",""),"Green")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","",$lb("","","",""),"Black")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart","",$lb("","","",""),"Brown")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2",$lb("","","",""),"Blue")

Example ‑ IsDefined Method – See if a Key is defined

The MyHats Property is defined as a Collection Array of Data Types

IsDefined Method - Returns a 1 if the Key is defined otherwise 0

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Write ActorOref.MyHats.IsDefined("Mesh") ;Is key "Mesh" defined? - Yes  
1

Write ActorOref.MyHats.IsDefined("Top") ;Is key "Top" defined? - Yes  
1

Write ActorOref.MyHats.IsDefined(1) ;Is key 1 defined? - Yes  
1

Write ActorOref.MyHats.IsDefined(3) ;Is key 3 defined? - No  
0

Example ‑ GetAt Method – Returns the Element for a Key

The MyHats Property is defined as a Collection Array of Data Types

GetAt Method - Returns the Element associated with a Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Write ActorOref.MyHats.GetAt(2) ;Get Hat at Key 2  
Straw

Set Key="Mesh"

Write ActorOref.MyHats.GetAt(Key) ;Get Hat at Key "Mesh"  
FruitMesh

Example ‑ Find a hat in the collection starting at Key

The MyHats Property is defined as a Collection Array of Data Types

Find Method - Finds the Element starting at the Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Write ActorOref.MyHats.Find("TopHat") ;find the Element TopHat and display

Top ;the key

Write ActorOref.MyHats.Find("Bowler") ;find the Element Bowler and display

1 ;the key

Write ActorOref.MyHats.Find("Bowler",2) ;find the Element Bowler starting at

<> ;key 2, since Bowler is at key 1,

;nothing is found

Example ‑ SetAt Method – Set or Replace an Element at a specific Key

The MyHats Property is defined as a Collection Array of Data Types

SetAt Method - Set or Replace an Element at the specific Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Do ActorOref.MyHats.SetAt("BigBirdHat",2) ;replace Hat at Key 2

Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Set Key = "" Do { ;cycle through hats  
 Set Hat=ActorOref.MyHats.GetNext(.Key) ;get hat at specified key  
 If Key'="" {

Write !,"Key and hat: "

Write Key," - ",Hat ;display key and hat

}  
 } While (Key '= "")

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key and hat: 1 - Bowler

Key and hat: 2 - BigBirdHat

Key and hat: Mesh - FruitMesh

Key and hat: Top – TopHat

1

Table ‑ Traversing Methods

|  |  |  |
| --- | --- | --- |
| Method |  | How Key or Slot is passed |
| Next | Traversing Forward | Key or Slot passed by Value |
| Previous | Traversing Backward | Key or Slot Passed by Value |
| GetNext | Traversing Forward | Key or Slot Passed by Reference |
| GetPrevious | Traversing Backward | Key or Slot Passed by Reference |

Example ‑ Next Method – Returns the Element for the next Key

The MyHats Property is defined as a Collection Array of Data Types

Next Method - Returns the Element for the next Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster  
 ;into memory

Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyHats.Next(Key) ;get the next Key  
    If Key'="" {  
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyHats.GetAt(Key) ;display the hat  
    }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 1 - Bowler

Key: 2 - BigBirdHat

Key: Mesh - FruitMesh

Key: Top – TopHat

Example ‑ Previous Method – Returns the Element for the previous Key

The MyHats Property is defined as a Collection Array of Data Types

Previous Method - Returns the Element for the previous Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster  
 ;into memory

Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Set Key = "" Do { ;start with Key ""

Set Key=ActorOref.MyHats.Previous(Key) ;get the previous Key  
    If Key'="" {  
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyHats.GetAt(Key) ;display the hat  
    }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: Top - TopHat

Key: Mesh - FruitMesh

Key: 2 - BigBirdHat

Key: 1 – Bowler

Example ‑ GetNext Method – Returns the Element for the next Key

The MyHats Property is defined as a Collection Array of Data Types

GetNext Method - Returns the Element for the next Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Set Key = "" Do { ;start with beginning key  
 Set Hat=ActorOref.MyHats.GetNext(.Key) ;get next key (passed by reference)  
 If Key'="" {

Write !,"Key and hat: "

Write Key," - ",Hat ;display key and hat  
 }

} While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key and hat: 1 - Bowler

Key and hat: 2 – BigBirdHat

Key and hat: Mesh - FruitMesh

Key and hat: Top - TopHat

Example ‑ *GetPrevious Method – Returns the Element for the Previous Key*

The MyHats Property is defined as a Collection Array of Data Types

GetPrevious Method - Returns the Element for the previous Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Set Key = "" Do { ;start with beginning key  
 Set Hat=ActorOref.MyHats.GetPrevious(.Key) ;get previous key

If Key'="" { ;(passed by reference)

Write !,"Key and hat: "

Write Key," - ",Hat ;display key with hat

}  
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key and hat: Top - TopHat

Key and hat: Mesh - FruitMesh

Key and hat: 2 - BigBirdHat

Key and hat: 1 - Bowler

Example ‑ RemoveAt Method – Remove the Element associated with a Key

The MyHats Property is defined as a Collection Array of Data Types

RemoveAt Method - Remove the Element associated with the Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(2) ;bring object Jodi Foster

;into memory

Do ActorOref.MyHats.RemoveAt("Mesh") ;remove Element associated

;with key Mesh

Write !,"Count: ",ActorOref.MyHats.Count() ;count of elements

Set Key = "" Do {  
 Set Hat=ActorOref.MyHats.GetNext(.Key)  
 If Key'="" {

Write !,"Key and hat: "

Write Key," - ",Hat ;display key and hat

}  
 } While (Key '= "")

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Key and hat: 1 - Bowler

Key and hat: 2 - BigBirdHat

Key and hat: Top – TopHat

1

Example ‑ Display MyHats Data using Embedded SQL

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

 New actorname,myhats  
 &sql(Declare MyCursor CURSOR FOR  
  SELECT Actor->Name, MyHats  
  INTO :actorname, :myhats  
  FROM MyPackage.Actor\_MyHats   
  ORDER BY Name)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !,"Name: ",actorname  
  Write ?25,"MyHats: ",myhats  
 &sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Name: Jodie Foster MyHats: Bowler

Name: Jodie Foster MyHats: BigBirdHat

Name: Jodie Foster MyHats: TopHat

Example ‑ MyHats Data using Dynamic SQL

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set MyQuery="SELECT Actor->Name,MyHats FROM MyPackage.Actor\_MyHats"  
   
 Set ResultSet=##class(%ResultSet).%New("%DynamicQuery:SQL")  
 ;Create a new Instance of %ResultSet  
  
 Set SC=ResultSet.Prepare(MyQuery) ;Prepare the Query  
 Set SC=ResultSet.Execute() ;Execute the Query  
  
 While ResultSet.Next() { ;Process the Query results  
     Write !,ResultSet.Data("Name")," - "  
     Write ResultSet.Data("MyHats")  
  }  
  
 Set SC=ResultSet.Close() ;Close the Query

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Jodie Foster - Bowler

Jodie Foster - BigBirdHat

Jodie Foster – TopHat

*“I am right 97% of the time, and don't care about the other 4%.”*

# Class & Object Properties - Collection List of References to Persistent Objects

Table 27 - Object Properties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object Property | Data Type | Reference  Object Class  and Name | Name we chose | Special Considerations |
| Data Types | %String |  | Name  (Name of Actor) |  |
| Collection List of Data Types | %String |  | MyShirts  (Shirts belonging to an Actor) | Total length cannot exceed 32k |
| Collection Array of Data Types | %String |  | MyHats  (Hats belonging to an Actor) | No length limit  In SQL must be handled as a Child table  Must specify a Key when accessing |
| Collection List of References to Persistent Objects |  | Persistent Objects | MyContacts  (Contacts of the Actor) |  |
| Collection Array of References to Persistent Objects |  | Persistent Objects | MyClients  (Clients of the Actor) |  |

Example ‑ Class Definition – Creating the Contacts Class

Class MyPackage.Contacts Extends %Persistent  
{  
  
Property ContactName As %String [ Required ];  
  
Index ContactNameIndex On ContactName;  
  
}

Example ‑ Populate the Contacts Class

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

For Name="Contact1","Contact2","Contact3" Do CreateObject(Name)

For Name="Contact4","Contact5","Contact6" Do CreateObject(Name)

Quit

CreateObject(Name) [] Public {  
  Set ContactsOref=##class(MyPackage.Contacts).%New() ;create a new object reference  
  Set ContactsOref.ContactName=Name ;populate the object with a name   
  Do ContactsOref.%Save() ;save the object  
  Set NewId=ContactsOref.%Id() ;get the newly assigned ID

Write !,"Id: ",NewId ;write ID of new Object  
  Write " – ",ContactsOref.ContactName ;write the name of the new Object  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 – Contact1

Id: 2 – Contact2

Id: 3 – Contact3

Id: 4 – Contact4

Id: 5 – Contact5

Id: 6 – Contact6

Example ‑ Actor Class Redefinition – add Reference Property that will point to the Contacts Class

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;  
  
Property MyHome As Address;  
  
Property MyShirts As list Of %String;  
  
Property MyHats As array Of %String;  
  
Property MyContacts As list Of Contacts;  
  
}

Example ‑ Clear and Count Methods - Deletes all elements and show the count

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

Clear Method – Clears or Deletes the Collection of Elements

Count Method - Returns the Number of Elements in the Collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

;  
 Do ActorOref.MyContacts.Clear() ;clear list of contacts  
 Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 0

1

Example ‑ Insert Method – Insert four Contacts into the Collection

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

Insert Method - Inserts an Oref at the end of the collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Set Contact1Oref=##class(MyPackage.Contacts).%OpenId(1) ;bring object Contact1

;into memory

Set Contact2Oref=##class(MyPackage.Contacts).%OpenId(2) ;bring object Contact2

;into memory

Set Contact3Oref=##class(MyPackage.Contacts).%OpenId(3) ;bring object Contact3

;into memory

Set Contact4Oref=##class(MyPackage.Contacts).%OpenId(4) ;bring object Contact4

;into memory

Do ActorOref.MyContacts.Insert(Contact1Oref) ;associate Contact1 with

;Actor Johnny Depp

Do ActorOref.MyContacts.Insert(Contact2Oref) ;associate Contact2 with  
 ;Actor Johnny Depp

Do ActorOref.MyContacts.Insert(Contact3Oref) ;associate Contact3 with  
 ;Actor Johnny Depp

Do ActorOref.MyContacts.Insert(Contact4Oref) ;associate Contact4 woith

;Actor Johnny Depp

Write !,"Count: ",ActorOref.MyContacts.Count()

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 4

1

Example ‑ Global generated from Class MyPackage.Actor

zw ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne","",$lb("","","",""),"Blue",$lb("BlueShirt","CyanShirt","RedShirt",""WhiteShirt"))

^MyPackage.ActorD(2)=$lb("","Jodie Foster","",$lb("","","",""),"Green","")

^MyPackage.ActorD(2,"MyHats",1)="Bowler"

^MyPackage.ActorD(2,"MyHats",2)="BigBirdHat"

^MyPackage.ActorD(2,"MyHats","Top")="TopHat"

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2",$lb("","","",""),"Cyan")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.",

"Marlboro","MA","01752"),"Brown")

^MyPackage.ActorD(5)=$lb("","Johnny Depp","",$lb("","","",""),"Tan","",$lb($lb("1"),$lb("2"),$lb("3"),$lb("4")))

^MyPackage.ActorD(6)=$lb("","Carol Burnett","",$lb("","","",""),"Red")

^MyPackage.ActorD(7)=$lb("","Will Smith","",$lb("","","",""),"Navy")

^MyPackage.ActorD(8)=$lb("","Ann Margaret","",$lb("","","",""),"Yellow")

^MyPackage.ActorD(9)=$lb("","Dean Martin","",$lb("","","",""),"Green")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","",$lb("","","",""),"Black")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart","",$lb("","","",""),"Brown")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2",$lb("","","",""),"Blue")

ZW ^MyPackage.ContactsD

^MyPackage.ContactsD=6

^MyPackage.ContactsD(1)=$lb("","Contact1")

^MyPackage.ContactsD(2)=$lb("","Contact2")

^MyPackage.ContactsD(3)=$lb("","Contact3")

^MyPackage.ContactsD(4)=$lb("","Contact4")

^MyPackage.ContactsD(5)=$lb("","Contact5")

^MyPackage.ContactsD(6)=$lb("","Contact6")

Example ‑ GetAt Method – Returns the Element associated with a Key

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

GetAt Method - Returns the Element associated with a Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

For Key=1:1:ActorOref.MyContacts.Count() { ;Display each element  
   Write !,"Key: ",Key ;of Collection List  
   Write " Name ",ActorOref.MyContacts.GetAt(Key).ContactName ;Property: MyContacts  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 1 - Contact1

Key: 2 - Contact2

Key: 3 - Contact3

Key: 4 – Contact4

Example ‑ Find Method – Finds the associated Key for a String

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

Find Code – Finds the associated Key for a String

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Set NameToFind="Contact3" ;Name to Find

Set FoundKey="" ;initialized FoundKey

For Key=1:1:ActorOref.MyContacts.Count() {

If NameToFind=ActorOref.MyContacts.GetAt(Key).ContactName Set FoundKey=Key Quit

}

If FoundKey’="" {

Write !, "Found: ", NameToFind," at Key: ",FoundKey

}

Else {

Write !, "Could not find: ", NameToFind

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Found: Contact3 at Key: 3

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Set NameToFind="Contact5" ;Name to Find

Set FoundKey="" ;initialized FoundKey

For Key=1:1:ActorOref.MyContacts.Count() {

If NameToFind=ActorOref.MyContacts.GetAt(Key).ContactName Set FoundKey=Key Quit

}

If FoundKey’="" {

Write !, "Found: ", NameToFind," at Key: ",FoundKey

}

Else {

Write !, "Could not find: ", NameToFind

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Could not find: Contact5

Table ‑ Traversing Methods

|  |  |  |
| --- | --- | --- |
| Method |  | How Key or Slot is passed |
| Next | Traversing Forward | Key or Slot passed by Value |
| Previous | Traversing Backward | Key or Slot Passed by Value |
| GetNext | Traversing Forward | Key or Slot Passed by Reference |
| GetPrevious | Traversing Backward | Key or Slot Passed by Reference |

Example ‑ Next Method – Returns the Element for the next Key

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

Next Method - Returns the Element for the next Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyContacts.Next(Key) ;get the next Key  
   If Key'="" {  
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyContacts.GetAt(Key).ContactName ;display the contact

   }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 1 - Contact1

Key: 2 - Contact2

Key: 3 - Contact3

Key: 4 - Contact4

Example ‑ Previous Method – Returns the Element for the previous Key

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

Previous Method - Returns the Element for the previous Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyContacts.Previous(Key) ;get the previous Key  
   If Key'="" {  
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyContacts.GetAt(Key).ContactName ;display the contact

   }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 4 - Contact4

Key: 3 - Contact3

Key: 2 - Contact2

Key: 1 - Contact1

Example ‑ GetNext Method – Returns the Element for the next Key

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

GetNext Method - Returns the Element for the next Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

 Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp   
 ;into memory  
  
 Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements  
 Set Key = "" Do { ;start with beginning Key  
    Set Contact=ActorOref.MyContacts.GetNext(.Key) ;get next Key  
    If Key'="" Write !,"Key: ",Key," ",Contact.ContactName  ;display Key with Contact  
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 1 - Contact1

Key: 2 - Contact2

Key: 3 - Contact3

Key: 4 - Contact4

Example ‑ GetPrevious Method – Returns the Element for the Previous Key

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

GetPrevious Method - Returns the Element for the previous Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

Set Key = "" Do { ;start with beginning Key  
 Set Contact=ActorOref.MyContacts.GetPrevious(.Key) ;get previous Key  
 If Key'="" Write !,"Key: ",Key," ",Contact.ContactName ;display Key with Contact  
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 4 - Contact4

Key: 3 - Contact3

Key: 2 - Contact2

Key: 1 - Contact1

Example ‑ InsertAt Method – Insert a new Contact between the 1st and 2nd Contact

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

InsertAt Method - Insert an Oref into a Collection at a specific key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Set Contact5Oref=##class(MyPackage.Contacts).%OpenId(5) ;bring object

;Contact 5 into memory

Do ActorOref.MyContacts.InsertAt(Contact5Oref,2) ;Insert Contact5 into the

;2nd Key pushing the other

;contacts out

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

For Key=1:1:ActorOref.MyContacts.Count() { ;display each element  
    Write !,"Key: ",Key ;of collection List  
    Write " - ",ActorOref.MyContacts.GetAt(Key).ContactName ;Property: MyContacts  
 }

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 5

Key: 1 - Contact1

Key: 2 - Contact5

Key: 3 - Contact2

Key: 4 - Contact3

Key: 5 - Contact4

1

Example ‑ SetAt Method – Set or Replace a specific Contact

The MyContacts Property is defined

as a Collection List of References to Persistent Objects

SetAt Method - Set or Replace an Element at a specific Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Set Contact6Oref=##class(MyPackage.Contacts).%OpenId(6) ;bring object into

;memory – Contact6

Do ActorOref.MyContacts.SetAt(Contact6Oref,2) ;Replace contact with

;Contact6 at the 2nd

;position

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

For Key=1:1:ActorOref.MyContacts.Count() { ;display each element  
   Write !,"Key: ",Key ;of Collection List  
   Write " - ",ActorOref.MyContacts.GetAt(Key).ContactName ;Property: MyContacts  
 }

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 5

Key: 1 - Contact1

Key: 2 – Contact6

Key: 3 - Contact2

Key: 4 - Contact3

Key: 5 - Contact4

1

Example ‑ RemoveAt Method – Remove a specific Contact

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

RemoveAt Method - Remove the Element associated with a Key

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ActorOref=##class(MyPackage.Actor).%OpenId(5) ;bring object Johnny Depp

;into memory

Do ActorOref.MyContacts.RemoveAt(3) ;remove contact at

;Key 3, Contact2

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

For Id=1:1:ActorOref.MyContacts.Count() { ;display each element  
    Write !,"Id: ",Id ;of Collection List  
   Write " - ",ActorOref.MyContacts.GetAt(Id).ContactName ;Property: MyContacts  
 }

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Id: 1 - Contact1

Id: 2 - Contact6

Id: 3 - Contact3

Id: 4 - Contact4

1

Example ‑ Display MyContacts Data using Embedded SQL

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

New id, actorname, mycontacts  
 &sql(Declare MyCursor CURSOR FOR  
  SELECT Id, Name, MyContacts  
  INTO :id, :actorname, :mycontacts  
  FROM MyPackage.Actor   
  ORDER BY Id)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !, "Id: ",id  
 Write " Name: ",actorname  
 Set ActorOref=##class(MyPackage.Actor).%OpenId(id)   
 Set Key = "" Do {

        Set Key=ActorOref.MyContacts.Next(Key) ;get the next Key  
        If Key'="" {  
            Write !,?12,"Key: ",Key  
            Write " - ",ActorOref.MyContacts.GetAt(Key).ContactName   
        }   
    } While (Key '= "")  
 &sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 Name: John Wayne

Id: 2 Name: Jodie Foster

Id: 3 Name: Clint Eastwood

Id: 4 Name: Julie Andrews

Id: 5 Name: Johnny Depp

Key: 1 - Contact1

Key: 2 – Contact6

Key: 3 - Contact3

Key: 4 - Contact4

Id: 6 Name: Carol Burnett

Id: 7 Name: Will Smith

Id: 8 Name: Ann Margaret

Id: 9 Name: Dean Martin

Id: 10 Name: Ally Sheedy

Id: 11 Name: Humphrey Bogart

Id: 12 Name: Katharine Hepburn

Example ‑ Display MyContacts Data using Dynamic SQL

The MyContacts Property is defined as a

Collection List of References to Persistent Objects

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set MyQuery="SELECT Id, Name FROM MyPackage.Actor"  
   
 Set ResultSet=##class(%ResultSet).%New("%DynamicQuery:SQL")  
  
 Set SC=ResultSet.Prepare(MyQuery)  
  
 Set SC=ResultSet.Execute()  
  
 While ResultSet.Next() {  
  Set Id=ResultSet.Data("ID")  
     Write !, "Id:",Id  
     Write "Name: ",ResultSet.Data("Name")  
  Set ActorOref=##class(MyPackage.Actor).%OpenId(Id)   
  Set Key = "" Do {   
        Set Key=ActorOref.MyContacts.Next(Key) ;get the next Key  
        If Key'="" {  
            Write !,?7,"Key: ",Key ;display the Key number  
            Write " Name ",ActorOref.MyContacts.GetAt(Key).ContactName   
       }   
    } While (Key '= "")  
  }  
  Set SC=ResultSet.Close()

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 Name John Wayne

Id: 2 Name Jodie Foster

Id: 3 Name Clint Eastwood

Id: 4 Name Julie Andrews

Id: 5 Name Johnny Depp

Key: 1 - Contact1

Key: 2 – Contact6

Key: 3 - Contact3

Key: 4 - Contact4

Id: 6 Name: Carol Burnett

Id: 7 Name: Will Smith

Id: 8 Name: Ann Margaret

Id: 9 Name: Dean Martin

Id: 10 Name: Ally Sheedy

Id: 11 Name: Humphrey Bogart

Id: 12 Name: Katharine Hepburn

*“It is not hard, when you don't know what you are talking about.”*

# Class & Object Properties - Collection Array of References to Persistent Objects

Table ‑ Object Properties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object Property | Data Type | Reference  Object Class  and Name | Name we chose | Special Considerations |
| Data Types | %String |  | Name  (Name of Actor) |  |
| Collection List of Data Types | %String |  | MyShirts  (Shirts belonging to an Actor) | Total length cannot exceed 32k |
| Collection Array of Data Types | %String |  | MyHats  (Hats belonging to an Actor) | No length limit  In SQL must be handled as a Child table  Must specify a Key when accessing |
| Collection List of References to Persistent Objects |  | Persistent Objects | MyContacts  (Contacts of the Actor) |  |
| Collection Array of References to Persistent Objects |  | Persistent Objects | MyClients  (Clients of the Actor) |  |

Example ‑ Class Definition – Creating the Clients Class

Class MyPackage.Clients Extends %Persistent  
{  
  
Property ClientName As %String [ Required ];  
  
Index ClientNameIndex On ClientName;  
  
}

Example ‑ Populate the Clients Class

For Name="Client1","Client2","Client3" Do CreateObject(Name)

For Name="Client4","Client5","Client6" Do CreateObject(Name)

Quit

CreateObject(Name) [] Public {  
   Set ActorOref=##class(MyPackage.Clients).%New() ;create new object reference  
   Set ActorOref.ClientName=Name ;populate object with a name   
   Do ActorOref.%Save() ;save the object  
   Set NewId=ActorOref.%Id() Write !,"Id: ",NewId ;write ID of new Object  
   Write " – ",ActorOref.ClientName ;write name of the new Object  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 – Client1

Id: 2 – Client2

Id: 3 – Client3

Id: 4 – Client4

Id: 5 – Client5

Id: 6 – Client6

Example ‑ Actor Class Redefinition – add Reference Property that will point to the Clients Class

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;  
  
Property MyHome As Address;  
  
Property MyShirts As list Of %String;  
  
Property MyHats As array Of %String;  
  
Property MyContacts As list Of Contacts;  
  
Property MyClients As array Of Clients;  
}

Example ‑ Clear and Count Methods - Deletes all elements and show the count

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

Clear Method – Clears or Deletes the Collection of Elements

Count Method - Returns the Number of Elements in the Collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

;  
 Do ActorOref.MyClients.Clear() ;clear Array of Clients  
 Write !,"Count: ",ActorOref.MyClients.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 0

1

Example ‑ SetAt Method – SetAt four Clients into the Collection

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

SetAt Method - Set an Oref at the specific Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Set Client1Oref=##class(MyPackage.Clients).%OpenId(1) ;bring object Client1

;into memory

Set Client2Oref=##class(MyPackage.Clients).%OpenId(2) ;bring object Client2

;into memory

Set Client3Oref=##class(MyPackage.Clients).%OpenId(3) ;bring object Client3

;into memory

Set Client4Oref=##class(MyPackage.Clients).%OpenId(4) ;bring object Client4

;into memory

Do ActorOref.MyClients.SetAt(Client1Oref,1) ;associate Client1 with

;Actress Carol Burnett at key 1

Do ActorOref.MyClients.SetAt(Client2Oref,2) ;associate Client2 with  
 ;Actress Carol Burnett at key 2

Do ActorOref.MyClients.SetAt(Client3Oref,3) ;associate Client3 with  
 ;Actress Carol Burnett at key 3

Do ActorOref.MyClients.SetAt(Client4Oref,4) ;associate Client4 with

;Actress Carol Burnett at key 4

Write !,"Count: ",ActorOref.MyClients.Count()

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 4

1

Example ‑ Global generated from Class MyPackage.Actor

ZW ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne","",$lb("","","",""),"Blue",$lb("BlueShirt","CyanShirt","RedShirt",""WhiteShirt"))

^MyPackage.ActorD(2)=$lb("","Jodie Foster","",$lb("","","",""),"Green","")

^MyPackage.ActorD(2,"MyHats",1)="Bowler"

^MyPackage.ActorD(2,"MyHats",2)="BigBirdHat"

^MyPackage.ActorD(2,"MyHats","Top")="TopHat"

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2",$lb("","","",""),"Cyan")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.","Marlboro",

"MA","01752"),"Brown")

^MyPackage.ActorD(5)=$lb("","Johnny Depp","",$lb("","","",""),"Tan","",$lb($lb("1"),$lb("6"),$lb("3"),$lb("4")))

^MyPackage.ActorD(6)=$lb("","Carol Burnett","",$lb("","","",""),"Red","","")

^MyPackage.ActorD(6,"MyClients",1)=1

^MyPackage.ActorD(6,"MyClients",2)=2

^MyPackage.ActorD(6,"MyClients",3)=3

^MyPackage.ActorD(6,"MyClients",4)=4

^MyPackage.ActorD(7)=$lb("","Will Smith","",$lb("","","",""),"Navy")

^MyPackage.ActorD(8)=$lb("","Ann Margaret","",$lb("","","",""),"Yellow")

^MyPackage.ActorD(9)=$lb("","Dean Martin","",$lb("","","",""),"Green")

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","",$lb("","","",""),"Black")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart","",$lb("","","",""),"Brown")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2",$lb("","","",""),"Blue")

ZW ^MyPackage.ClientsD

^MyPackage.ClientsD=6

^MyPackage.ClientsD(1)=$lb("","Client1")

^MyPackage.ClientsD(2)=$lb("","Client2")

^MyPackage.ClientsD(3)=$lb("","Client3")

^MyPackage.ClientsD(4)=$lb("","Client4")

^MyPackage.ClientsD(5)=$lb("","Client5")

^MyPackage.ClientsD(6)=$lb("","Client6")

Example ‑ IsDefined Method – See if a Key is defined

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

IsDefined Method - Returns a 1 if the Key is defined otherwise 0

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Write ActorOref.MyClients.IsDefined(1) ;Is key 1 defined? - Yes  
1

Write ActorOref.MyClients.IsDefined(3) ;Is key 3 defined? - Yes  
1

Write ActorOref.MyClients.IsDefined(6) ;Is key 6 defined? - No  
0

In

Example 28‑7 we see a demonstration of the IsDefined Method that determines if a *key* exists.

Example ‑ GetAt Method – Returns the Elements associated with the Key

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

GetAt Method - Returns the Element associated with a Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Write !,"Count: ",ActorOref.MyClients.Count() ;count of elements

For Id=1:1:ActorOref.MyClients.Count() { ;Display each element  
   Write !,"Id: ",Id ;of Collection Array  
   Write " - ",ActorOref.MyClients.GetAt(Id).ClientName ;Property: MyClients  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Id: 1 - Client1

Id: 2 - Client2

Id: 3 - Client3

Id: 4 – Client4

Example ‑ Find Method – Finds the associated ID for a String

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

Find Code – Finds the associated Id for a String

 Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett   
 ;into memory  
   
 Set NameToFind="Client3" ;Name to Find  
 Set FoundId="" ;initialized FoundId  
  
 For Id=1:1:ActorOref.MyClients.Count() {  
    If ActorOref.MyClients.IsDefined(Id) { ;Does the Id exist?  
       If NameToFind= ActorOref.MyClients.GetAt(Id).ClientName Set FoundId=Id Quit  
   }  
 }

 If FoundId’="" {

Write !, "Found: ", NameToFind," at Id: ",FoundId

}  
 Else{

Write !, "Could not find: ", NameToFind

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Found: Client3 at Id: 3

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

 Set NameToFind="Client5" ;Name to Find  
 Set FoundId="" ;initialized FoundId  
  
 For Id=1:1:ActorOref.MyClients.Count() {  
    If ActorOref.MyClients.IsDefined(Id) { ;Does the Id exist?  
       If NameToFind= ActorOref.MyClients.GetAt(Id).ClientName Set FoundId=Id Quit

    }  
 }

 If FoundId’="" {

Write !, "Found: ", NameToFind," at Id: ",FoundId

}  
 Else{

Write !, "Could not find: ", NameToFind

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Could not find: Client5

Table ‑ Traversing Methods

|  |  |  |
| --- | --- | --- |
| Method |  | How Key or Slot is passed |
| Next | Traversing Forward | Key or Slot passed by Value |
| Previous | Traversing Backward | Key or Slot Passed by Value |
| GetNext | Traversing Forward | Key or Slot Passed by Reference |
| GetPrevious | Traversing Backward | Key or Slot Passed by Reference |

Example ‑ Next Method – Returns the Element for the next Key

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

Next Method - Returns the Element for the next Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Write !,"Count: ",ActorOref.MyClients.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyClients.Next(Key) ;get the next Key  
   If Key'="" {   
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyClients.GetAt(Key).ClientName ;display the Client

   }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 1 - Client1

Key: 2 - Client2

Key: 3 - Client3

Key: 4 - Client4

Example ‑ Previous Method – Returns the Element for the previous Key

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

Previous Method - Returns the Element for the previous Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Write !,"Count: ",ActorOref.MyClients.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyClients.Previous(Key) ;get the previous Key  
   If Key'="" {  
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyClients.GetAt(Key).ClientName ;display the Client

