IBM Developer for z Systems – Experienced Training

Reusable Code – Concepts and functionality for developing reusable code libraries using Snippets and Templates

DevOps

Jon Sayles, IBM - jsayles@us.ibm.com

Last Revision Date: May 2021

IBM Trademarks and Copyrights

Course Assumptions

1. You know ISPF and have used it for at least two years, doing production z/OS work in COBOL, PL/I or Assembler
   - Note that all of the workshops in this course are in **COBOL** – although files exist that are Assembler, PL/I, REXX and other languages for you to experiment with – time permitting

2. You have:
   - Experience with Eclipse or IDz
     - Formal Training
     - And/or at least 6 months of production use
   - IDz installed and running on your workstation at version 14 or later

- Note that all ISPF discussion/examples and screen captures assume IBM-installed ISPF product defaults – not any 3rd party or custom Dialog Manager applications you may have installed on your mainframe

UNIT

Reusable Code

Topics in this module:

- Reusable Code: Terms and Concepts
- Snippets
- Language Templates
- Program Templates

What is Code Reuse?

Reusing pre-written parts of a program (COBOL, PL/I, Assembler, Easytrieve, REXX. etc.), JCL file, MVS utility, BMS/MFS screen, etc. in the construction of other programs/utilities/etc.

Production code reuse has long been an objective for software development architects. An excellent in-depth treatise on Code Reuse cam be found here: https://en.wikipedia.org/wiki/Code_reuse

Most of us have bought into code reuse and throughout our careers as z/OS developers begin new programming assignments by

1. Hunting down an existing program that is similar to the work under construction
2. Copying the entire source file
3. Cannibalizing the source code in the file:
    1. Deleting most of the PROCEDURE and DATA DIVISION
    2. Editing the ENVIRONMENT and IDENTIFICATION DIVISION

Why Code Reuse?

Code reuse can save time and resources and reduce redundancy by taking advantage of assets that have already been created in some form within the software product development process

Productivity:

- Reusing working/syntactically-correct code takes less time to develop, and less time to test

Consistency/Standards Conformance/Maintain-ability:

- Maintenance and Support costs are lowered, as reusable code consists of recognizable patterns

Education (new to z/OS):

- By leveraging a catalog of reusable functions, new-to-z/OS developers can be given working examples of arcane language, statements and code patterns… accelerating time to mastery

Code Quality:

- Reusable code will be - or should be syntactically-correct, and well-tested

Problems with Code Reuse include the:

- Possible inability to tweak details which may affect performance or the desired output
- Time and cost of acquiring, learning, and configuring the library

Examples of Code Reuse - Terms & Concepts

Software libraries

- Common operations, such as:
  - Accessing external storage
  - Interfacing with external programs
  - Manipulating information (numbers, words, names, locations, dates, etc.) in common ways, are needed by many different programs.

Design patterns: https://www.geeksforgeeks.org/software-design-patterns/

- A design pattern is a general solution to a recurring problem

Frameworks

- Class-based language (Java, C++, .Net,) developers often reuse large pieces of software via third-party applications and frameworks

Functional Decomposition

- In modular-development higher-order functions can be used in many cases where design patterns or frameworks were formerly used

Components and reusable patterns:

- Embedded SQL Cursor
- Control Break logic
- Master-File Update logic

These terms & definitions are strictly defined in Computer Science. However in practice they are used with a lot less "rigor"

Five Categories of Code Reuse

1. Miscellaneous Copy/Paste operations
2. Project-Level reusable code
3. Enterprise-Level standard code
4. Example statements
5. Entry-Level Training

Note that the above is a simplistic and subjective breakdown

IDz's Code Reuse Options

Snippets

- Flexible and simple method of code reuse
- Snippet scope can be from anywhere from a keyword to an entire program
- Can define any number of custom variables to manage idiosyncratic requirements
- Can export/import Snippets with Workspace
- Can include Snippets view in custom Perspective

Language-specific Code Templates

- Most granular form of code reuse
- Integrates with Content Assist
- Typically used for statements – but could extend to more code

Program Templates – available in COBOL and PL/I

- Useful if creating a new COBOL or PL/I program using the **New program** wizard
- IDz shops with standard (not customized) CICS and SQL statements
- Can be customized

Code Snippets

Sometimes, instead of entire programs you might want to:

- Save some code temporarily for reuse – similar to the ISPF: "CREATE" and "COPY" command line commands
- Create a paragraph, computation, complex conditional – that can be re-purposed in other programs
- Provide a library of "standardized - Best Practices" routines – using your shop's coding conventions
- Provide a library of syntactically-correct and infrequently used/high-value statements:
  - Job Cards
  - Database routines
  - Complex COBOL code: UNSTRING etc.

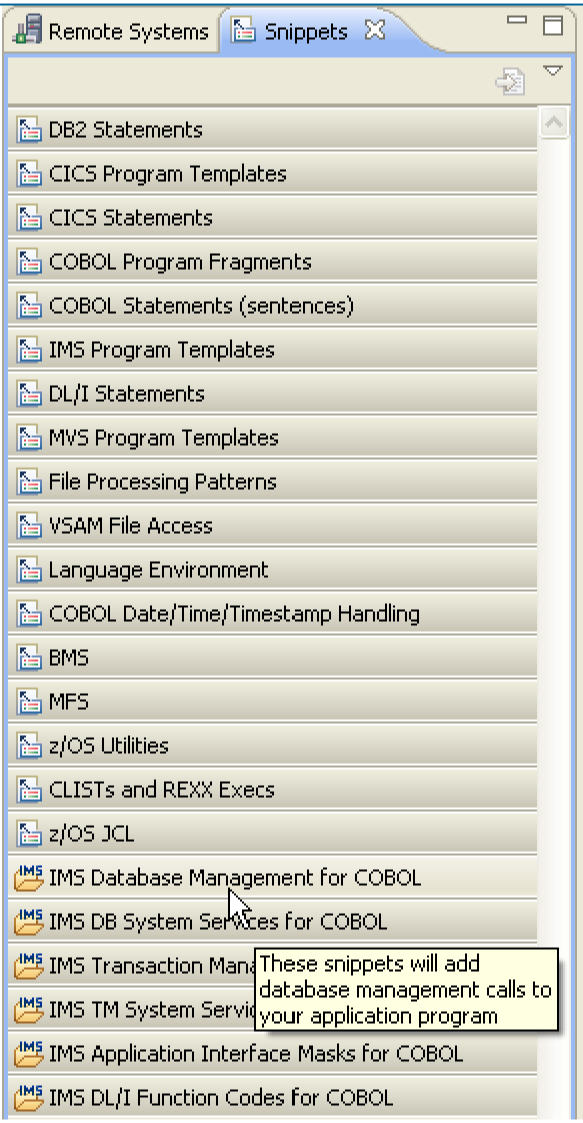
Snippets are the often the best way of doing this. You access them through a Snippets view, which you get to by:

- From Window > Show View > other…
- Type: snippets – and select the **Snippets** view

On the right are a group of custom Snippets that we have created. You will see a subset of these in your workspace.

Individual Snippets are contained in "drawers" which are the accordion menus that collapse/expand on-click.

Snippets can be Exported/Imported (for sharing)

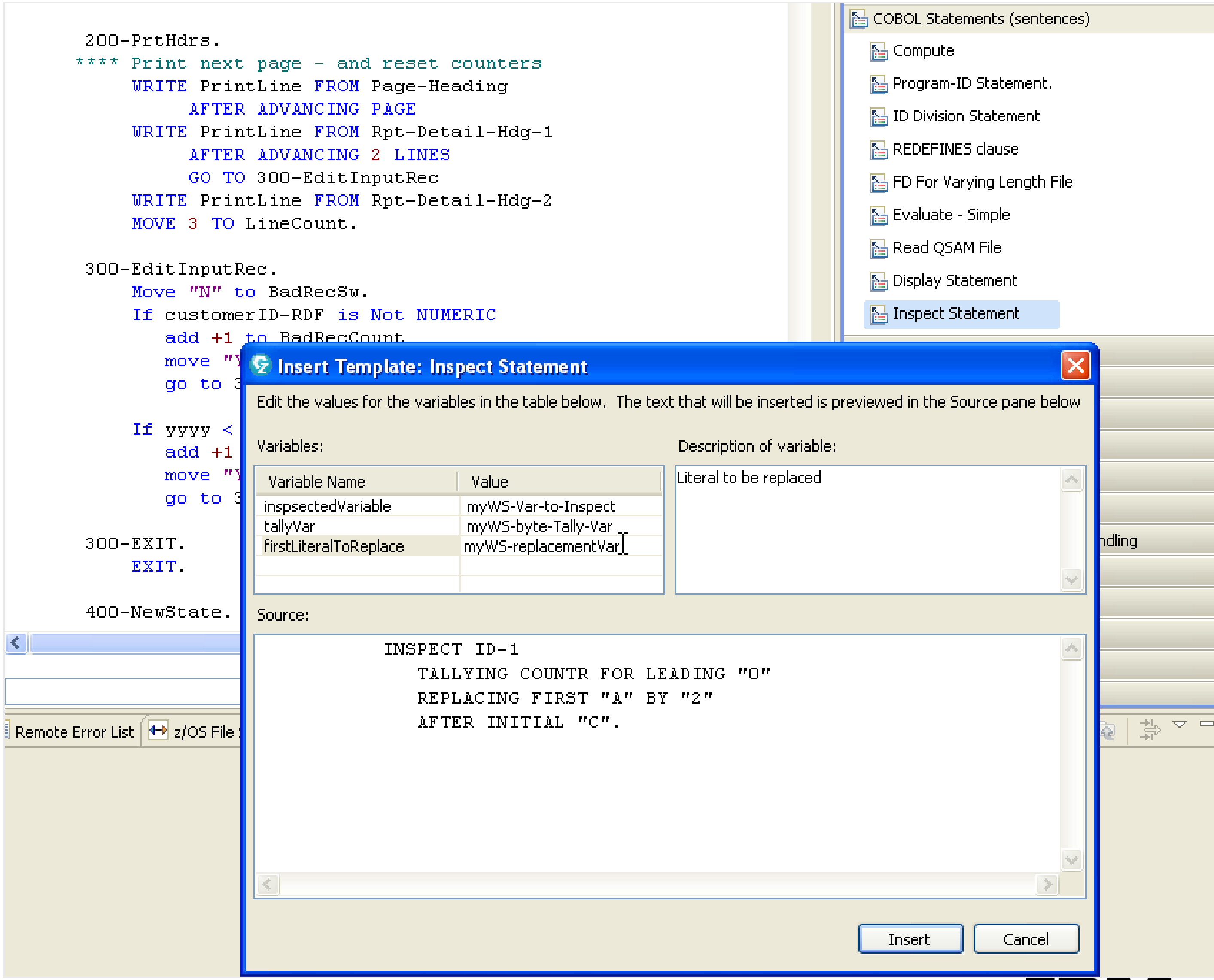


These snippets will add database management calls to your application program

Using Code Snippets (ISPF "COPY" command line command)

To use an existing code Snippet follow the steps below:

1. Place your cursor at the exact focal point (position in the source) where you want a code snippet inserted
2. Find your Code Snippet in the snippet drawers
3. Double-Click the Snippet
4. If there are variables in the snippet, you can:
   - **Accept the defaults**
   - **Over-ride the values before the code is inserted**
5. Click Insert



```
 200-PrtHdrs.
**** Print next page - and reset counters
     WRITE PrintLine FROM Page-Heading
           AFTER ADVANCING PAGE
     WRITE PrintLine FROM Rpt-Detail-Hdg-1
           AFTER ADVANCING 2 LINES
           GO TO 300-EditInputRec
     WRITE PrintLine FROM Rpt-Detail-Hdg-2
     MOVE 3 TO LineCount.

 300-EditInputRec.
     Move "N" to BadRecSw.
     If customerID-RDF is Not NUMERIC
        add +1 to BadRecCount
        move "Y
        go to 3

     If yyyy <
        add +1
        move "Y
        go to 3

 300-EXIT.
     EXIT.

 400-NewState.
```

Edit the values for the variables in the table below. The text that will be inserted is previewed in the Source pane below

Variables:

Variable Name	Value
inspsectedVariable	myWS-Var-to-Inspect
tallyVar	myWS-byte-Tally-Var
firstLiteralToReplace	myWS-replacementVar

Description of variable:

Literal to be replaced

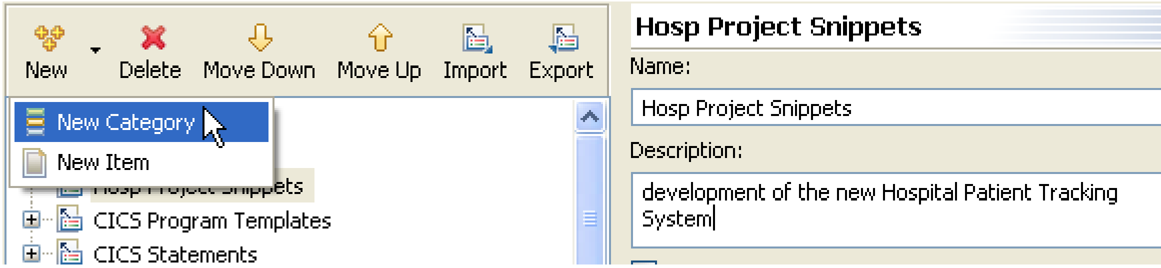
Source:

```
           INSPECT ID-1
              TALLYING COUNTR FOR LEADING "0"
              REPLACING FIRST "A" BY "2"
              AFTER INITIAL "C".
```

Creating Code Snippets - 1 of 3

To create a new code Snippet follow the steps below:

1. Create a new Snippet category
   - **Right-click over the Snippets view**
   - **Select Customize**
   - **From Customize Palette, under New**
     Select: New Category
   - **Name the Category**
   - **Add a description**
   - **Click OK**