   }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 4 - Client4

Key: 3 - Client3

Key: 2 - Client2

Key: 1 - Client1

Example ‑ GetNext Method – Returns the Element for the next Key

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

GetNext Method - Returns the Element for the next Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Write !,"Count: ",ActorOref.MyClients.Count() ;count of elements

Set Key = "" Do { ;start with beginning Key  
 Set Client=ActorOref.MyClients.GetNext(.Key) ;get next Key  
 If Key'="" Write !,"Key: ",Key," ",Client.ClientName ;display Key with Client  
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 1 - Client1

Key: 2 - Client2

Key: 3 - Client3

Key: 4 - Client4

Example ‑ GetPrevious Method – Returns the Element for the Previous Key

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

GetPrevious Method - Returns the Element for the previous Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Write !,"Count: ",ActorOref.MyContacts.Count() ;count of elements

Set Key = "" Do { ;start with beginning Key  
 Set Client=ActorOref.MyClients.GetPrevious(.Key) ;get previous Key  
 If Key'="" Write !,"Key: ",Key," ",Client.ClientName ;display Key with Client  
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Key: 4 - Client4

Key: 3 - Client3

Key: 2 - Client2

Key: 1 - Client1

Example ‑ SetAt Method – Replace a specific Client

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

SetAt Method - Set an Oref at the specific Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Set Client6Oref=##class(MyPackage.Clients).%OpenId(6) ;bring object into

;memory – Client6

Do ActorOref.MyClients.SetAt(Client6Oref,2) ;Replace Client with

;Client6 at the 2nd

;position

Write !,"Count: ",ActorOref.MyClients.Count() ;count of elements

For Id=1:1:ActorOref.MyClients.Count() { ;Display each element  
   Write !,"Id: ",Id ;of Collection Array  
   Write " - ",ActorOref.MyClients.GetAt(Id).ClientName ;Property: MyClients  
 }

Write !,ActorOref.%Save() ;Save the Object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 5

Id: 1 - Client1

Id: 2 – Client6

Id: 3 - Client3

Id: 4 - Client4

1

Example ‑ RemoveAt Method – Remove a specific Client

The MyClients Property is defined as a

Collection Array of References to Persistent Objects

RemoveAt Method - Remove the Element associated with the Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(6) ;bring object Carol Burnett

;into memory

Do ActorOref.MyClients.RemoveAt(3) ;Remove Client at

;Key 3, Client3

 Write !,"Count: ",ActorOref.MyClients.Count() ;count of elements  
 Set Key = "" Do { ;start with beginning Key  
    Set Client=ActorOref.MyClients.GetNext(.Key) ;get next Key  
    If Key'="" Write !,"Key: ",Key," ",Client.ClientName ;display Key with Client  
 } While (Key '= "")

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Id: 1 - Client1

Id: 2 - Client6

Id: 4 - Client4

1

Example ‑ Display MyClients Data using Embedded SQL

 New actorname,myclients  
 &sql(Declare MyCursor CURSOR FOR  
  SELECT Actor->Name, MyClients  
  INTO :actorname, :myclients  
  FROM MyPackage.Actor\_MyClients   
  ORDER BY Name)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !,"Name: ",actorname  
  Write ?25,"MyClients: ",myclients  
 &sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Name: Carol Burnett MyClients: 1

Name: Carol Burnett MyClients: 6

Name: Carol Burnett MyClients: 4

Example ‑ Display MyClients Data using Dynamic SQL

Set MyQuery="SELECT Actor->Name,MyClients FROM MyPackage.Actor\_MyClients"  
   
 Set ResultSet=##class(%ResultSet).%New("%DynamicQuery:SQL")  
 ;Create a new Instance of %ResultSet  
  
 Set SC=ResultSet.Prepare(MyQuery) ;Prepare the Query  
  
 Set SC=ResultSet.Execute() ;Execute the Query  
  
 While ResultSet.Next() { ;Process the Query results  
     Write !,ResultSet.Data("Name")," - "  
     Write "Client: ",ResultSet.Data("MyClients")  
  }  
  
 Set SC=ResultSet.Close() ;Close the Query

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Carol Burnett - Client: 1

Carol Burnett - Client: 6

Carol Burnett - Client: 4

*“Let another praise you, and not your own mouth; a stranger, and not your own lips.” - Proverbs 27:2*

# Class & Object Properties - Collection List of References to Embedded Objects

Table ‑ Object Properties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object Property | Data Type | Reference  Object Class  and Name | Name we chose | Special Considerations |
| Data Types | %String |  | Name  (Name of Actor) |  |
| Collection List of Data Types | %String |  | MyShirts  (Shirts belonging to an Actor) | Total length cannot exceed 32k |
| Collection Array of Data Types | %String |  | MyHats  (Hats belonging to an Actor) | No length limit  In SQL must be handled as a Child table  Must specify a Key when accessing |
| Collection List of References to Persistent Objects |  | Persistent Objects | MyContacts  (Contacts of the Actor) |  |
| Collection Array of References to Persistent Objects |  | Persistent Objects | MyClients  (Clients of the Actor) |  |
| Collection List of References to Embedded Objects |  | Embedded Objects  %Serial Class | MyRentals  (Rental Property  of the Actor) |  |
| Collection Array of References to Embedded Objects |  | Embedded Objects  %Serial Class | MyPets  (Pets of the Actor) |  |

Example ‑ Class Definition – Creating the Rentals Class

Class MyPackage.Rentals Extends %SerialObject  
{  
  
Property Street As %String(MAXLEN = 80);  
  
Property City As %String(MAXLEN = 30);  
  
Property State As %String(MAXLEN = 2);  
  
Property Zip As %String(MAXLEN = 10);  
  
}

Example ‑ Actor Class Redefinition – add Reference Property that will point to the Rentals Class

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;  
  
Property MyHome As Address;  
  
Property MyShirts As list Of %String;  
  
Property MyHats As array Of %String;  
  
Property MyContacts As list Of Contacts;  
  
Property MyClients As array Of Clients;  
  
Property MyRentals As list Of Rentals;  
}

Example ‑ Clear and Count Methods - Deletes all elements and show the count

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

Clear Method – Clears or Deletes the Collection of Elements

Count Method - Returns the Number of Elements in the Collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

;  
 Do ActorOref.MyRentals.Clear() ;clear list of Rentals  
 Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you run the above code from the Terminal, you should get the following output.

Count: 0

1

Example ‑ Insert Method – Insert three Rentals into the Collection

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

Insert Method - Inserts an Oref at the end of the collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Set AddressOref1=##class(MyPackage.Rentals).%New() ;create new Address Oref1

Set AddressOref1.Street="123 Main St." ;Set the Street

Set AddressOref1.City="Pittsburgh" ;Set the City

Set AddressOref1.State="PA" ;Set the State

Set AddressOref1.Zip="01600" ;Set the Zip

Set AddressOref2=##class(MyPackage.Rentals).%New() ;create new Address Oref2

Set AddressOref2.Street="53 Elm St." ;Set the Street

Set AddressOref2.City="L.A." ;Set the City

Set AddressOref2.State="CA" ;Set the State

Set AddressOref2.Zip="95602" ;Set the Zip

Set AddressOref3=##class(MyPackage.Rentals).%New() ;create new Address Oref3

Set AddressOref3.Street="9 Pershing St." ;Set the Street

Set AddressOref3.City="Worcester" ;Set the City

Set AddressOref3.State="MA" ;Set the State

Set AddressOref3.Zip="01752" ;Set the Zip

Do ActorOref.MyRentals.Insert(AddressOref1) ;associate AddressOref1 with Actor

Do ActorOref.MyRentals.Insert(AddressOref2) ;associate AddressOref2 with Actor

Do ActorOref.MyRentals.Insert(AddressOref3) ;associate AddressOref3 with Actor

Write !,ActorOref.%Save() ;Save the object

Example ‑ Global generated from Class MyPackage.Actor

ZW ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne","",$lb("","","",""),"Blue",$lb("BlueShirt","CyanShirt","RedShirt","WhiteShirt"))

^MyPackage.ActorD(2)=$lb("","Jodie Foster","",$lb("","","",""),"Green","")

^MyPackage.ActorD(2,"MyHats",1)="Bowler"

^MyPackage.ActorD(2,"MyHats",2)="BigBirdHat"

^MyPackage.ActorD(2,"MyHats","Top")="TopHat"

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2",$lb("","","",""),"Cyan")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.","Marlboro",

"MA","01752"),"Brown")

^MyPackage.ActorD(5)=$lb("","Johnny Depp","",$lb("","","",""),"Tan","",$lb($lb("1"),$lb("6"),$lb("3"),$lb("4")))

^MyPackage.ActorD(6)=$lb("","Carol Burnett","",$lb("","","",""),"Red","","")

^MyPackage.ActorD(6,"MyClients",1)=1

^MyPackage.ActorD(6,"MyClients",2)=6

^MyPackage.ActorD(6,"MyClients",4)=4

^MyPackage.ActorD(7)=$lb("","Will Smith","",$lb("","","",""),"Navy")

^MyPackage.ActorD(8)=$lb("","Ann Margaret","",$lb("","","",""),"Yellow")

^MyPackage.ActorD(9)=$lb("","Dean Martin","",$lb("","","",""),"Green","","",

$lb($lb($lb("123 Main St.","Pittsburgh","PA","01600")),$lb($lb("53 Elm St.","L.A.","CA","95602")),$lb($lb("9 Pershing St.","Worcester","MA","01752"))))

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","",$lb("","","",""),"Black")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart","",$lb("","","",""),"Brown")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2",$lb("","","",""),"Blue")

Example ‑ GetAt Method – Returns the Elements associated with the Key

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

GetAt Method - Returns the Element associated with a Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

For Key=1:1:ActorOref.MyRentals.Count() { ;Display each element  
   Write !,"Key: ",Key ;of Collection List  
   Write " - ",ActorOref.MyRentals.GetAt(Key).Street ;Property: MyRentals – Street

   Write ?30,ActorOref.MyRentals.GetAt(Key).City ;Property: MyRentals - City  
   Write ?50,ActorOref.MyRentals.GetAt(Key).State ;Property: MyRentals - State  
   Write ?60,ActorOref.MyRentals.GetAt(Key).Zip ;Property: MyRentals - Zip  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Key: 1 - 123 Main St. Pittsburgh PA 01600

Key: 2 - 53 Elm St. L.A. CA 95602

Key: 3 - 9 Pershing St. Worcester MA 01752

Example ‑ Find Method – Finds the associated KEY for a String

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

Find Code – Finds the associated Key for a String

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Set StreetToFind="53 Elm St." ;Street to Find

Set FoundKey="" ;initialized FoundKey

For Key=1:1:ActorOref.MyRentals.Count() {

    If StreetToFind=ActorOref.MyRentals.GetAt(Key).Street Set FoundKey=Key

}

If FoundKey’="" {

Write !, "Found: ", StreetToFind," at Key: ",FoundKey

}

Else{

Write !, "Could not find: ", StreetToFind

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Found: 53 Elm St. at Key: 2

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set CityToFind="Worcester" ;City to Find

Set FoundKey="" ;initialized FoundKey

For Key=1:1:ActorOref.MyRentals.Count() {

   If CityToFind=ActorOref.MyRentals.GetAt(Key).City Set FoundKey=Key

}

If FoundKey’="" {

Write !, "Found: ", CityToFind," at Key: ",FoundKey

}

Else{

Write !, "Could not find: ", CityToFind

}

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Found: Worcester at Key: 3

Table ‑ Traversing Methods

|  |  |  |
| --- | --- | --- |
| Method |  | How Key or Slot is passed |
| Next | Traversing Forward | Key or Slot passed by Value |
| Previous | Traversing Backward | Key or Slot Passed by Value |
| GetNext | Traversing Forward | Key or Slot Passed by Reference |
| GetPrevious | Traversing Backward | Key or Slot Passed by Reference |

Example ‑ Next Method – Returns the Element for the next Key

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

Next Method - Returns the Element for the next Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyRentals.Next(Key) ;get the next Key  
   If Key'="" {  
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyRentals.GetAt(Key).Street ;Property: MyRentals – Street

Write ?30,ActorOref.MyRentals.GetAt(Key).City ;Property: MyRentals - City  
 Write ?50,ActorOref.MyRentals.GetAt(Key).State ;Property: MyRentals - State  
 Write ?60,ActorOref.MyRentals.GetAt(Key).Zip ;Property: MyRentals - Zip  
   }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Key: 1 - 123 Main St. Pittsburgh PA 01600

Key: 2 - 53 Elm St. L.A. CA 95602

Key: 3 - 9 Pershing St. Worcester MA 01752

Example ‑ Previous Method – Returns the Element for the previous Key

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

Previous Method - Returns the Element for the previous Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyRentals.Previous(Key) ;get the previous Key  
   If Key'="" {  
 Write !,"Key: ",Key ;display the Key number  
 Write " - ",ActorOref.MyRentals.GetAt(Key).Street ;Property: MyRentals – Street

Write ?30,ActorOref.MyRentals.GetAt(Key).City ;Property: MyRentals - City  
 Write ?50,ActorOref.MyRentals.GetAt(Key).State ;Property: MyRentals - State  
 Write ?60,ActorOref.MyRentals.GetAt(Key).Zip ;Property: MyRentals - Zip  
   }   
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Key: 3 - 9 Pershing St. Worcester MA 01752

Key: 2 - 53 Elm St. L.A. CA 95602

Key: 1 - 123 Main St. Pittsburgh PA 01600

Example ‑ GetNext Method – Returns the Element for the next Key

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

GetNext Method - Returns the Element for the next Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

Set Key = "" Do { ;start with beginning Key  
 Set Contact=ActorOref.MyRentals.GetNext(.Key) ;get next Key  
 If Key'="" {

Write !,"Key: ",Key," " ;display Key

Write " - ",ActorOref.MyRentals.GetAt(Key).Street ;Property: MyRentals – Street

Write ?30,ActorOref.MyRentals.GetAt(Key).City ;Property: MyRentals - City  
 Write ?50,ActorOref.MyRentals.GetAt(Key).State ;Property: MyRentals - State  
 Write ?60,ActorOref.MyRentals.GetAt(Key).Zip ;Property: MyRentals - Zip  
 }  
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Key: 1 - 123 Main St. Pittsburgh PA 01600

Key: 2 - 53 Elm St. L.A. CA 95602

Key: 3 - 9 Pershing St. Worcester MA 01752

Example ‑ GetPrevious Method – Returns the Element for the Previous Key

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

GetPrevious Method - Returns the Element for the previous Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

Set Key = "" Do {   
 Set Contact=ActorOref.MyRentals.GetPrevious(.Key) ;get next Key  
 If Key'="" {

Write !,"Key: ",Key," " ;display Key

Write " - ",ActorOref.MyRentals.GetAt(Key).Street ;Property: MyRentals – Street

Write ?30,ActorOref.MyRentals.GetAt(Key).City ;Property: MyRentals - City  
 Write ?50,ActorOref.MyRentals.GetAt(Key).State ;Property: MyRentals - State  
 Write ?60,ActorOref.MyRentals.GetAt(Key).Zip ;Property: MyRentals - Zip  
 }  
 } While (Key '= "")