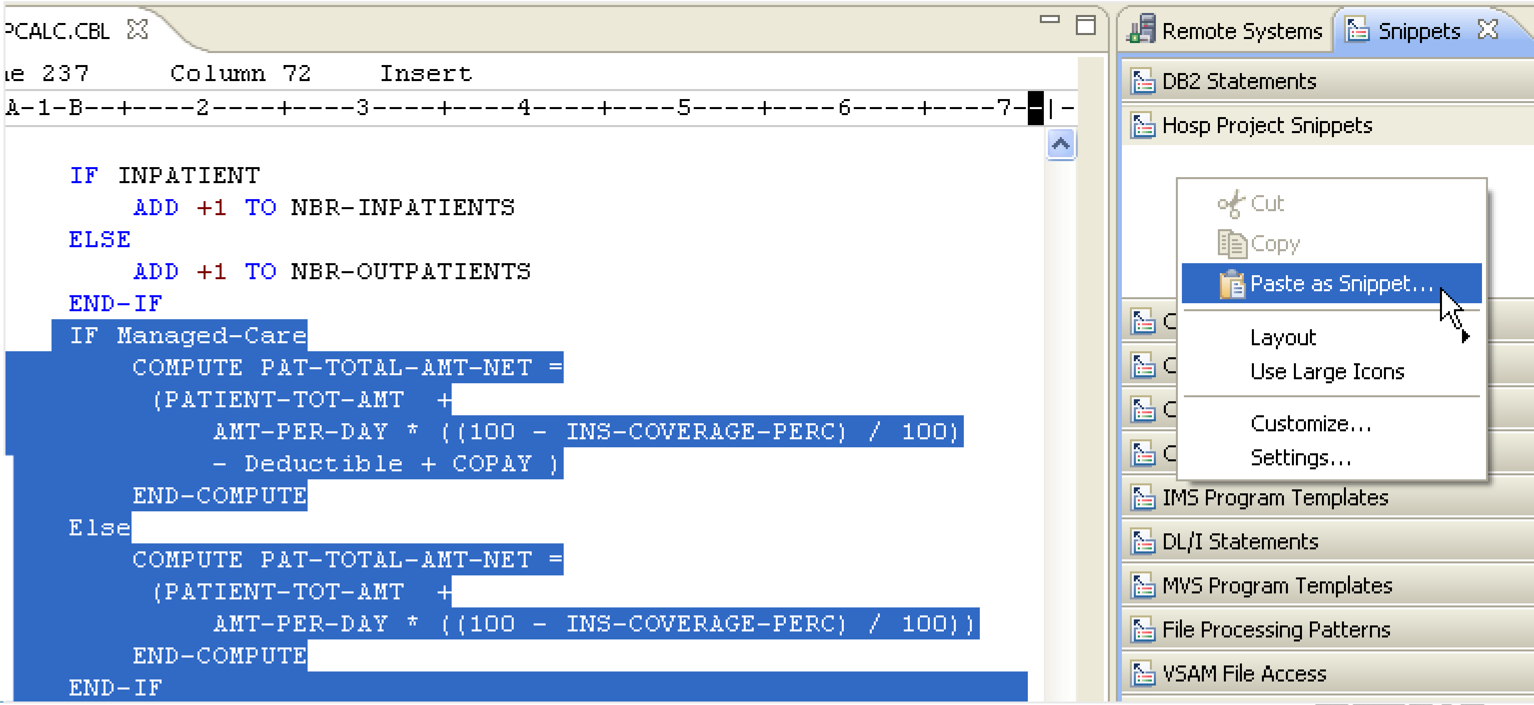


Hosp Project Snippets

Name:
Hosp Project Snippets

Description:
development of the new Hospital Patient Tracking System

2. Select and copy the code you wish to turn into a Snippet
3. Expand the category you wish to add the Snippet to, and select Paste as Snippet…



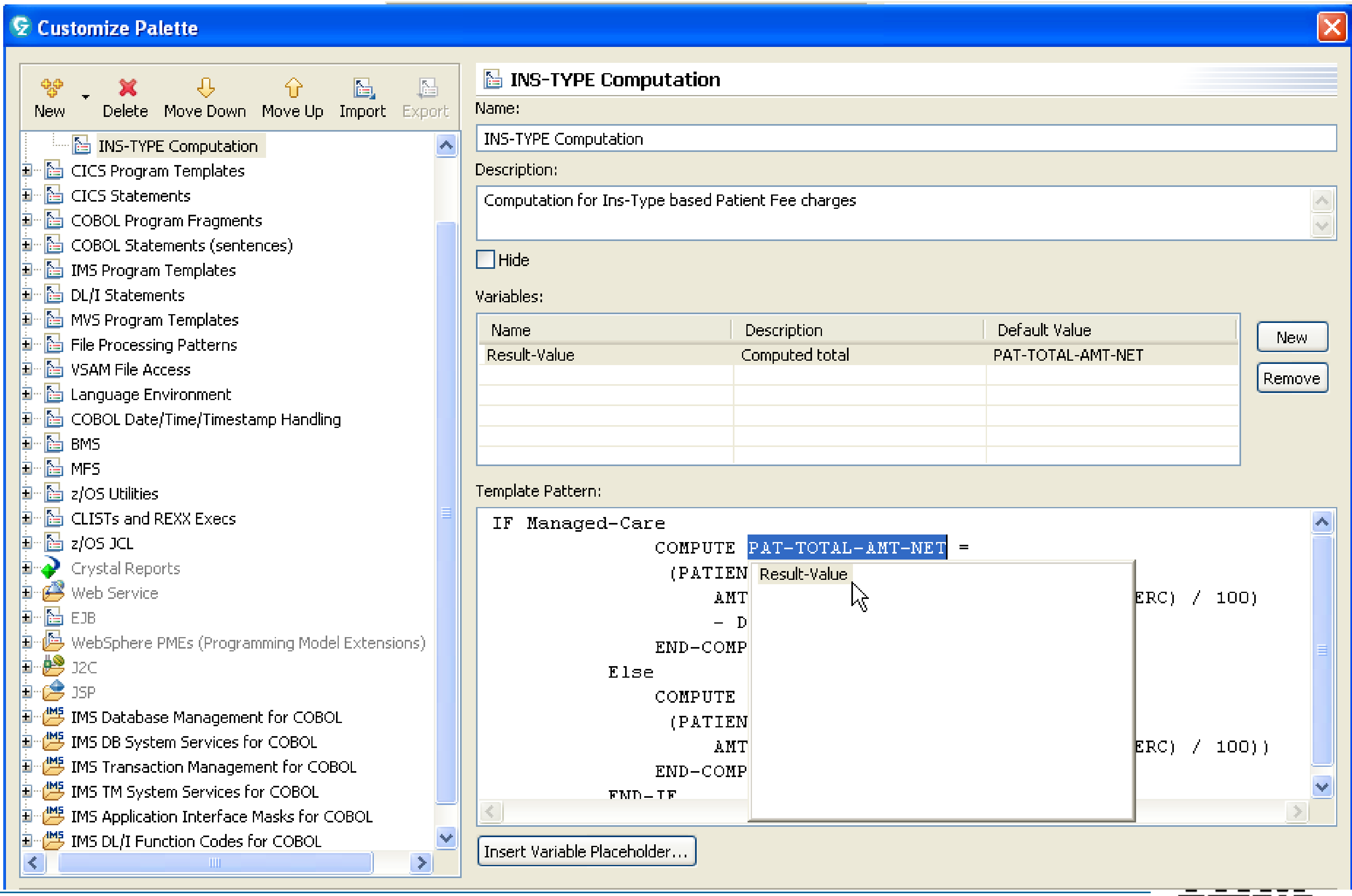
```
ne 237      Column 72     Insert
A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-

       IF  INPATIENT
           ADD +1 TO NBR-INPATIENTS
       ELSE
           ADD +1 TO NBR-OUTPATIENTS
       END-IF
       IF Managed-Care
           COMPUTE PAT-TOTAL-AMT-NET =
             (PATIENT-TOT-AMT   +
                 AMT-PER-DAY * ((100 - INS-COVERAGE-PERC) / 100)
                 - Deductible + COPAY )
           END-COMPUTE
       Else
           COMPUTE PAT-TOTAL-AMT-NET =
             (PATIENT-TOT-AMT   +
                 AMT-PER-DAY * ((100 - INS-COVERAGE-PERC) / 100))
           END-COMPUTE
       END-IF
```

Note that these steps are similar to an ISPF "CREATE" Command Line command

Creating Code Snippets – 2 of 3

4. Rename the Snippet and give it a Description
5. Optionally add Variables to be filled in by Snippet users (or they can accept the defaults)



Customize Palette

New | Delete | Move Down | Move Up | Import | Export

- INS-TYPE Computation
- CICS Program Templates
- CICS Statements
- COBOL Program Fragments
- COBOL Statements (sentences)
- IMS Program Templates
- DL/I Statements
- MVS Program Templates
- File Processing Patterns
- VSAM File Access
- Language Environment
- COBOL Date/Time/Timestamp Handling
- BMS
- MFS
- z/OS Utilities
- CLISTs and REXX Execs
- z/OS JCL
- Crystal Reports
- Web Service
- EJB
- WebSphere PMEs (Programming Model Extensions)
- J2C
- JSP
- IMS Database Management for COBOL
- IMS DB System Services for COBOL
- IMS Transaction Management for COBOL
- IMS TM System Services for COBOL
- IMS Application Interface Masks for COBOL
- IMS DL/I Function Codes for COBOL

INS-TYPE Computation

Name: INS-TYPE Computation

Description: Computation for Ins-Type based Patient Fee charges

Hide

Variables:

Name	Description	Default Value
Result-Value	Computed total	PAT-TOTAL-AMT-NET

New | Remove

Template Pattern:

```
IF Managed-Care
          COMPUTE PAT-TOTAL-AMT-NET =
           (PATIEN Result-Value
              AMT                        ERC) / 100)
              - D
          END-COMP
       Else
          COMPUTE
           (PATIEN
              AMT                        ERC) / 100))
          END-COMP
       END-IF
```

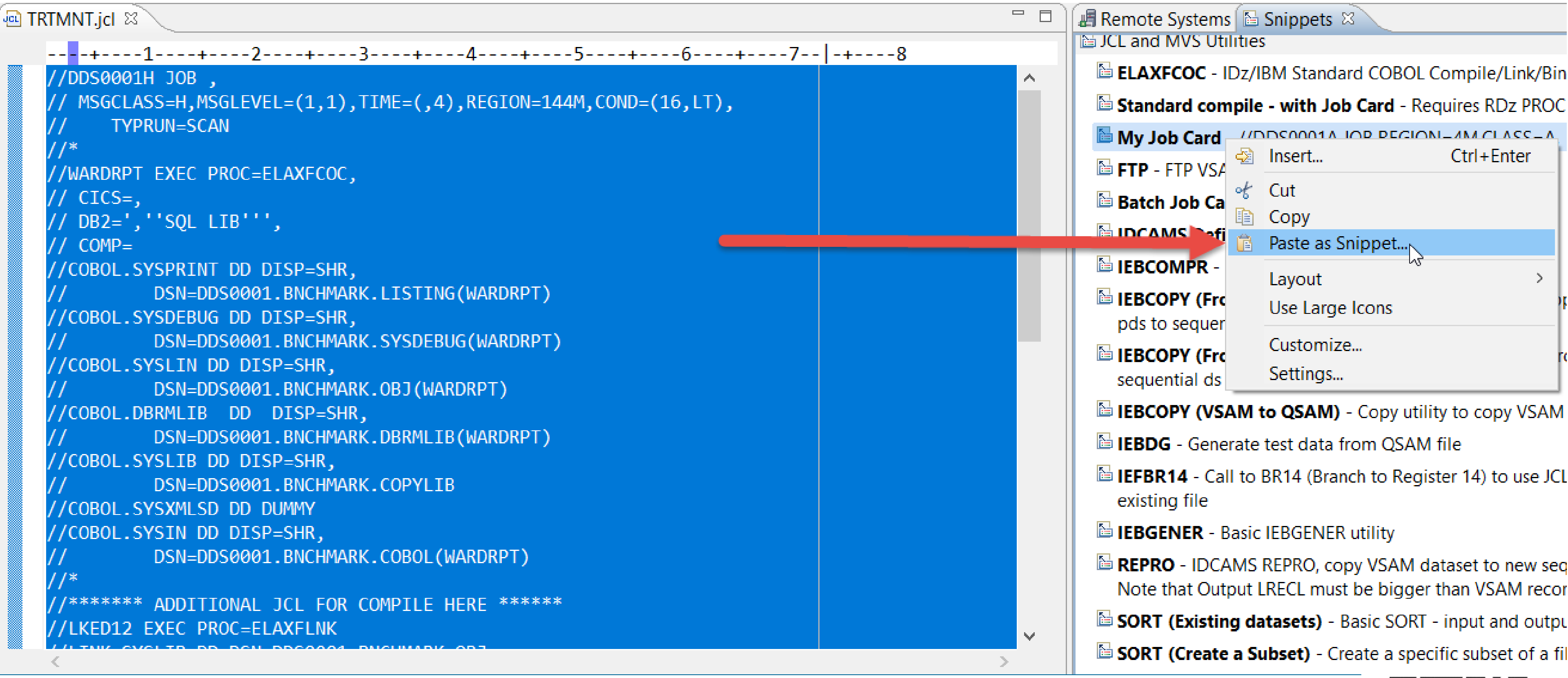
Insert Variable Placeholder...

Creating Code Snippets - 3 of 3

You can also create Code Snippets by:

- **Select & Copy the code** you want to turn into a Snippet
- **Right-Click** over an existing Snippet drawer
- Select **Paste as Snippet…**

This will create a new Snippet Item at the top of the Snippet Drawer you've right-clicked over



TRTMNT.jcl

```
---+----1----+----2----+----3----+----4----+----5----+----6----+----7--|-+----8
//DDS0001H JOB ,
// MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M,COND=(16,LT),
//    TYPRUN=SCAN
//*
//WARDRPT EXEC PROC=ELAXFCOC,
// CICS=,
// DB2=',''SQL LIB''',
// COMP=
//COBOL.SYSPRINT DD DISP=SHR,
//        DSN=DDS0001.BNCHMARK.LISTING(WARDRPT)
//COBOL.SYSDEBUG DD DISP=SHR,
//        DSN=DDS0001.BNCHMARK.SYSDEBUG(WARDRPT)
//COBOL.SYSLIN DD DISP=SHR,
//        DSN=DDS0001.BNCHMARK.OBJ(WARDRPT)
//COBOL.DBRMLIB  DD  DISP=SHR,
//        DSN=DDS0001.BNCHMARK.DBRMLIB(WARDRPT)
//COBOL.SYSLIB DD DISP=SHR,
//        DSN=DDS0001.BNCHMARK.COPYLIB
//COBOL.SYSXMLSD DD DUMMY
//COBOL.SYSIN DD DISP=SHR,
//        DSN=DDS0001.BNCHMARK.COBOL(WARDRPT)
//*
//******* ADDITIONAL JCL FOR COMPILE HERE ******
//LKED12 EXEC PROC=ELAXFLNK
```