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Key: 3 - 9 Pershing St. Worcester MA 01752

Key: 2 - 53 Elm St. L.A. CA 95602

Key: 1 - 123 Main St. Pittsburgh PA 01600

Example ‑ InsertAt Method – Insert a new Address between the 1st and 2nd Address

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

InsetAt Method - Insert an Oref into a Collection at the specific key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Set AddressOref=##class(MyPackage.Rentals).%New() ;create new Address ActorOref

Set AddressOref.Street="39 Pinewold Ave" ;Set the Street

Set AddressOref.City="Burlington" ;Set the City

Set AddressOref.State="MA" ;Set the State

Set AddressOref.Zip="01803" ;Set the Zip

Do ActorOref.MyRentals.InsertAt(AddressOref,2) ;Insert AddressOref

;at Key position 2

Do ActorOref.%Save() ;save the object

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

For Id=1:1:ActorOref.MyRentals.Count() { ;Display each element  
   Write !,"Id: ",Id ;of Collection List  
   Write " - ",ActorOref.MyRentals.GetAt(Id).Street ;Property: MyRentals – Street

   Write ?30,ActorOref.MyRentals.GetAt(Id).City ;Property: MyRentals - City  
   Write ?50,ActorOref.MyRentals.GetAt(Id).State ;Property: MyRentals - State  
   Write ?60,ActorOref.MyRentals.GetAt(Id).Zip ;Property: MyRentals - Zip  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Id: 1 - 123 Main St. Pittsburgh PA 01600

Id: 2 - 39 Pinewold Ave Burlington MA 01803

Id: 3 - 53 Elm St. L.A. CA 95602

Id: 4 - 9 Pershing St. Worcester MA 01752

Example ‑ SetAt Method – Replace a specific Address

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

SetAt Method – Set an Oref at the specified Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Set AddressOref=##class(MyPackage.Rentals).%New() ;create new ActorOref

Set AddressOref.Street="1040 Lincoln St." ;Set the Street

Set AddressOref.City="Dallas" ;Set the City

Set AddressOref.State="TX" ;Set the State

Set AddressOref.Zip="00000" ;Set the Zip

Do ActorOref.MyRentals.SetAt(AddressOref,2) ;Replace AddressOref at Key 2

Do ActorOref.%Save() ;save the object

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

For Id=1:1:ActorOref.MyRentals.Count() { ;Display each element  
   Write !,"Id: ",Id ;of Collection List  
   Write " - ",ActorOref.MyRentals.GetAt(Id).Street ;Property: MyRentals – Street

   Write ?30,ActorOref.MyRentals.GetAt(Id).City ;Property: MyRentals - City  
   Write ?50,ActorOref.MyRentals.GetAt(Id).State ;Property: MyRentals - State  
   Write ?60,ActorOref.MyRentals.GetAt(Id).Zip ;Property: MyRentals - Zip  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 4

Id: 1 - 123 Main St. Pittsburgh PA 01600

Id: 2 - 1040 Lincoln St. Dallas TX 00000

Id: 3 - 53 Elm St. L.A. CA 95602

Id: 4 - 9 Pershing St. Worcester MA 01752

Example ‑ RemoveAt Method – Remove a specific Address

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

RemoveAt Method - Remove the Element associated with the Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(9) ;bring object Dean Martin

;into memory

Do ActorOref.MyRentals.RemoveAt(2) ;Remove contact at

;Key 2

Do ActorOref.%Save() ;save the object

Write !,"Count: ",ActorOref.MyRentals.Count() ;count of elements

For Id=1:1:ActorOref.MyRentals.Count() { ;Display each element  
   Write !,"Id: ",Id ;of Collection List  
   Write " - ",ActorOref.MyRentals.GetAt(Id).Street ;Property: MyRentals – Street

   Write ?30,ActorOref.MyRentals.GetAt(Id).City ;Property: MyRentals - City  
   Write ?50,ActorOref.MyRentals.GetAt(Id).State ;Property: MyRentals - State  
   Write ?60,ActorOref.MyRentals.GetAt(Id).Zip ;Property: MyRentals - Zip  
 }

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Count: 3

Id: 1 - 123 Main St. Pittsburgh PA 01600

Id: 2 - 53 Elm St. L.A. CA 95602

Id: 3 - 9 Pershing St. Worcester MA 01752

Example ‑ Display MyRentals Data using Embedded SQL

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

New id, actorname, myrentals  
 &sql(Declare MyCursor CURSOR FOR  
  SELECT Id, Name, MyRentals  
  INTO :id, :actorname, :myrentals  
  FROM MyPackage.Actor   
  ORDER BY Id)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !, "Id: ",id  
 Write " Name: ",actorname  
 Set ActorOref=##class(MyPackage.Actor).%OpenId(id)   
 For Key=1:1:ActorOref.MyRentals.Count() { ;Display each element  
   Write !,"Key: ",Key ;of Collection List  
   Write " - ",ActorOref.MyRentals.GetAt(Key).Street ;Property: MyRentals – Street

   Write ?30,ActorOref.MyRentals.GetAt(Key).City ;Property: MyRentals - City  
   Write ?50,ActorOref.MyRentals.GetAt(Key).State ;Property: MyRentals - State  
   Write ?60,ActorOref.MyRentals.GetAt(Key).Zip ;Property: MyRentals - Zip  
 }

&sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 Name: John Wayne

Id: 2 Name: Jodie Foster

Id: 3 Name: Clint Eastwood

Id: 4 Name: Julie Andrews

Id: 5 Name: Johnny Depp

Id: 6 Name: Carol Burnett

Id: 7 Name: Will Smith

Id: 8 Name: Ann Margaret

Id: 9 Name: Dean Martin

Key: 1 - 123 Main St. Pittsburgh PA 01600

Key: 2 - 53 Elm St. L.A. CA 95602

Key: 3 - 9 Pershing St. Worcester MA 01752

Id: 10 Name: Ally Sheedy

Id: 11 Name: Humphrey Bogart

Id: 12 Name: Katharine Hepburn

Example ‑ Display MyRentals Data using Dynamic SQL

The MyRentals Property is defined as a

Collection List of References to Embedded Objects

Set MyQuery="SELECT Id, Name FROM MyPackage.Actor"  
   
 Set ResultSet=##class(%ResultSet).%New("%DynamicQuery:SQL")  
  
 Set SC=ResultSet.Prepare(MyQuery)  
  
 Set SC=ResultSet.Execute()  
  
 While ResultSet.Next() {  
  Set Id=ResultSet.Data("ID")  
     Write !, "Id:",Id  
     Write " Name: ",ResultSet.Data("Name")  
  Set ActorOref=##class(MyPackage.Actor).%OpenId(Id)   
  Set Key = "" Do {   
        Set Key=ActorOref.MyRentals.Next(Key) ;get the next Key  
        If Key'="" {  
   Write !,"Key: ",Key  
   Write " - ",ActorOref.MyRentals.GetAt(Key).Street ;Prop: MyRentals – Street

   Write ?30,ActorOref.MyRentals.GetAt(Key).City ;Prop: MyRentals - City  
   Write ?50,ActorOref.MyRentals.GetAt(Key).State ;Prop: MyRentals - State  
   Write ?60,ActorOref.MyRentals.GetAt(Key).Zip ;Prop: MyRentals - Zip  
       }   
    } While (Key '= "")  
  }  
  Set SC=ResultSet.Close()

If you save the above code in a routine and then run the it from the Terminal, you should get the following output.

Id: 1 Name John Wayne

Id: 2 Name Jodie Foster

Id: 3 Name Clint Eastwood

Id: 4 Name Julie Andrews

Id: 5 Name Johnny Depp

Id: 6 Name: Carol Burnett

Id: 7 Name: Will Smith

Id: 8 Name: Ann Margaret

Id: 9 Name: Dean Martin

Key: 1 - 123 Main St. Pittsburgh PA 01600

Key: 2 - 53 Elm St. L.A. CA 95602

Key: 3 - 9 Pershing St. Worcester MA 01752

Id: 10 Name: Ally Sheedy

Id: 11 Name: Humphrey Bogart

Id: 12 Name: Katharine Hepburn

*“Whatever you think it’s gonna take, double it. That applies to money, time, stress. It’s gonna be harder that you think and take longer that you think.”*

*– Richard A. Cortese*

# Class & Object Properties - Collection Array of References to Embedded Objects

Table ‑ Object Properties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object Property | Data Type | Reference  Object Class  and Name | Name we chose | Special Considerations |
| Data Types | %String |  | Name  (Name of Actor) |  |
| Collection List of Data Types | %String |  | MyShirts  (Shirts belonging to an Actor) | Total length cannot exceed 32k |
| Collection Array of Data Types | %String |  | MyHats  (Hats belonging to an Actor) | No length limit  In SQL must be handled as a Child table  Must specify a Key when accessing |
| Collection List of References to Persistent Objects |  | Persistent Objects | MyContacts  (Contacts of the Actor) |  |
| Collection Array of References to Persistent Objects |  | Persistent Objects | MyClients  (Clients of the Actor) |  |
| Collection List of References to Embedded Objects |  | Embedded Objects  %Serial Class | MyRentals  (Rental Property of the Actor) |  |
| Collection Array of References to Embedded Objects |  | Embedded Objects  %Serial Class | MyPets  (Pets of the Actor) |  |

Example ‑ Class Definition – Creating the Pets Class

Class MyPackage.Pets Extends %SerialObject  
{  
  
Property Name As %String;

Property Breed As %String;  
  
Property Color As %String;

Property Weight As %String;  
  
}

Example ‑ Actor Class Redefinition – add Reference Property that will point to the Pets Class

Class MyPackage.Actor Extends %Persistent  
{  
  
Property Name As %String [ Required ];  
  
Index NameIndex On Name;  
  
Property MyAccountant As Accountants;  
  
Property MyHome As Address;  
  
Property MyShirts As list Of %String;  
  
Property MyHats As array Of %String;  
  
Property MyContacts As list Of Contacts;  
  
Property MyClients As array Of Clients;  
  
Property MyHomes As list Of Rentals;  
  
Property MyPets As array Of Pets;  
}

Example ‑ Clear and Count Methods - Deletes all elements and show the count

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

Clear Method – Clears or Deletes the Collection of Elements

Count Method - Returns the Number of Elements in the Collection

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann Margaret

;into memory

Do ActorOref.MyPets.Clear() ;clear Array of Pets  
 Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 0

1

Example ‑ SetAt Method – Insert three Pets into the Collection

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

SetAt Method - Set an Oref at the specific Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Set PetsOref1=##class(MyPackage.Pets).%New() ;create new Pet Oref1

Set PetsOref1.Name="Sandy" ;Set the Name

Set PetsOref1.Breed="Dog" ;Set the Breed

Set PetsOref1.Color="Brown" ;Set the Colr

Set PetsOref1.Weight="35" ;Set the Weight

Set PetsOref2=##class(MyPackage.Pets).%New() ;create new Pet Oref2

Set PetsOref2.Name="Tiger" ;Set the Name

Set PetsOref2.Breed="Cat" ;Set the Breed

Set PetsOref2.Color="Striped" ;Set the Color

Set PetsOref2.Weight="10" ;Set the Weight

Set PetsOref3=##class(MyPackage.Pets).%New() ;create new Pet Oref3

Set PetsOref3.Name="Lips" ;Set the Name

Set PetsOref3.Breed="Pig" ;Set the Breed

Set PetsOref3.Color="N/A" ;Set the Color

Set PetsOref3.Weight="70" ;Set the Weight

Do ActorOref.MyPets.SetAt(PetsOref1,1) ;associate PetsOref1 with Actor at Key 1

Do ActorOref.MyPets.SetAt(PetsOref2,2) ;associate PetsOref2 with Actor at Key 2

Do ActorOref.MyPets.SetAt(PetsOref3,3) ;associate PetsOref3 with Actor at Key 3

Write !,"Count: ",ActorOref.MyPets.Count()

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 3

1

Example ‑ Global generated from Class MyPackage.Actor

zw ^MyPackage.ActorD

^MyPackage.ActorD=13

^MyPackage.ActorD(1)=$lb("","John Wayne","","",$lb("BlueShirt","CyanShirt",

"RedShirt","WhiteShirt"))

^MyPackage.ActorD(2)=$lb("","Jodie Foster","","","")

^MyPackage.ActorD(2,"MyHats",1)="Bowler"

^MyPackage.ActorD(2,"MyHats",2)="BigBirdHat"

^MyPackage.ActorD(2,"MyHats","Top")="TopHat"

^MyPackage.ActorD(3)=$lb("","Clint Eastwood","2",$lb("","","",""),"Cyan")

^MyPackage.ActorD(4)=$lb("","Julie Andrews","",$lb("123 Main St.","

Marlboro","MA","01752"),"Brown")

^MyPackage.ActorD(5)=$lb("","Johnny Depp","",$lb("","","",""),"Tan","",$lb($lb("1"),$lb("6"),$lb("3"),$lb("4")))

^MyPackage.ActorD(6)=$lb("","Carol Burnett","",$lb("","","",""),"Red","","")

^MyPackage.ActorD(6,"MyClients",1)=1

^MyPackage.ActorD(6,"MyClients",2)=6

^MyPackage.ActorD(6,"MyClients",4)=4

^MyPackage.ActorD(7)=$lb("","Will Smith","",$lb("","","",""),"Navy")

^MyPackage.ActorD(8)=$lb("","Ann Margaret","",$lb("","","",""),"Yellow","","","")

^MyPackage.ActorD(8,"MyPets",1)=$lb("Sandy","Dog","Brown","35")

^MyPackage.ActorD(8,"MyPets",2)=$lb("Tiger","Cat","Striped","10")

^MyPackage.ActorD(8,"MyPets",3)=$lb("Lips","Pig","N/A","70")

^MyPackage.ActorD(9)=$lb("","Dean Martin","",$lb("","","",""),

"Green","","",$lb($lb($lb("123 Main St.","Pittsburgh","PA","01600")),$lb($lb("53 Elm St.","L.A.","CA","95602")),$lb($lb("9 Pershing St.","Worcester","MA","01752"))))

^MyPackage.ActorD(10)=$lb("","Ally Sheedy","",$lb("","","",""),"Black")

^MyPackage.ActorD(11)=$lb("","Humphrey Bogart","",$lb("","","",""),"Brown")

^MyPackage.ActorD(12)=$lb("","Katharine Hepburn","2",$lb("","","",""),"Blue")

Example ‑ IsDefined Method – See if a Key is defined

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

IsDefined Method – see if a Key is defined

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Write ActorOref.MyPets.IsDefined(1) ;Is key 1 defined? - Yes  
1

Write ActorOref.MyPets.IsDefined(2) ;Is key 2 defined? - Yes  
1

Write ActorOref.MyPets.IsDefined(4) ;Is key 4 defined? - No  
0

Example ‑ GetAt Method – Returns the Element associated with a Key

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

GetAt Method - Returns the Element associated with a Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements

For Key=1:1:ActorOref.MyPets.Count() { ;Display each element  
   Write !,"Key: ",Key ;of Collection Array  
   Write " - ",ActorOref.MyPets.GetAt(Key).Name ;Property: MyPets – Name

   Write ?20,ActorOref.MyPets.GetAt(Key).Breed ;Property: MyPets - Breed  
   Write ?30,ActorOref.MyPets.GetAt(Key).Color ;Property: MyPets - Color  
   Write ?40,ActorOref.MyPets.GetAt(Key).Weight ;Property: MyPets - Weight  
 }

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 3

Key: 1 - Sandy Dog Brown 35

Key: 2 - Tiger Cat Striped 10

Key: 3 - Lips Pig N/A 70

Example ‑ Find Method – Finds the associated Id for a String

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

Find Code – Finds the associated Id for a String

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Set NameToFind="Sandy" ;Pet Name to Find

Set FoundId="" ;initialized FoundId

For Id=1:1:ActorOref.MyPets.Count() {

    If NameToFind=ActorOref.MyPets.GetAt(Id).Name Set FoundId=Id

}

If FoundId’="" {

Write !, "Found: ", NameToFind," at Id: ",FoundId

}

Else{

Write !, "Could not find: ", NameToFind

}

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Found: Sandy at Id: 1

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Set ColorToFind="Striped" ;Color to Find

Set FoundId="" ;initialized FoundId

For Id=1:1:ActorOref.MyPets.Count() {

    If ColorToFind=ActorOref.MyPets.GetAt(Id).Color Set FoundId=Id

}

If FoundId’="" {

Write !, "Found: ", ColorToFind," at Id: ",FoundId

}

Else{

Write !, "Could not find: ", ColorToFind

}

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Found: Striped at Id: 2

Table ‑ Traversing Methods

|  |  |  |
| --- | --- | --- |
| Method |  | How Key or Slot is passed |
| Next | Traversing Forward | Key or Slot passed by Value |
| Previous | Traversing Backward | Key or Slot Passed by Value |
| GetNext | Traversing Forward | Key or Slot Passed by Reference |
| GetPrevious | Traversing Backward | Key or Slot Passed by Reference |

Example ‑ Next Method – Returns the Element for the next Key

The MyPets Property is defined as a

Collection Array of References to Embedded Objects.