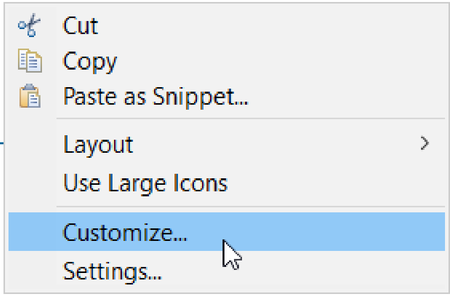
Remote Systems | Snippets

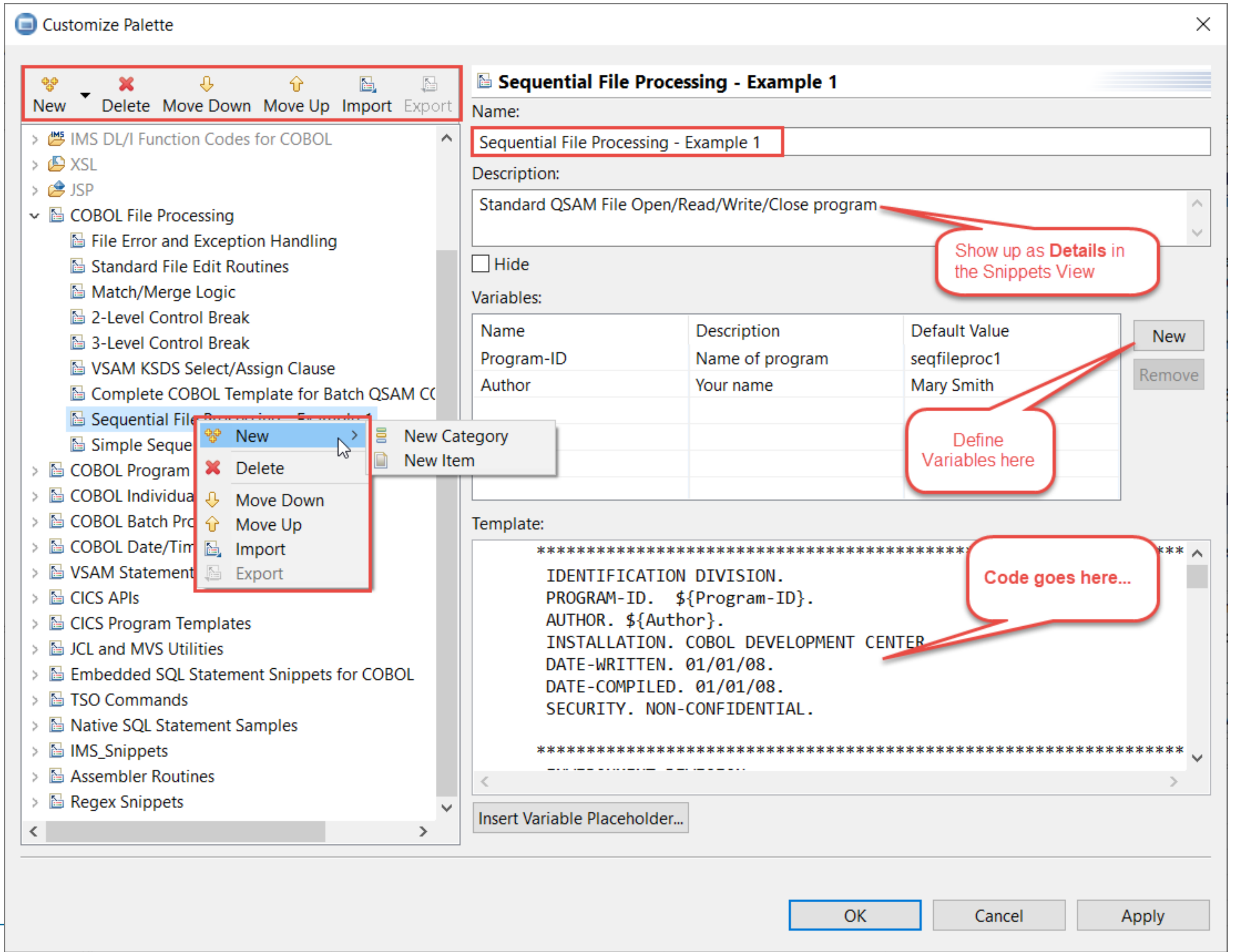
JCL and MVS Utilities

- **ELAXFCOC** - IDz/IBM Standard COBOL Compile/Link/Bin
- **Standard compile - with Job Card** - Requires RDz PROC
- **My Job Card** - //DDS0001A JOB REGION=4M,CLASS=A
- **FTP** - FTP VSA
- **Batch Job Ca**
- **IDCAMS Defi**
- **IEBCOMPR** -
- **IEBCOPY (Fro** pds to sequer
- **IEBCOPY (Fro** sequential ds
- **IEBCOPY (VSAM to QSAM)** - Copy utility to copy VSAM
- **IEBDG** - Generate test data from QSAM file
- **IEFBR14** - Call to BR14 (Branch to Register 14) to use JCL existing file
- **IEBGENER** - Basic IEBGENER utility
- **REPRO** - IDCAMS REPRO, copy VSAM dataset to new seq Note that Output LRECL must be bigger than VSAM recor
- **SORT (Existing datasets)** - Basic SORT - input and outpu
- **SORT (Create a Subset)** - Create a specific subset of a fil

The Customize Palette Dialog

Options to define new Drawers, New Snippets, Modify Drawers & Snippets, Import, Export, etc.





Sequential File Processing - Example 1

Name: Sequential File Processing - Example 1

Description: Standard QSAM File Open/Read/Write/Close program

[ ] Hide

Variables:

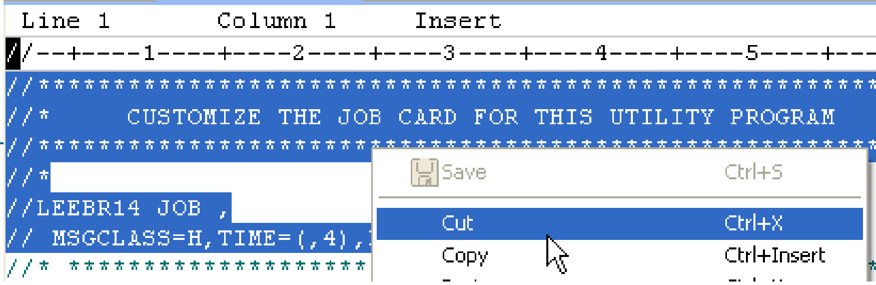
Name	Description	Default Value
Program-ID	Name of program	seqfileproc1
Author	Your name	Mary Smith

Template:

```
      ********************************************************************
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  ${Program-ID}.
       AUTHOR. ${Author}.
       INSTALLATION. COBOL DEVELOPMENT CENTER.
       DATE-WRITTEN. 01/01/08.
       DATE-COMPILED. 01/01/08.
       SECURITY. NON-CONFIDENTIAL.

      ********************************************************************
```

Example: Create and Use a Code Snippet for a Job Card

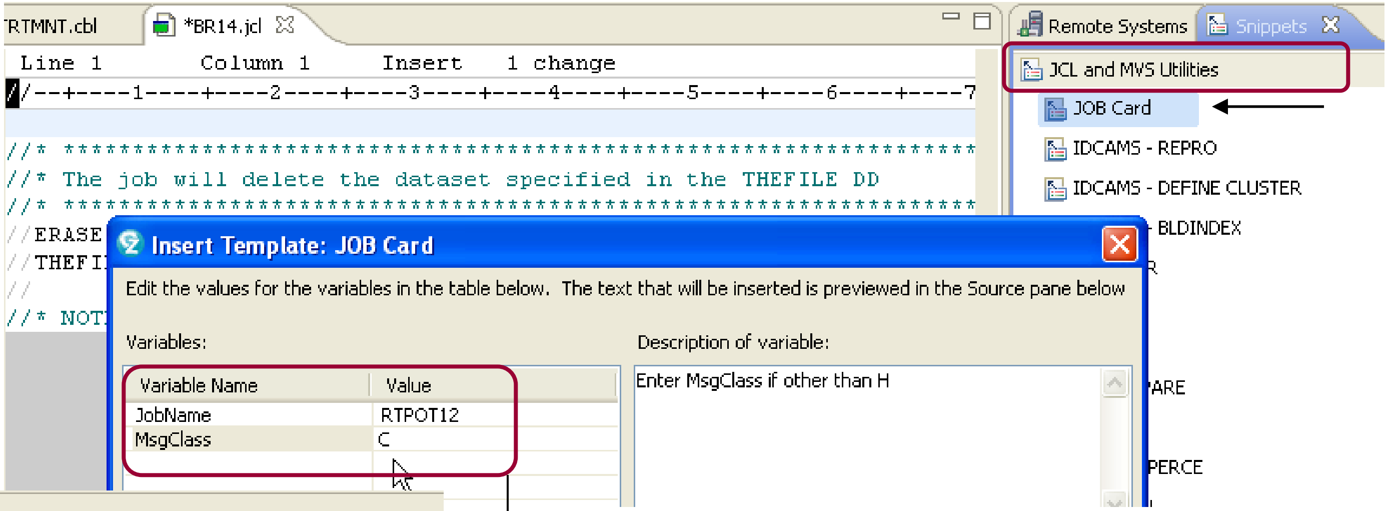


```
Line 1        Column 1       Insert
/--+----1----+----2----+----3----+----4----+----5----+---
//***********************************************************
//*     CUSTOMIZE THE JOB CARD FOR THIS UTILITY PROGRAM
//**********************
//*
//LEEBR14 JOB ,
// MSGCLASS=H,TIME=(,4),
//* *********************
```

- From your PDS open a piece of JCL that contains a valid Job Card
- Select and Cut (**Ctrl+X**) the Job card
- Follow the previous steps to add the Job card snippet to your JCL category
  - During the process of creating the Snippet add JobName and MsgClass as variables – to be filled in by the developer during the reuse of the Snippet

To use the Job Card Snippet

- Open a piece of JCL that does not currently have a Job Card, and set your cursor focus to line 1/byte 1
- From the Snippets view, Expand the JCL category. Find and double-click your **JOB Card** Snippet
- At the prompt, enter a new **JobName** and a new **MsgClass** value and click **OK** and verify your work
- Verify your work
- Submit the job



```
Line 1        Column 1       Insert    1 change
/--+----1----+----2----+----3----+----4----+----5----+----6----+----7

//* ************************************************************
//* The job will delete the dataset specified in the THEFILE DD
//* ************************************************************
//ERASE
//THEFI
//
//* NOT
```

Insert Template: JOB Card

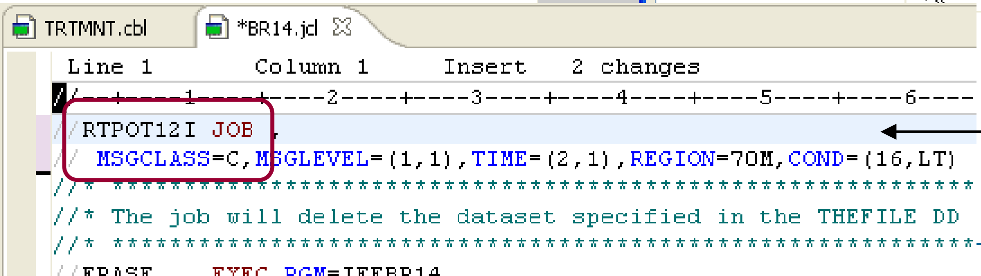
Edit the values for the variables in the table below. The text that will be inserted is previewed in the Source pane below

Variables:

Variable Name	Value
JobName	RTPOT12
MsgClass	C

Description of variable:

Enter MsgClass if other than H



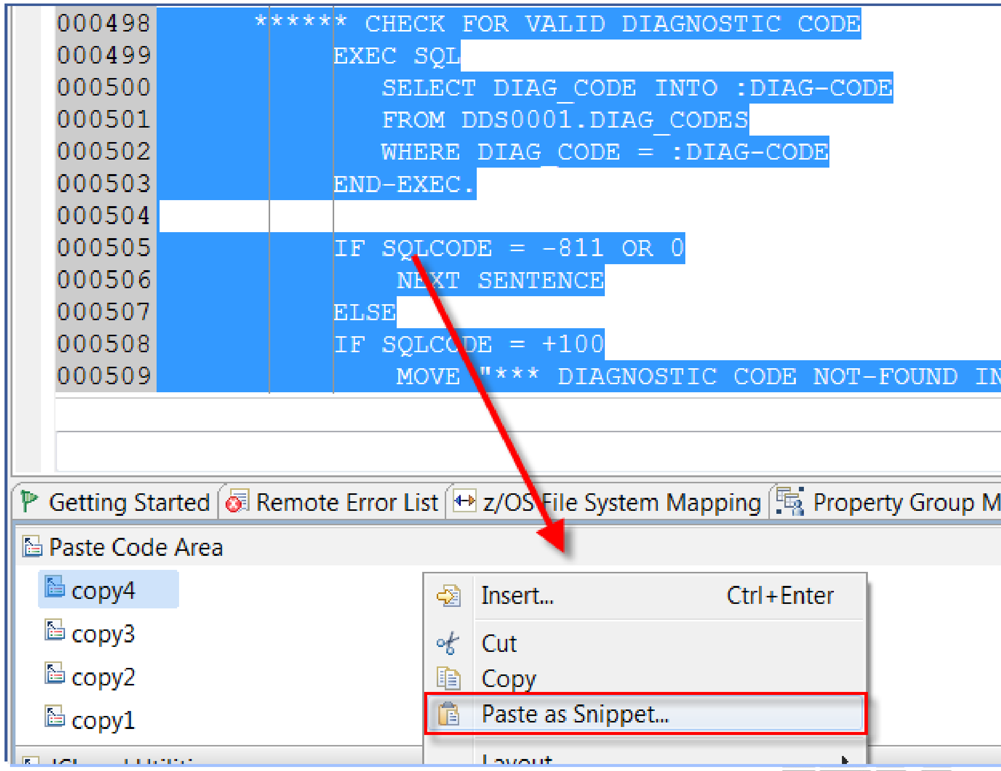
```
Line 1        Column 1       Insert    2 changes
/--+----1----+----2----+----3----+----4----+----5----+----6----
//RTPOT12I JOB ,
// MSGCLASS=C,MSGLEVEL=(1,1),TIME=(2,1),REGION=70M,COND=(16,LT)
//* ************************************************************
//* The job will delete the dataset specified in the THEFILE DD
//* ************************************************************
//ERASE     EXEC PGM=IEFBR14
```

Using Code Snippets as a Scratch Pad Area for Multiple Paste Buffers

Occasionally you may need to create multiple "copy/paste buffers" – if you need to say, replicate a set of changes across multiple programs.

This can be accomplished using **Snippets**:

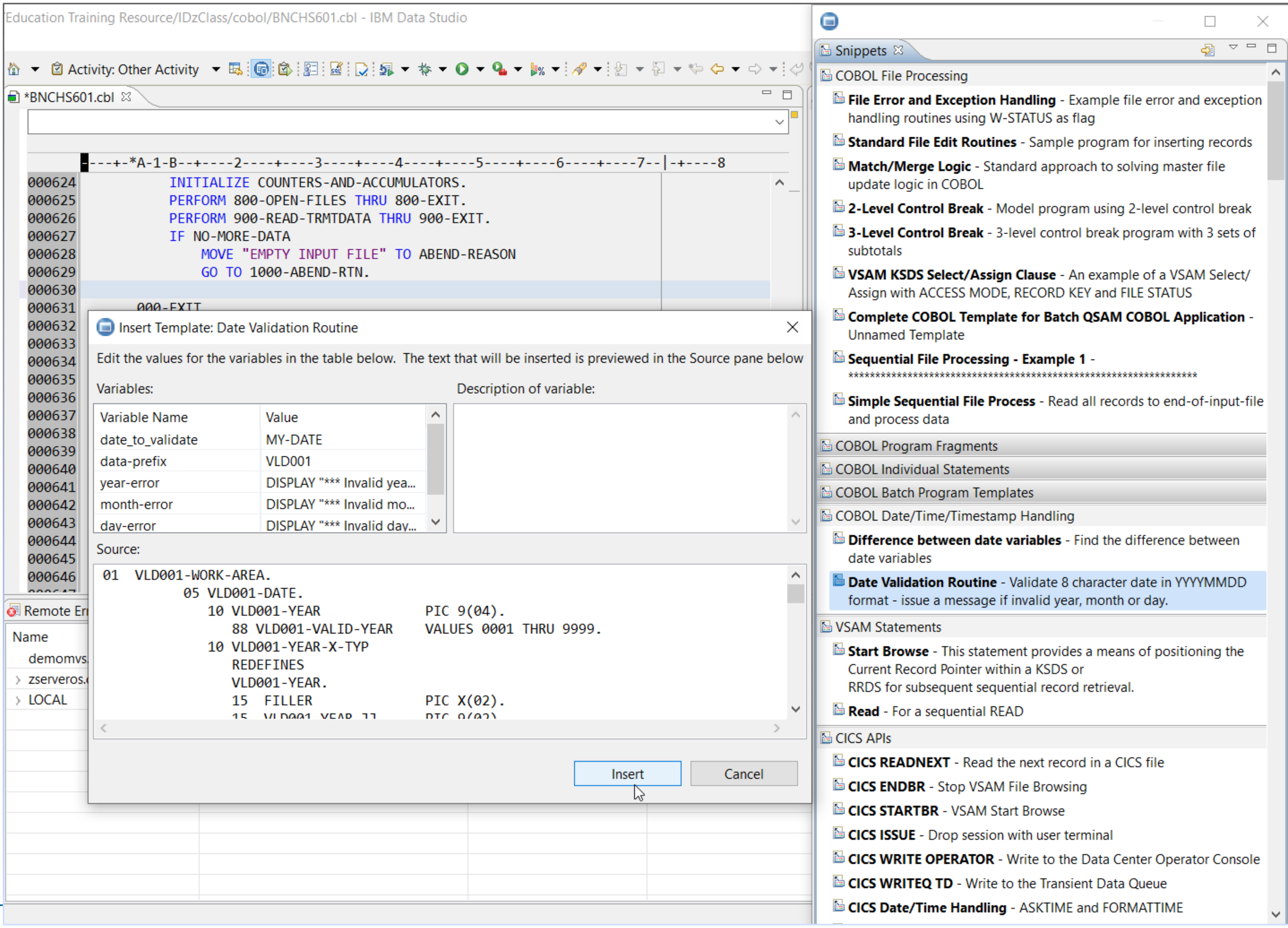
1. Open a program
2. Copy and create a Snippet from a code fragment
   - Optionally customize the Snippet to include Variables – for generalized use
3. Create another Snippet
4. Repeat from step 1 until you've created separate Snippets for each code fragment
5. Apply the Snippets to your program(s)
6. Optionally **Export** the Snippets to other developers on your team



```
000498       ****** CHECK FOR VALID DIAGNOSTIC CODE
000499              EXEC SQL
000500                 SELECT DIAG_CODE INTO :DIAG-CODE
000501                 FROM DDS0001.DIAG_CODES
000502                 WHERE DIAG_CODE = :DIAG-CODE
000503              END-EXEC.
000504
000505              IF SQLCODE = -811 OR 0
000506                  NEXT SENTENCE
000507              ELSE
000508              IF SQLCODE = +100
000509                  MOVE "*** DIAGNOSTIC CODE NOT-FOUND IN
```

Optimizing the use of the Snippets View

If you have built out a decent collection of Snippets consider dragging/dropping the Snippets view outside of the workbench - or over to a dual monitor for optimal use



Education Training Resource/IDzClass/cobol/BNCHS601.cbl - IBM Data Studio

Activity: Other Activity

*BNCHS601.cbl

```
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-+----8
           INITIALIZE COUNTERS-AND-ACCUMULATORS.
           PERFORM 800-OPEN-FILES THRU 800-EXIT.
           PERFORM 900-READ-TRMTDATA THRU 900-EXIT.
           IF NO-MORE-DATA
               MOVE "EMPTY INPUT FILE" TO ABEND-REASON
               GO TO 1000-ABEND-RTN.

       000-EXIT
```

Remote Er

Name
- demomvs
- zserveros.
- LOCAL

Insert Template: Date Validation Routine

Edit the values for the variables in the table below. The text that will be inserted is previewed in the Source pane below

Variables:

Variable Name	Value
date_to_validate	MY-DATE
data-prefix	VLD001
year-error	DISPLAY "*** Invalid yea...
month-error	DISPLAY "*** Invalid mo...
day-error	DISPLAY "*** Invalid day...

Description of variable:

Source:

```
01  VLD001-WORK-AREA.
          05 VLD001-DATE.
             10 VLD001-YEAR             PIC 9(04).
                88 VLD001-VALID-YEAR    VALUES 0001 THRU 9999.
             10 VLD001-YEAR-X-TYP
                REDEFINES
                VLD001-YEAR.
                15  FILLER              PIC X(02).
```

Insert | Cancel

Snippets

COBOL File Processing
- **File Error and Exception Handling** - Example file error and exception handling routines using W-STATUS as flag
- **Standard File Edit Routines** - Sample program for inserting records
- **Match/Merge Logic** - Standard approach to solving master file update logic in COBOL
- **2-Level Control Break** - Model program using 2-level control break
- **3-Level Control Break** - 3-level control break program with 3 sets of subtotals
- **VSAM KSDS Select/Assign Clause** - An example of a VSAM Select/ Assign with ACCESS MODE, RECORD KEY and FILE STATUS
- **Complete COBOL Template for Batch QSAM COBOL Application** - Unnamed Template
- **Sequential File Processing - Example 1** - **
- **Simple Sequential File Process** - Read all records to end-of-input-file and process data

COBOL Program Fragments

COBOL Individual Statements

COBOL Batch Program Templates

COBOL Date/Time/Timestamp Handling
- **Difference between date variables** - Find the difference between date variables
- **Date Validation Routine** - Validate 8 character date in YYYYMMDD format - issue a message if invalid year, month or day.

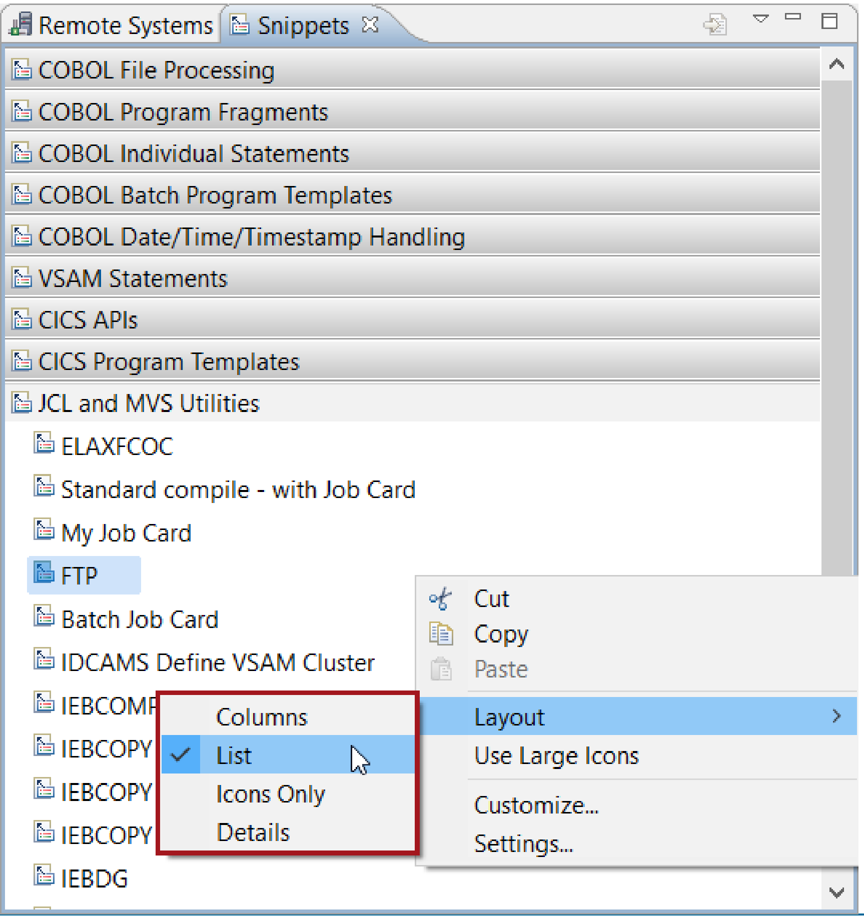
VSAM Statements
- **Start Browse** - This statement provides a means of positioning the Current Record Pointer within a KSDS or RRDS for subsequent sequential record retrieval.
- **Read** - For a sequential READ