Next Method - Returns the Element for the next Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements

Set Key = "" Do { ;start with Key 0

Set Key=ActorOref.MyPets.Next(Key) ;get the next Key  
   If Key'="" {  
 Write !,"Key: ",Key ;display the Key   
 Write " - ",ActorOref.MyPets.GetAt(Key).Name ;Property: MyPets – Name

Write ?20,ActorOref.MyPets.GetAt(Key).Breed ;Property: MyPets - Breed  
 Write ?30,ActorOref.MyPets.GetAt(Key).Color ;Property: MyPets - Color  
 Write ?40,ActorOref.MyPets.GetAt(Key).Weight ;Property: MyPets - Weight  
   }   
 } While (Key '= "")

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 3

Key: 1 - Sandy Dog Brown 35

Key: 2 - Tiger Cat Striped 10

Key: 3 - Lips Pig N/A 70

Example ‑ Previous Method – Returns the Element for the previous Key

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

Previous Method - Returns the Element for the previous Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements

Set Key = "" Do { ;start with Key null

Set Key=ActorOref.MyPets.Previous(Key) ;get the next Key  
   If Key'="" {  
 Write !,"Key: ",Key ;display the Key   
 Write " - ",ActorOref.MyPets.GetAt(Key).Name ;Property: MyPets – Name

Write ?20,ActorOref.MyPets.GetAt(Key).Breed ;Property: MyPets - Breed  
 Write ?30,ActorOref.MyPets.GetAt(Key).Color ;Property: MyPets - Color  
 Write ?40,ActorOref.MyPets.GetAt(Key).Weight ;Property: MyPets - Weight  
   }    
 } While (Key '= "")

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 3

Key: 3 - Lips Pig N/A 70

Key: 2 - Tiger Cat Striped 10

Key: 1 - Sandy Dog Brown 35

Example ‑ GetNext Method – Returns the Element for the next Key

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

GetNext Method - Returns the Element for the next Key

 Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann

;Margaret into memory  
 Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements  
 Set Key = "" Do {   
    Set MyPetsOref=ActorOref.MyPets.GetNext(.Key)   
    If Key'="" {  
      Write !, "Key: ",Key

      Write " – ",MyPetsOref.Name ;Property: MyPets – Name  
      Write ?20,MyPetsOref.Breed ;Property: MyPets - Breed  
      Write ?30,MyPetsOref.Color ;Property: MyPets - Color  
      Write ?40,MyPetsOref.Weight ;Property: MyPets - Weight  
    }   
 } While (Key '= "")

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 3

Key: 1 - Sandy Dog Brown 35

Key: 2 - Tiger Cat Striped 10

Key: 3 - Lips Pig N/A 70

Example ‑ GetPrevious Method – Returns the Element for the Previous Key

The MyPets Property is defined as a

Collection Array of References to Embedded Objects

GetPrevious Method - Returns the Element for the previous Key

 Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann

;Margaret into memory  
 Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements  
 Set Key = "" Do {   
    Set MyPetsOref=ActorOref.MyPets.GetPrevious(.Key)   
    If Key'="" {  
      Write !, "Key: ",Key

      Write " – ",MyPetsOref.Name ;Property: MyPets – Name  
      Write ?20,MyPetsOref.Breed ;Property: MyPets - Breed  
      Write ?30,MyPetsOref.Color ;Property: MyPets - Color  
      Write ?40,MyPetsOref.Weight ;Property: MyPets - Weight  
    }   
 } While (Key '= "")

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 3

Key: 3 - Lips Pig N/A 70

Key: 2 - Tiger Cat Striped 10

Key: 1 - Sandy Dog Brown 35

Example ‑ SetAt Method – Replace a specific Pet

The MyPets Property is defined as a Collection Array of References to Embedded Objects

SetAt Method - Set an Oref at the specific Key

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Set PetsOref=##class(MyPackage.Pets).%New() ;create new Pet Oref

Set PetsOref.Name="Trixie" ;Set the Name

Set PetsOref.Breed="Dog" ;Set the Breed

Set PetsOref.Color="White" ;Set the Color

Set PetsOref.Weight="50" ;Set the Weight

Do ActorOref.MyPets.SetAt(PetsOref,2) ;associate PetsOref with the Actress

;and replace the second pet

Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements  
 Set Key = "" Do {   
    Set MyPetsOref=ActorOref.MyPets.GetNext(.Key)   
    If Key'="" {  
      Write !, "Key: ",Key

      Write " – ",MyPetsOref.Name ;Property: MyPets – Name  
      Write ?20,MyPetsOref.Breed ;Property: MyPets - Breed  
      Write ?30,MyPetsOref.Color ;Property: MyPets - Color  
      Write ?40,MyPetsOref.Weight ;Property: MyPets - Weight  
    }   
 } While (Key '= "")

Write !,ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 3

Key: 1 - Sandy Dog Brown 35

Key: 2 - Trixie Dog White 50

Key: 3 - Lips Pig N/A 70

Example ‑ RemoveAt Method – Remove a specific Pet

The MyPets Property is defined as a Collection Array of References to Embedded Objects

RemoveAt Method – Remove a specific Pet

Set ActorOref=##class(MyPackage.Actor).%OpenId(8) ;bring object Ann ;Margaret into memory

Do ActorOref.MyPets.RemoveAt(3) ;Remove Pet at Key 3

Write !,"Count: ",ActorOref.MyPets.Count() ;count of elements

 Set Key = "" Do {   
    Set MyPetsOref=ActorOref.MyPets.GetNext(.Key)   
    If Key'="" {  
      Write !, "Key: ",Key

      Write " – ",MyPetsOref.Name ;Property: MyPets – Name  
      Write ?20,MyPetsOref.Breed ;Property: MyPets - Breed  
      Write ?30,MyPetsOref.Color ;Property: MyPets - Color  
      Write ?40,MyPetsOref.Weight ;Property: MyPets – Weight

}  
 }  While Key'=""

Write !, ActorOref.%Save() ;Save the object

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Count: 2

Key: 1 - Sandy Dog Brown 35

Key: 2 - Trixie Dog White 50

1

Example ‑ Display Array of References to Embedded Objects using Embedded SQL

 New actorname,mypets  
 &sql(Declare MyCursor CURSOR FOR  
  SELECT Actor->Name, MyPets  
  INTO :actorname, :mypets  
  FROM MyPackage.Actor\_MyPets   
  ORDER BY Name)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !,"Name: ",actorname  
  Write ?25,"MyPets: ",mypets  
 &sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Name: Ann Margaret MyPets: SandyDogBrown35

Name: Ann Margaret MyPets:TrixieDogWhite50

Example ‑ Display Array of References to Embedded Objects using Dynamic SQL

 Set MyQuery="SELECT Actor->Name,MyPets FROM MyPackage.Actor\_MyPets"  
   
 Set ResultSet=##class(%ResultSet).%New("%DynamicQuery:SQL")  
 ;Create a new Instance of %ResultSet  
  
 Set SC=ResultSet.Prepare(MyQuery) ;Prepare the Query  
  
 Set SC=ResultSet.Execute() ;Execute the Query  
  
 While ResultSet.Next() { ;Process the Query results  
     Write !,ResultSet.Data("Name")," - "  
     Write "Pet Name: ",$LI(ResultSet.Data("MyPets"),1)  
     Write ?35," Breed: ",$LI(ResultSet.Data("MyPets"),2)  
     Write ?50," Color: ",$LI(ResultSet.Data("MyPets"),3)  
     Write ?65," Weight: ",$LI(ResultSet.Data("MyPets"),4)  
  }  
  
 Set SC=ResultSet.Close() ;Close the Query

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Ann Margaret - Pet Name: Sandy Breed: Dog Color: Brown Weight: 35

Ann Margaret - Pet Name: Trixie Breed: Dog Color: White Weight: 50

*“Everything is funny as long as it is happening to somebody else.” – Will Rogers*

# Object Relationships

Table ‑ Table comparing One-to-Many and Parent-to-Child Relationships

|  |  |  |  |
| --- | --- | --- | --- |
| Relationship | One-to-Many | Parent-to-Child |  |
| Persons to Cars | Yes | No | Cars are not dependent upon persons and a person may own many cars |
| Books to Pages | No | Yes | Pages of a book cannot exist without a Book |
| Lawyer to Clients | Yes | No | Clients are not dependent upon Lawyers and Lawyers can have many clients |
| Parent to Children | No | Yes | Children cannot exist without Parents |

Table ‑ One-to-many relationship of a Lawyer to his clients

|  |  |  |  |
| --- | --- | --- | --- |
|  | Classes | | |
|  | MyPackage.Lawyer |  | MyPackage.Client |
| **Name** | LawyerName |  | ClientName |
| **Relationship Name** | MyClients |  | MyLawyer |
| **Inverse** | MyLawyer |  | MyClients |
| **Cardinality** | Many  (Many Clients) |  | One  (One Lawyer) |

Example ‑ Define the Lawyer Class

Class MyPackage.Lawyer Extends %Persistent  
 {  
  
 Property LawyerName As %String [ Required ];  
  
 }

Example ‑ Define the Client Class

Class MyPackage.Client Extends %Persistent  
 {  
  
 Property ClientName As %String [ Required ];  
  
 }

Example ‑ Add a Relationship to the Lawyer Class

Class MyPackage.Lawyer Extends %Persistent  
 {  
  
 Property LawyerName As %String [ Required ];  
  
 Relationship MyClients As MyPackage.Client [ Cardinality = many, Inverse = MyLawyer ];  
  
 }

Example ‑ The Relationship will automatically be added to the Client Class

Class MyPackage.Client Extends %Persistent  
 {  
  
 Property ClientName As %String [ Required ];  
  
 Relationship MyLawyer As MyPackage.Lawyer [ Cardinality = one, Inverse = MyClients ];  
  
 Index MyLawyerIndex On MyLawyer;  
  
 }

Example ‑ Populate the Lawyer and Client Classes

Set LawyerOref=##class(MyPackage.Lawyer).%New() ;create new lawyer

Set LawyerOref.LawyerName="HighPoweredLawyer" ;lawyer name

Do LawyerOref.%Save() ;save lawyer

Set ClientOref1=##class("MyPackage.Client").%New() ;create new client

Set ClientOref1.ClientName="PoorClient1" ;client name

Do ClientOref1.%Save() ;save client

Set ClientOref2=##class("MyPackage.Client").%New() ;create new client

Set ClientOref2.ClientName="PoorClient2" ;client name

Do ClientOref2.%Save() ;save client

Set ClientOref3=##class("MyPackage.Client").%New() ;create new client

Set ClientOref3.ClientName="PoorClient3" ;client name

Do ClientOref3.%Save() ;save client

Example ‑ Link the Lawyer with the three Clients

Set LawyerOref=##class(MyPackage.Lawyer).%OpenId(1)

Set ClientOref1=##class(MyPackage.Client).%OpenId(1)

Set ClientOref2=##class(MyPackage.Client).%OpenId(2)

Set ClientOref3=##class(MyPackage.Client).%OpenId(3)

Set ClientOref1.MyLawyer=LawyerOref ;link lawyer with Client1

Set ClientOref2.MyLawyer=LawyerOref ;link lawyer with Client2

Set ClientOref3.MyLawyer=LawyerOref ;link lawyer with Client3

Do LawyerOref.%Save()

Do ClientOref1.%Save()

Do ClientOref2.%Save()

Do ClientOref3.%Save()

Example ‑ Access the relationship links

; Find MyClients from the Lawyer Side

Set LawyerOref=##class(MyPackage.Lawyer).%OpenId(1)

Write LawyerOref.LawyerName ;lawyer name

HighPoweredLawyer

;Using the GetAt method, and going from the Lawyer side, we access Client1

Write LawyerOref.MyClients.GetAt(1).ClientName

PoorClient1

;Using the GetAt method, and going from the Lawyer side, we access Client2

Write LawyerOref.MyClients.GetAt(2).ClientName

PoorClient2

;Using the GetAt method, and going from the Lawyer side, we access Client3

Write LawyerOref.MyClients.GetAt(3).ClientName

PoorClient3

;Access the Lawyer from the Client Side

Set ClientOref=##class(MyPackage.Client).%OpenId(1)

Write ClientOref.MyLawyer.LawyerName

HighPoweredLawyer

Example ‑ Add another Client (Insert Method) and link it with the Lawyer

Set ClientOref4=##class(MyPackage.Client).%New() ;define a new cliet

Set ClientOref4.ClientName="PoorClient4" ;Client4

Do ClientOref4.%Save() ;Save Client4

Set LawyerOref=##class(MyPackage.Lawyer).%OpenId(1) ;bring our Lawyer into memory

Do LawyerOref.MyClients.Insert(ClientOref4) ;link Client 4

;with our Lawyer

Do LawyerOref.%Save()

Do ClientOref4.%Save()

Set ClientOref=##class(MyPackage.Client).%OpenId(4)

Write ClientOref.ClientName

PoorClient4

Write ClientOref.MyLawyer.LawyerName

HighPoweredLaywer

Example ‑ Displaying our Relationship data between Lawyer and Clients

Set LawyerOref=##class(MyPackage.Lawyer).%OpenId(1)  
   
 Set Key="" Do {  
 Set ClientOref=LawyerOref.MyClients.GetNext(.Key)  
 If ClientOref'="" {  
 Write "Key: ",Key," Name: ",ClientOref.ClientName,! }  
 } While Key'=""

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Key: 1 Name: PoorClient1

Key: 2 Name: PoorClient2

Key: 3 Name: PoorClient3

Key: 4 Name: PoorClient4

Example ‑ Displaying our Relationship data between Lawyer and Clients, using UnSwizzle

Set LawyerOref=##class(MyPackage.Lawyer).%OpenId(1)  
   
 Set Key="" Do {  
 Set ClientOref=LawyerOref.MyClients.GetNext(.Key)  
 If ClientOref'="" {  
 Write "Key: ",Key," Name: ",ClientOref.ClientName,!  
 Do LawyerOref.MyClients.%UnSwizzleAt(Key)  
 }  
 } While Key'=""

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

Key: 1 Name: PoorClient1

Key: 2 Name: PoorClient2

Key: 3 Name: PoorClient3

Key: 4 Name: PoorClient4

Example ‑ Embedded SQL to display the Relationship between Lawyer and Clients

 New lawyerName,clientName  
 &sql(Declare MyCursor CURSOR FOR  
  SELECT MyLawyer->LawyerName, ClientName  
  INTO :lawyerName, :clientName  
  FROM MyPackage.Client  
  ORDER BY LawyerName)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !,"Lawyer Name: ",lawyerName  
  Write ?30," - Client Name: ",clientName  
 &sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Lawyer Name: HighPoweredLawyer - Client Name: PoorClient1

Lawyer Name: HighPoweredLawyer - Client Name: PoorClient2

Lawyer Name: HighPoweredLawyer - Client Name: PoorClient3

Lawyer Name: HighPoweredLawyer - Client Name: PoorClient4

Example ‑ Global generated from Class MyPackage.Client

ZW ^MyPackage.ClientD

^MyPackage.ClientD(1)=$lb("","PoorClient1","1")

^MyPackage.ClientD(2)=$lb("","PoorClient2","1")

^MyPackage.ClientD(3)=$lb("","PoorClient3","1")

^MyPackage.ClientD(4)=$lb("","PoorClient4","1")

Example ‑ Global generated from Class MyPackage.Lawyer

ZW ^MyPackage.LawyerD

^MyPackage.LawyerD(1)=$lb("","HighPoweredLawyer")

Example ‑ Delete a Client

 Set Key="" Do {  
 Set LawyerOref=##class(MyPackage.Lawyer).%OpenId(1)  
 Set ClientOref=LawyerOref.MyClients.GetNext(.Key)  
 If ClientOref'="" {  
 Write !,"Key: ",Key  
 Write " Name: ",ClientOref.ClientName  
 If ClientOref.ClientName="PoorClient2" {  
 Set status=##class(MyPackage.Client).%DeleteId(Key)  
 If status=1 Write " – deleted."  
 If status’=1 Write " – deleted failed."

}  
 }  
 } While Key'=""

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Key: 1 Name: PoorClient1

Key: 2 Name: PoorClient2 - deleted.

Key: 3 Name: PoorClient3

Key: 4 Name: PoorClient4

Example ‑ Delete the Lawyer

Set Status=##class(MyPackage.Lawyer).%DeleteId(1)

If Status=1 Write "Lawyer Delete succeded."

If Status’=1 Write "Lawyer Delete failed."

Lawyer Delete failed.

Example ‑ Delete a Lawyer after deleting all associated Clients

 Set LawyerOref=##class(MyPackage.Lawyer).%OpenId(1)  
   
 Set Key="" Do {  
 Set ClientOref=LawyerOref.MyClients.GetNext(.Key)  
 If Key'="" {  
 Set Status=##class(MyPackage.Client).%DeleteId(Key)  
 If Status=1 Write !,"Delete successful for key: ",Key  
 If Status’=1 Write !,"Delete failed for key: ",Key  
 }  
 } While Key'=""  
  
 Set Status=##class(MyPackage.Lawyer).%DeleteId(1)  
 If Status=1 Write !,"Lawyer Delete succeded."  
 If Status’=1 Write !,"Lawyer Delete failed."

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Delete successful for key: 1

Delete successful for key: 2

Delete successful for key: 3

Lawyer Delete succeded.

Table ‑ One-to-many relationship of a Parent to his children

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Classes** | | |
| **Class** | **MyPackage.Parent** |  | **MyPackage.Child** |
| **Name** | ParentName |  | ChildName |
| **Relationship Name** | MyChildren |  | MyParent |
| **Inverse** | MyParent |  | MyChildren |
| **Cardinality** | Children |  | Parent |

Example ‑ Define the Parent Class

Class MyPackage.Parent Extends %Persistent  
 {  
  
 Property ParentName As %String;  
  
 }

Example ‑ Define the Child Class

Class MyPackage.Child Extends %Persistent  
 {  
  
 Property ChildName As %String;  
  
 }

Example ‑ Add a Relationship to the Parent Class

Class MyPackage.Parent Extends %Persistent  
 {  
  
 Property ParentName As %String;  
  
 Relationship MyChildren As MyPackage.Child [ Cardinality = children, Inverse = MyParent ];  
  
 }

Example ‑ The Relationship will automatically be added to the Child Class

Class MyPackage.Child Extends %Persistent  
 {  
  
 Property ChildName As %String;  
  
 Relationship MyParent As MyPackage.Parent [ Cardinality = parent, Inverse = MyChildren ];  
  
 }

Example ‑ Populate the Parent and Child Classes and Link the two

Set ParentOref=##class(MyPackage.Parent).%New() ;create new Parent

Set ParentOref.ParentName="Mr John Parent" ;Parent name

Set ChildOref1=##class("MyPackage.Child").%New() ;create new Child

Set ChildOref1.ChildName="Little Susie Child" ;Child name

Set ChildOref2=##class("MyPackage.Child").%New() ;create new Child

Set ChildOref2.ChildName="Little Judy Child" ;Child name

Set ChildOref3=##class("MyPackage.Child").%New() ;create new Child

Set ChildOref3.ChildName="Mean Cheryl Child" ;Child name

Set ChildOref1.MyParent=ParentOref ;link Parent with first Child

Set ChildOref2.MyParent=ParentOref ;link Parent with second Child

Set ChildOref3.MyParent=ParentOref ;link Parent with third Child

Do ParentOref.%Save() ;save Parent and Children

Example ‑ Access the relationship links

Set ParentOref=##class(MyPackage.Parent).%OpenId(1)

Write ParentOref.ParentName ;Parent name

Mr John Parent

;Using the GetAt method, and going from the Parent side, we access Child1

Write ParentOref.MyChildren.GetAt(1).ChildName

Little Susie Child

;Using the GetAt method, and going from the Parent side, we access Child2

Write ParentOref.MyChildren.GetAt(2).ChildName

Little Judy Child

;Using the GetAt method, and going from the Parent side, we access Child3

Write ParentOref.MyChildren.GetAt(3).ChildName

Mean Cheryl Child

Example ‑ Add another Client (Insert Method) and associate it with the Parent

Set ChildOref=##class("MyPackage.Child").%New() ;create new Child

Set ChildOref.ChildName="Little Chuck Child" ;Child name

Set ParentOref=##class(MyPackage.Parent).%OpenId(1) ;bring our Parent into

Do ParentOref.MyChildren.Insert(ChildOref) ;memory assocate Child

;with our Parent

Do ParentOref.%Save() ;Save Parent and Child

Example ‑ Displaying our Relationship data between Parent and Children

Set ParentOref=##class(MyPackage.Parent).%OpenId(1)  
  
 Set Key="" Do {  
    Set ChildOref=ParentOref.MyChildren.GetNext(.Key)  
    If ChildOref'="" {

Write "Key: ",Key

Write " Name: ",ChildOref.ChildName,!

}  
 } While Key'=""

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Key: 1 Name: Little Susie Child

Key: 2 Name: Little Judy Child

Key: 3 Name: Mean Cheryl Child

Key: 4 Name: Little Chuck Child

Example ‑ Displaying our Relationship data between Parent and Children, using UnSwizzle

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ParentOref=##class(MyPackage.Parent).%OpenId(1)  
  
 Set Key="" Do {  
    Set ChildOref=ParentOref.MyChildren.GetNext(.Key)  
    If ChildOref'="" {

Write "Key: ",Key

Write " Name: ",ChildOref.ChildName,!

       Do ParentOref.MyChildren.%UnSwizzleAt(Key)

}  
 } While Key'=""

Example ‑ Embedded SQL to display the Relationship between Parent and Children

New ParentName,ChildName  
 &sql(Declare MyCursor CURSOR FOR  
  SELECT MyParent->ParentName, ChildName  
  INTO :ParentName, :ChildName  
  FROM MyPackage.Child  
  ORDER BY ParentName)  
 &sql(OPEN MyCursor)  
 &sql(FETCH MyCursor)  
 While (SQLCODE = 0) {  
  Write !,"Parent Name: ",ParentName  
  Write ?30," - Child Name: ",ChildName  
 &sql(FETCH MyCursor)  
 }  
 &sql(CLOSE MyCursor)

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Parent Name: Mr John Parent - Child Name: Mean Cheryl Child

Parent Name: Mr John Parent - Child Name: Little Susie Child

Parent Name: Mr John Parent - Child Name: Little Judy Child

Parent Name: Mr John Parent - Child Name: Little Chuck Child

Example ‑ Global generated from Class MyPackage.Child

zw ^MyPackage.ChildD

^MyPackage.ChildD=4

^MyPackage.ChildD(1,1)=$lb("","Mean Cheryl Child")

^MyPackage.ChildD(1,2)=$lb("","Little Susie Child")

^MyPackage.ChildD(1,3)=$lb("","Little Judy Child")

^MyPackage.ChildD(1,4)=$lb("","Little Chuck Child")

Example ‑ Global generated from Class MyPackage.Parent

zw ^MyPackage.ParentD

^MyPackage.ParentD=1

^MyPackage.ParentD(1)=$lb("","Mr John Parent")

Example ‑ Delete a Child

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

Set ChildKey="" Do {  
 Set ParentKey=1  
  Set ParentOref=##class(MyPackage.Parent).%OpenId(ParentKey)  
  Set ChildOref=ParentOref.MyChildren.GetNext(.ChildKey)  
  If ChildOref'="" {  
  Write !,"ChildKey: ",ChildKey  
  Write " Name: ",ChildOref.ChildName  
  If ChildOref.ChildName="Little Chuck Child" {  
   Set CombinedKey=ParentKey\_"||"\_ChildKey  
  Set status=##class(MyPackage.Child).%DeleteId(CombinedKey)  
  If status=1 Write " - deleted."  
  If status'=1 Write " - deleted failed."  
     }  
    }  
 } While ChildKey'=""

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

ChildKey: 1 Name: Mean Cheryl Child

ChildKey: 2 Name: Little Susie Child

ChildKey: 3 Name: Little Judy Child

ChildKey: 4 Name: Little Chuck Child - deleted.

Example ‑ Delete the Parent

Set Status=##class(MyPackage.Parent).%DeleteId(1)

If Status=1 Write "Parent Delete succeeded."

If Status’=1 Write "Parent Delete failed."

The above code may be run from the Terminal.

Parent Delete succeeded.

*Customer: ”I have a message on my screen that says: 'Disk Full'. What can that be?”*

*Tech Support: “Maybe your disk is full”.*

*Customer: ”Hmmm. OK.”*

# Methods

Example ‑: Class Definition for MyPackage.Cars

Class MyPackage.Cars Extends %Persistent  
 {

}

Example ‑ Class and Property Definitions for MyPackage.Cars

Class MyPackage.Cars Extends %Persistent  
 {  
  
 Property MakeModel As %String;   
  
 Property Year As %String;  
  
 Property Color As %String;

Property EnteredBy As %String;

Property EnteredById As %String;

}

Example ‑ Class Method: AddNewCar

Class MyPackage.Cars Extends %Persistent  
 {  
  
 Property MakeModel As %String;   
  
 Property Year As %String;  
  
 Property Color As %String;

Property EnteredBy As %String;

Property EnteredById As %String;

ClassMethod AddNewCar(UserName As %String, UserId As %String)  
{  
  Set CarOref=##class(MyPackage.Cars).%New() ;create a new object  
  
  Read !,"Enter New MakeModel: ",MakeModel ;accept MakeModel from user  
  Read !,"Enter New Year: ",Year ;accept Year from user  
  Read !,"Enter New Color: ",Color ;accept Color from user  
  
  Set CarOref.MakeModel=MakeModel ;Set MakeModel into object  
  Set CarOref.Year=Year ;Set Year into object  
  Set CarOref.Color=Color ;Set Color into object  
  Set CarOref.EnteredBy=UserName ;Set EnteredBy  
  Set CarOref.EnteredById=UserId ;Set EnteredId  
  
  Set Status=CarOref.%Save()

If Status'=1 Write "Error from CarOref.%Save()" Quit Status  
  Set Id=CarOref.%Id()  
  Write !,"New Object with Id: ",Id," Saved",!

Quit $$$OK

}

}

Example ‑ Run AddNewCar Method

Kill ;Kill all local variables

Write ##class(MyPackage.Cars).AddNewCar("Jack Frost","12543") ;you run this

Enter New MakeModel: Chevy

Enter New Year: 2001

Enter New Color: Yellow

New Object with ID: 1 Saved

1

Kill ;Kill all local variables

Write ##class(MyPackage.Cars).AddNewCar("Amy Frost","43783")

Enter New MakeModel: Ford

Enter New Year: 2008

Enter New Color: Green

New Object with ID: 2 Saved

1

Kill ;Kill all local variables

Write ##class(MyPackage.Cars).AddNewCar("Jill Frost","95602")

Enter New MakeModel: Dodge

Enter New Year: 2010

Enter New Color: Blue

New Object with ID: 3 Saved

1

Example ‑ DisplayCar Instance Method

Class MyPackage.Cars Extends %Persistent  
 {  
  
 Property MakeModel As %String;   
  
 Property Year As %String;  
  
 Property Color As %String;

Property EnteredBy As %String;

Property EnteredById As %String;

ClassMethod AddNewCar(UserName As %String, UserId As %String)  
{  
  Set CarOref=##class(MyPackage.Cars).%New() ;create a new object  
  
  Read !,"Enter New MakeModel: ",MakeModel ;accept MakeModel from user  
  Read !,"Enter New Year: ",Year ;accept Year from user  
  Read !,"Enter New Color: ",Color ;accept Color from user  
  
  Set CarOref.MakeModel=MakeModel ;Set MakeModel into object  
  Set CarOref.Year=Year ;Set Year into object  
  Set CarOref.Color=Color ;Set Color into object  
  Set CarOref.EnteredBy=UserName ;Set EnteredBy  
  Set CarOref.EnteredById=UserId ;Set EnteredId  
  
  Set Status=CarOref.%Save()

If Status'=1 Write "Error from CarOref.%Save()" Quit Status  
  Set Id=CarOref.%Id()  
  Write !,"New Object with Id: ",Id," Saved",!

Quit $$$OK

}

Method DisplayCar() As %String

{

 Write !,"MakeModel : ",..MakeModel   
  Write !,"Year : ",..Year   
  Write !,"Color : ",..Color

  Write !,"EnteredBy : ",..EnteredBy

  Write !,"EnteredById : ",..EnteredById

Write !

Quit $$$OK

}

}

Example ‑ Run DisplayCar Method

Set oref=##class(MyPackage.Cars).%OpenId(1) ;open ID 1

Write oref.DisplayCar()

;DisplayCar passing oref

MakeModel : Chevy

Year : 2001

Color : Yellow

EnteredBy : Jack Frost

EnteredById : 125431

1

Set oref=##class(MyPackage.Cars).%OpenId(2) ;open ID 2

Write oref.DisplayCar() ;DisplayCar passing oref

MakeModel : Ford

Year : 2008

Color : Green

EnteredBy : Amy Frost

EnteredById : 437831

1

Set oref=##class(MyPackage.Cars).%OpenId(3) ;open ID 3

Write oref.DisplayCar() ;DisplayCar passing oref

MakeModel : Dodge

Year : 2010

Color : Blue

EnteredBy : Jill Frost

EnteredById : 956021

1

Example ‑ Class Method: AddNewCar - Description

/// Description for AddNewCar Class Method

ClassMethod AddNewCar(UserName As %String, UserId As %String) As %String  
{

.

.

.  
}

Example ‑ Class Method: AddNewCar

ClassMethod AddNewCar(UserName As %String, UserId As %String) As %String  
{

.

.

.  
}

Example ‑ Instance Method: DisplayCar

Method DisplayCar() As %String   
{

.

.

.  
}

Example ‑ Class Method: AddNewCar – demonstrate Private Method

ClassMethod AddNewCar(UserName As %String, UserId As %String) As %String [Private ]  
{

.

.

.  
}

Example ‑ Class Method: AddNewCar – Attempting to access a Private Method

Write ##Class(MyPackage.Cars).AddNewCar("Jack Frost","12543")

^

<PRIVATE METHOD>

Example ‑ Class Method: AddNewCar – no input or output parameters

ClassMethod AddNewCar()  
{

.

.

.  
}

Example ‑ How to call a Method with no input or output parameters.

Do ##CLASS(MyPackage.Cars).AddNewCar()

Example ‑ Class Method: AddNewCar – Method with 2 input parameters and one output parameter

ClassMethod AddNewCar(UserName, UserId) As %Status  
{

.

.

.

Quit 1  
}

Example ‑ How to call a Method with two input and one output parameter

Write ##Class(MyPackage.Cars).AddNewCar("Jack Frost","12543")

1

Set Status=##Class(MyPackage.Cars).AddNewCar("Jack Frost","12543")

Example ‑ Class Method: AddNewCar – Method with 2 input parameters with data types and one output parameter

ClassMethod AddNewCar(UserName As %String, UserId As %String) As %Status  
{

.

.

.

Quit 1  
}

Example ‑ Class Method: AddNewCar – Method with 2 input parameters with data types and default values and one output parameter

ClassMethod AddNewCar(UserName As %String = "UserName Default", UserId As

%String = "UserId Default") As %Status  
{

.

.

.

Quit 1  
}

Example ‑ Class Method: AddNewCar – demonstrate input parameters with default value

ClassMethod AddNewCar(UserName As %String = "UserName Default", UserId As

%String = "UserId Default") As %Status  
{

.

.

.

Write !,UserName

Write !,UserId

Write !

Quit 1  
}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Write ##Class(MyPackage.Cars).AddNewCar(,)

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

UserName Default

UserId Default

1

Example ‑ Class Method: AddNewCar – Method with a Public List

ClassMethod AddNewCar(UserName, UserId) As %Status [PublicList = (Var1, Var2)]  
  
{  
}

Example ‑ Class Method: AddNewCar

/// This method adds a New Car Object to the database.  
ClassMethod AddNewCar(UserName As %String = "UserName Default",

UserId As %String = "UserId Default") [ PublicList = (MakeModel, Year, Color) ]  
{  
  Set CarOref=##class(MyPackage.Cars).%New() ;create a new object  
  
  If $G(MakeModel)="" { ;if MakeModel not already defined,  
   Read !,"Enter MakeModel: ",MakeModel ;accept MakeModel from user  
  }   
  If $G(Year)="" { ;if Year not already defined,  
   Read !,"Enter Year: ",Year ;accept Year from user  
  }  
  If $G(Color)="" { ;if Color not already defined,  
   Read !,"Enter Color: ",Color ;accept Color from user  
  }  
  
  Set CarOref.MakeModel=MakeModel ;Set MakeModel into object  
  Set CarOref.Year=Year ;Set Year into object  
  Set CarOref.Color=Color ;Set Color into object  
  Set CarOref.EnteredBy=UserName ;Set EnteredBy  
  Set CarOref.EnteredById=UserId ;Set EnteredId  
  
  Do CarOref.%Save()  
  Set Id=CarOref.%Id()  
  Write !,"New Object with Id: ",Id," Saved",!  
  Quit $$$OK  
}

Example ‑ Run AddNewCar Method

Kill ;Kill all local variables

Set MakeModel="Chevy"

Set Year="2005"

Set Color="Green"

Write ##class(MyPackage.Cars).AddNewCar("Jack Frost","12543")

New Object with ID: 4 Saved

1

Kill ;Kill all local variables

Set MakeModel="Ford"

Set Year="1"

Set Color="Cyan"

Write ##class(MyPackage.Cars).AddNewCar("","") ;Note the null parameters

New Object with ID: 5 Saved

1

Kill ;Kill all local variables

Set MakeModel="Dodge"

Set Year="1999"

Set Color="Purple"

Write ##class(MyPackage.Cars).AddNewCar("Jill Frost","95602")

New Object with ID: 6 Saved

1

Example ‑ Instance Method: AddNewCar

/// This method adds a New Car Object to the database.  
Method AddNewCar(UserName As %String = "UserName Default",

UserId As %String = "UserId Default") [ PublicList = (MakeModel, Year, Color) ]  
{  
  
  If $G(MakeModel)="" { ;if MakeModel not already defined,  
   Read !,"Enter New MakeModel: ",MakeModel ;accept MakeModel from user  
  }   
  If $G(Year)="" { ;if Year not already defined,  
   Read !,"Enter New Year: ",Year ;accept Year from user  
  }  
  If $G(Color)="" { ;if Color not already defined,  
   Read !,"Enter New Color: ",Color ;accept Color from user  
  }  
  
  Set ..MakeModel=MakeModel ;Set MakeModel into object  
  Set ..Year=Year ;Set Year into object  
  Set ..Color=Color ;Set Color into object  
  Set ..EnteredBy=UserName ;Set EnteredBy  
  Set ..EnteredById=UserId ;Set EnteredId  
  Set sc=..%Save()  
  Set Id=..%Id()  
  Write !,"New Object with Id: ",Id," Saved"  
  Quit $$$OK  
}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Set CarOref=##class(MyPackage.Cars).%New() ;create a new object, this needs

;to be done before the

;Instance Method is invoked.

Write CarOref.AddNewCar("Snow Frost","54545") ;When you call the Instance

;Method, use the newly created

;Oref as the base of your call

## 

# SQL Queries and Class Queries

Example ‑ Class MyPackage.Cars with SQL Query

Class MyPackage.Cars Extends %Persistent  
 {  
  
 Property MakeModel As %String;  
  
 Property Year As %Numeric;  
  
 Property Color As %String;  
  
 Property EnteredBy As %String;  
  
 Property EnteredById As %String;  
  
 Query DisplayAll() As %SQLQuery  
 {  
 Select \* From MyPackage.Cars  
 }

}

Example ‑ Running SQL Query ResultSet

Set ResultSet=##class(%ResultSet).%New("MyPackage.Cars:DisplayAll")  
 Set Status=ResultSet.Execute()

If Status=1 {  
  While ResultSet.Next() {  
 Write !!,"MakeModel: ",ResultSet.MakeModel  
 Write !,"Year: ",ResultSet.Year  
 Write !,"Color: ",ResultSet.Color  
 Write !,"Entered by: ",ResultSet.EnteredBy  
 Write !,"Entered by id: ",ResultSet.EnteredById  
   }

}

If you save the above code in a routine and then run it from the Terminal, you should get the following output.

MakeModel: Chevy

Year: 2001

Color: Yellow

Entered by: Jack Frost

Entered by id: 12543

MakeModel: Ford

Year: 2008

Color: Green

Entered by: Amy Frost

Entered by id: 43783

MakeModel: Dodge

Year: 2010

Color: Blue

Entered by: Jill Frost

Entered by id: 95602

MakeModel: Chevy

Year: 2005

Color: Green

Entered by: Jack Frost

Entered by id: 12543

Example ‑ Checking to ensure that the SQL Query is valid

Set ResultSet=##class(%ResultSet).%New()  
Set ResultSet.ClassName = "MyPackage.Cars" ;Query Class and Method  
Set ResultSet.QueryName = "DisplayAll" ;passed to Resultset  
  
Set QueryIsValid = ResultSet.QueryIsValid() ;validate the Query

If 'QueryIsValid {

; - do error reporting

Quit

}

 Set Status=ResultSet.Execute()

If Status=1 {  
  While ResultSet.Next() {  
 Write !!,"MakeModel: ",ResultSet.MakeModel  
 Write !,"Year: ",ResultSet.Year  
 Write !,"Color: ",ResultSet.Color  
 Write !,"Entered by: ",ResultSet.EnteredBy  
 Write !,"Entered by id: ",ResultSet.EnteredById  
   }

}

Example ‑ SQL Query with Input Parameters.

Class MyPackage.Cars Extends %Persistent  
 {  
  
 Property MakeModel As %String;  
  
 Property Year As %Numeric;  
  
 Property Color As %String;  
  
 Property EnteredBy As %String;  
  
 Property EnteredById As %String;  
  
 Query DisplayAll(InputMakeModel) As %SQLQuery  
 {

Select \* From MyPackage.Cars where MakeModel = :InputMakeModel

}

}

Example ‑ Running SQL Query with Input Parameters

Set ResultSet=##class(%ResultSet).%New()  
 Set ResultSet.ClassName = "MyPackage.Cars" ;Query Class and Method  
 Set ResultSet.QueryName = "DisplayAll" ;passed to Resultset

 Set Status=ResultSet.Execute("Ford") ; parameter Ford is input here

If Status=1 {  
  While ResultSet.Next() {  
 Write !!,"MakeModel: ",ResultSet.MakeModel  
 Write !,"Year: ",ResultSet.Year  
 Write !,"Color: ",ResultSet.Color  
 Write !,"Entered by: ",ResultSet.EnteredBy  
 Write !,"Entered by id: ",ResultSet.EnteredById  
   }

}

MakeModel: Ford

Year: 2008

Color: Green

Entered by: Amy Frost

Entered by id: 43783

Example ‑ Class Query structure

Class MyPackage.Cars Extends %Persistent  
 {  
  
 Property MakeModel As %String;  
  
 Property Year As %Numeric;  
  
 Property Color As %String;  
  
 Property EnteredBy As %String;  
  
 Property EnteredById As %String;

Query DisplayData(Input) As %Query(ROWSPEC ="MakeModel:%String,Year:%String,Color:%String")  
 {  
 }  
  
 ClassMethod DisplayDataExecute(ByRef qHandle As %Binary) As %Status  
 {  
 Quit $$$OK  
 }  
  
 ClassMethod DisplayDataClose(ByRef qHandle As %Binary) As %Status

[ PlaceAfter = DisplayDataExecute ]  
 {  
 Quit $$$OK  
 }  
  
 ClassMethod DisplayDataFetch(ByRef qHandle As %Binary, ByRef Row As %List,

ByRef AtEnd As %Integer = 0) As %Status [ PlaceAfter = DisplayDataExecute ]  
 {  
 Quit $$$OK  
 }

Example ‑ Class Query

Query DisplayData() As %Query(ROWSPEC = "MakeModel:%String,Year:%String,Color:%String")

[ SqlName = MyCars, SqlProc ]  
{  
}  
  
ClassMethod DisplayDataExecute(ByRef qHandle As %Binary) As %Status  
{  
    Set qHandle=0  
    Quit $$$OK  
}  
  
ClassMethod DisplayDataAllClose(ByRef qHandle As %Binary) As %Status

[ PlaceAfter = DisplayDataExecute ]  
{  
    Quit $$$OK  
}  
  
ClassMethod DisplayDataFetch(ByRef qHandle As %Binary, ByRef Row As %List,

ByRef AtEnd As %Integer = 0) As %Status [ PlaceAfter = DisplayDataExecute ]  
{  
   Set Id=qHandle ; qHandle is used to iterate through the objects  
 Set Id=Id+1 ;increment qHandle to get the next object  
 Set Oref=##class(MyPackage.Cars).%OpenId(Id) ;get next object  
 If '$IsObject(Oref) Set AtEnd=1,Row="" Quit $$$OK ;end of objects is reached  
 Set MakeModel=Oref.MakeModel  
 Set Year=Oref.Year  
   Set Color=Oref.Color  
   Set Row=$LB(MakeModel,Year,Color) ;Row must be $Listbuild  
   Set qHandle=Id ;Reset qHandle to get next Object  
   Quit $$$OK  
}}

Example ‑ ResultSet to run the Query

Set ResultSet=##class(%ResultSet).%New()  
 Set ResultSet.ClassName="MyPackage.Cars"  
 Set ResultSet.QueryName="DisplayData"

Set StatusCode=ResultSet.Execute()  
 If StatusCode'=1 Write "Invalid Status Code Returned" Quit

While ResultSet.%Next() {Do ResultSet.%Print() }

Chevy 2001 Yellow

Ford 2008 Green

Dodge 2010 Blue

Chevy 2005 Green

Ford 1 Cyan

Dodge 1999 Purple

*Customer: ”Do I need a computer to use your software?”*

*Tech Support: ”It helps”*

Example ‑ Write an External File

  ;Line1:

WriteFile ;  
  ;Line2:

Set Oref=##class(%File).%New("C:\FILE.TXT") ;create new Oref  
  ;Line3:

Do Oref.%Close() ;close the file  
 ;Line4:

Do Oref.Open("WSN",10) ;open the file  
 ;Line5:

If 'Oref.IsOpen Quit "0 – File not open" ;is the file open?  
 ;Line6:

Do Oref.WriteLine("First Record")  
 ;Line7:

Do Oref.WriteLine("Second Record")  
 ;Line8:

Do Oref.WriteLine("Third Record")  
 ;Line9:

Do Oref.WriteLine("Fourth Record")  
 ;Line10:

Do Oref.%Close()

;Line11;

Quit 1

Example ‑ Read a file and display its records

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

;Line1:

ReadFile ;

;Line2:  
 Set Oref=##class(%File).%New("C:\FILE.TXT") ;create Oref for file  
  ;Line3:   
 Do Oref.%Close() ;close file before opening it  
  ;Line4:   
 Do Oref.Open("R",10) ;open file for read, timeout 10 sec  
  ;Line5:

If 'Oref.IsOpen Quit "0 – File not open" ;file open?

  ;Line6:   
 Set InCount=0 ;init counter of records read  
  ;Line7:

  While 'Oref.AtEnd {

;Line8:  
 Set InRecord=Oref.ReadLine() ;read record from file

;Line9:  
 Set X=$Increment(InCount) ;increment counter

;Line10:  
 Write !,InRecord ;display record

;Line11:  
  }

  ;Line12:   
 Use 0 Write !,InCount," Records read"  
  ;Line13:   
 Use 0 Write !,"End of File reached"  
  ;Line14:   
 Quit 1

Example ‑ Running Routine ^ReadFile

Do ^ReadFile

First Record

Second Record

Third Record

Forth Record

4 Records read

End of File reached

Example ‑ Read and write a file

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

ReadAndWrite ;

Set InOref=##class(%File).%New("C:\FILE.TXT") ;create Oref for input file  
 Do InOref.%Close() ;close file before opening it  
 Do InOref.Open("R",10) ;open file for read  
 If 'InOref.IsOpen Quit  Quit "0 – Input File cannot be opened."   
 Set InCount=0 ;init counter of records read

Set OutOref=##class(%File).%New("C:\FILE2.TXT") ;create Oref output file

Do OutOref.%Close() ;close file before opening

Do OutOref.Open("WSN",10) ;open the file for writing

If 'OutOref.IsOpen Quit  Quit "0 – Output File cannot be opened."

Set OutCount=0 ;counter of records written

  While 'InOref.AtEnd {

Set InRecord=Oref.ReadLine() ;read record from file

Set X=$Increment(InCount) ;increment counter

Set X=$Increment(OutCount) ;increment counter

Do OutOref.WriteLine(InRecord) ;write out record

}

Use 0 Write !,InCount," Records read"

Use 0 Write !,OutCount," Records written"

Use 0 Write !,"End of File reached"

Quit

Example ‑ Running Routine ^ReadAndWrite

Do ^ReadAndWrite

First Record

Second Record

Third Record

Fourth Record

4 Records read

4 Records written

End of File reached

Example ‑ Cycle through several files

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

CycleThruFiles ;

For File="C:\FILE1.TXT","C:\FILE2.TXT","C:\FILE3.TXT" {

Set Oref=##class(%File).%New(File) ;create Oref for the file

Do Oref.Open("WSN",10) ;open file for read

Do Oref.WriteLine("Data") ;write data for the file

Do Oref.%Close() ;close the file

}

Set ResultSet=##class(%ResultSet).%New("%Library.File:FileSet")  
      
  Set sc=ResultSet.Execute("C:\","FILE\*.TXT") ;execute the Query  
    If $SYSTEM.Status.IsError(sc) {  
        Quit "0 - Error on Execute Query"  
    }  
  
    While ResultSet.Next() { ;return the data  
  Set FileName=ResultSet.Data("Name")

Write !,FileName

  }

Example ‑ Running Routine ^CycleThruFiles

Do ^CycleThruFiles

C:\FILE.TXT

C:\FILE1.TXT

C:\FILE2.TXT

C:\FILE3.TXT

Example ‑ Search multiple files for a specific string

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

SearchForString  
  
 For File="C:\FILE1.TXT","C:\FILE3.TXT","C:\FILE5.TXT" { ;create files  
 Set Oref=##class(%File).%New(File) ;create Oref for the file

Do Oref.Open("WSN",10) ;open file for read

Do Oref.%Close() ;close the file

}  
 For File="C:\FILE2.TXT","C:\FILE4.TXT","C:\FILE6.TXT" { ;files - "fleas"  
 Set Oref=##class(%File).%New(File) ;create Oref for the file

Do Oref.Open("WSN",10) ;open file for read

Do Oref.WriteLine("My dog has fleas") ;write data for the file

Do Oref.%Close() ;close the file

}  
  
 Set ResultSet=##class(%ResultSet).%New("%Library.File:FileSet")  
      
  Set sc=ResultSet.Execute("C:\","FILE\*.TXT") ;execute the Query  
    If $SYSTEM.Status.IsError(sc) {  
        Quit "0 - Error on Execute Query"  
    }  
  
    While ResultSet.Next() { ;return the data  
      Set FileName=ResultSet.Data("Name") ;get filename  
     Set Oref=##class(%File).%New(FileName) ;establish new Oref  
     Do Oref.Open("R",10) ;open file  
     If 'Oref.IsOpen Continue ;file not open?  
      While 'Oref.AtEnd {  
 Set InRecord=Oref.ReadLine() ;read record   
 If InRecord["fleas" { ;file contains "fleas"  
 Use 0 Write !,"File: "  
 Write FileName," contains string 'fleas'."  
 }   
 }  
 }

Example ‑ Running Routine ^SearchForString

Do ^SearchForString

File: C:\FILE2.TXT contains string 'fleas'.

File: C:\FILE4.TXT contains string 'fleas'.

File: C:\FILE6.TXT contains string 'fleas'.

Example ‑ Search multiple files in multiple directories for a specific string

To run this code you must put the code in a routine, save the routine, and then run the routine from the terminal.

SearchForString2  
  
 ; Creating file for subdirectory C:\SUBDIR\  
   Do ##class(%Library.File).CreateDirectory("C:\SUBDIR\")  
 For File="FILE1.TXT","FILE3.TXT","FILE5.TXT" {  
     Set Oref=##class(%File).%New("C:\SUBDIR\"\_File) ;create Oref for file  
     Do Oref.%Close() ;close file before opening it  
     Do Oref.Open("WSN",10) ;open file for write  
     If 'Oref.IsOpen Continue   
     Do Oref.WriteLine("Data") ;write data for the file  
     Do Oref.%Close()   
   }  
 For File="FILE2.TXT","FILE4.TXT","FILE6.TXT" {  
     Set Oref=##class(%File).%New("C:\SUBDIR\"\_File) ;create Oref for file  
     Do Oref.%Close() ;close file before opening it  
     Do Oref.Open("WSN",10) ;open file for write  
     If 'Oref.IsOpen Continue   
     Do Oref.WriteLine("My dog has fleas") ;write data for the file  
     Do Oref.%Close()   
 }  
  
 ; Creating file for subdirectory C:\SUBDIR\SUB2DIR\  
   Do ##class(%Library.File).CreateDirectory("C:\SUBDIR\SUB2DIR")  
 For File="FILE1.TXT","FILE3.TXT","FILE5.TXT" {  
     Set Oref=##class(%File).%New("C:\SUBDIR\SUB2DIR\"\_File)  
     Do Oref.%Close() ;close file before opening it  
     Do Oref.Open("WSN",10) ;open file for write  
     If 'Oref.IsOpen Continue   
     Do Oref.WriteLine("Data") ;write data for the file  
     Do Oref.%Close()  
  }  
 For File="FILE2.TXT","FILE4.TXT","FILE6.TXT" {  
     Set Oref=##class(%File).%New("C:\SUBDIR\SUB2DIR\"\_File)  
     Do Oref.%Close() ;close file before opening it  
     Do Oref.Open("WSN",10) ;open file for read  
     If 'Oref.IsOpen Continue   
     Do Oref.WriteLine("My dog has fleas") ;write data for the file  
     Do Oref.%Close()   
 }  
      
  Do MultiLevelSearch("C:\SUBDIR\")  
  Quit  
  
MultiLevelSearch(Dir)  
 Set ResultSet=##class(%ResultSet).%New("%Library.File:FileSet")  
  Set sc=ResultSet.Execute(Dir,"") ;execute the Query  
    If $SYSTEM.Status.IsError(sc) {  
        Quit "0 - Error on Execute Query"  
    }  
  
    While ResultSet.Next() { ;return the data  
      Set FileName=ResultSet.Data("Name") ;get filename  
      Set FileType=ResultSet.Data("Type")  
     If FileType="D" Do MultiLevelSearch(FileName) ;"D" means directory  
     Set Oref=##class(%File).%New(FileName) ;establish new Oref  
     Do Oref.Open("R",10) ;open file  
     If 'Oref.IsOpen Continue ;file not open?  
      While 'Oref.AtEnd {  
 Set InRecord=Oref.ReadLine() ;read record   
 If InRecord["fleas" { ;file contains "fleas"  
    Use 0 Write !,"File: "  
    Write FileName," contains string 'fleas'."  
 }   
     }  
 }

Example ‑ Running Routine ^SearchForString2

Do ^SearchForString2

File: C:\SUBDIR\FILE2.TXT contains string 'fleas'.

File: C:\SUBDIR\FILE4.TXT contains string 'fleas'.

File: C:\SUBDIR\FILE6.TXT contains string 'fleas'.

File: C:\SUBDIR\SUB2DIR\FILE2.TXT contains string 'fleas'.

File: C:\SUBDIR\SUB2DIR\FILE4.TXT contains string 'fleas'.

File: C:\SUBDIR\SUB2DIR\FILE6.TXT contains string 'fleas'.

File: C:\SUBDIR\SUB2DIR\FILE6.TXT contains string 'fleas'.

Example ‑ Retrieve date information about a File

Set Oref=##class(%File).%New("C:\FILE.TXT")

Do Oref.WriteLine("Rec1")

Do Oref.WriteLine("Rec2")

Do Oref.WriteLine("Rec3")

Do Oref.Close()

Set File="C:\FILE.TXT"

Set CreateDate=##Class(%File).GetFileDateCreated(File)

Write $Zdatetime(CreateDate)

Set ModifiedDate=##Class(%File).GetFileDateModified(File)

Write $Zdatetime(ModifiedDate)

Example ‑ Check on the existence of a File

Write ##class(%Library.File).Exists("C:\FILE.TXT") ;Return 1, file exists

1

Write ##class(%Library.File).Exists("C:\FILExxx.TXT") ;Returns 0, file does not exist

0

Example ‑ Is the File Writeable?

Write ##Class(%Libarary.File).Writeable("C:\FILE.TXT") ;Return 1,file writeable

1

Example ‑ Copying a file

Set File="C:\FILE.TXT"

Set NewFile="C:\NEWFILE.TXT"

Write ##Class(%Library.File).CopyFile(File,NewFile) ;Return 1, Copy successful

1

Example ‑ Renaming a file

Set File="C:\NEWFILE.TXT"

Set NewFile="C:\NEWFILE2.TXT"

Write ##Class(%Library.File).Rename(File,NewFile) ;Return 1, Rename successful

1

Example ‑ Delete a File

Set NewFile="C:\NEWFILE2.TXT"

Write ##Class(%Library.File).Delete(NewFile) ;Return 1, Delete successful

1

*Tech Support: “I need you to boot the computer.”*

*Customer: (THUMP! Pause.) “No, that didn't help.”*

# Caché ObjectScript – Object Commands Reference Table

This chapter lists some of the *Object Commands* for quick reference.

## General Help

|  |  |
| --- | --- |
| General Help on Object Calls | Do $system.OBJ.Help() |
|  | Do $system.OBJ.Help(method) |
| General Help on Version Calls | Do $system.Version.Help() |
|  | Do $system.Version.Help(method) |
| General Help on SQL Calls | Do $system.SQL.Help() |
|  | Do $system.SQL.Help(method) |

## Calling a Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Calling a Class Method | |  | | --- | | Do ##class(package.class).method(params) | | Write ##class(package.class).method(params) | | Set Status=##class(package.class).method(params) | |
| Create a new Oref (Object Reference) | Set Oref=##class(package.class).%New() |
| Opening an existing Object | Set Oref=##class(package.class).%OpenId(Id) |
| Calling an Instance Method | |  | | --- | | Do oref.method(params) | | Write oref.method(params) | | Set var=oref.method(params) | |

## Save and Delete Calls

|  |  |
| --- | --- |
| Save an object | Set status=oref.%Save() |
| Delete an existing object | Set status=##class(package.class).%DeleteId(Id) |
| Delete all saved objects (warning – this commands will kill the entire global, use with caution) | Set status=##class(package.class).%DeleteExtent() |

## Status Calls

|  |  |
| --- | --- |
| Return a good status | Quit $$$OK |
| Return an error status | Quit $$$ERROR($$$GeneralError,message) |
| Check if good status | If $$$ISOK(status) |
| Check if error status | If $$$ISERR(status) |
| Print the status (after an error) | |  | | --- | | Do $system.Status.DisplayError(status) | | Do $system.Status.DecomposeStatus(status) | |

## Validate Calls

|  |  |
| --- | --- |
| Validate a string item | If ##class(%Library.String).IsValid(dataitem) |
| Validate an numeric item | If ##class(%Library.Numeric).IsValid(dataitem) |
| Validate an integer item | If ##class(%Library.Integer).IsValid(dataitem) |
| Validate a time item | If ##class(%Library.Time).IsValid(dataitem) |
| Validate a date item | If ##class(%Library.Date).IsValid(dataitem) |
| Ensure an Id exists | |  | | --- | | Write ##class(package.class).%ExistsId(Id) | | Write Oref.%ExistsId(Id) | |
| Ensure an Oref exists | If $IsObject(Oref) |

## Link Objects Call

|  |  |  |  |
| --- | --- | --- | --- |
| Link two properties together | |  | | --- | | Set oref1.property=oref2 | | Set oref2.property=oref1 | |

## Tests Calls

|  |  |
| --- | --- |
| Test whether a class exists | If ##class(%Dictionary.ClassDefinition).  %Exists($LB("package.classname")) |
| Test whether an object is valid | If $IsObject(oref) |

## Obtain a Value Calls

|  |  |
| --- | --- |
| Obtain a property's value | Set value=oref.property |
| Obtain the Id of a saved object | Set Id=oref.%Id() |

## Set Calls

|  |  |
| --- | --- |
| Set a Property to a value | Set oref.property=value |

## Populate Call

|  |  |  |
| --- | --- | --- |
| Populate a class | Set =##class(package.class).Populate(Num,{1,0})  Where  -Num=number of items  -1 for Verbose, 0 for not Verbose   |  | | --- | | Note: the class needs to extend %Populate and the affected properties need POPSPEC parameters. | |

## List/Display/Dump Calls

|  |  |
| --- | --- |
| List all objects in memory | Do $system.OBJ.ShowObjects() – pass "D" for details |
| Display an Oref | Do $system.OBJ.Dump(oref) |

## System/Product/Version Calls

|  |  |
| --- | --- |
| See what system you are on | Write $system.Version.GetBuildOS() |
| See what product version you are running | Write $system.Version.GetProduct() |
| Return the version number of the current object library | Write $system.OBJ.Version() |

## General Miscellaneous Calls

|  |  |
| --- | --- |
| View Class Name (must be inside a class) | Write $CLASSNAME |
| View Current Date/Time | Write $NOW() - 62416,35449.664935  Write $system.SYS.Horolog() - 62417,22848  Write $system.SYS.TimeStamp() - 62416,53709.534 (GMT) |
| View UserName | Write $USERNAME |
| View Platform | Write $system.Version.GetPlatform() |
| View Operating System | Write $system.Version.GetOS() |
| View Process Id | Write $system.SYS.ProcessID() |
| View Namespace | Write $system.SYS.NameSpace() |
| View Time Zone (delta time in minutes from GMT) | Write $system.SYS.TimeZone() |

## Chapter 35 Summary

This chapter lists some of the Object Commands for quick reference.

# Caché ObjectScript - Good Programming Concepts

*Tech Support: "Type 'fix' with an 'f'."*

*Customer: "Is that 'f' as in 'fix'?"*

*“The man who smiles when things go wrong has thought of someone he can blame it on.”*