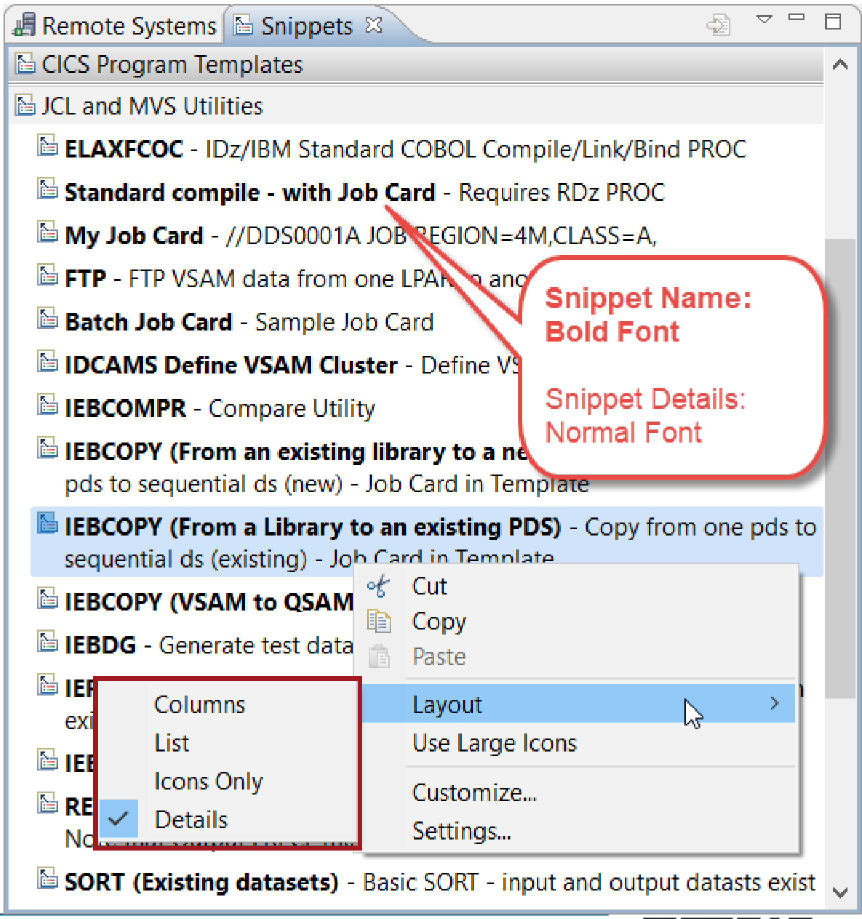
CICS APIs
- **CICS READNEXT** - Read the next record in a CICS file
- **CICS ENDBR** - Stop VSAM File Browsing
- **CICS STARTBR** - VSAM Start Browse
- **CICS ISSUE** - Drop session with user terminal
- **CICS WRITE OPERATOR** - Write to the Data Center Operator Console
- **CICS WRITEQ TD** - Write to the Transient Data Queue
- **CICS Date/Time Handling** - ASKTIME and FORMATTIME

Snippets View - Layout

There are several options for the Snippets View U.I.

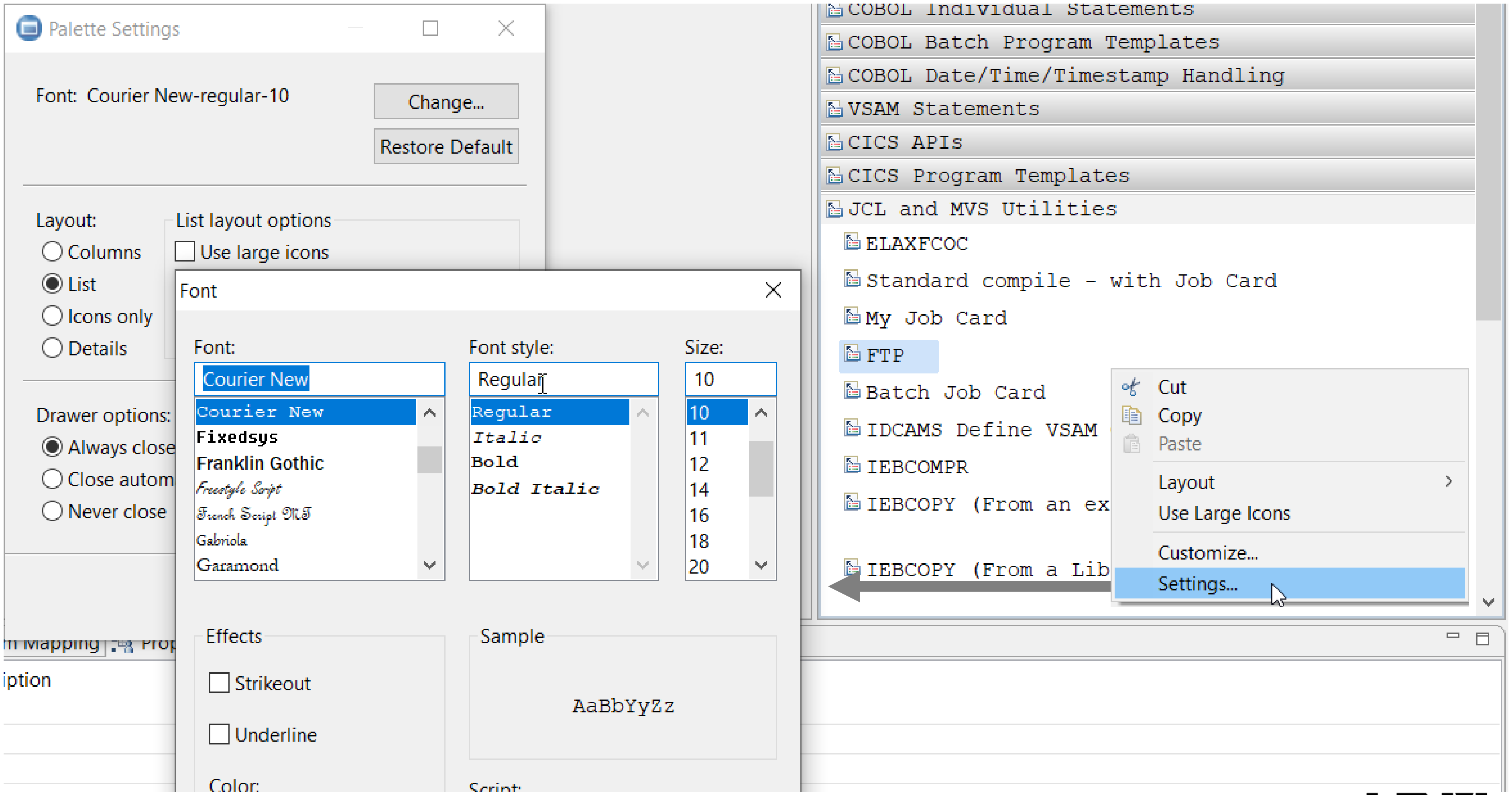
Typically you choose either List or Details




Remote Systems
Snippets
CICS Program Templates
JCL and MVS Utilities
ELAXFCOC - IDz/IBM Standard COBOL Compile/Link/Bind PROC
Standard compile - with Job Card - Requires RDz PROC
My Job Card - //DDS0001A JOB REGION=4M,CLASS=A,
FTP - FTP VSAM data from one LPAR to ano
Batch Job Card - Sample Job Card
IDCAMS Define VSAM Cluster - Define VS
IEBCOMPR - Compare Utility
IEBCOPY (From an existing library to a ne pds to sequential ds (new) - Job Card in Template
IEBCOPY (From a Library to an existing PDS) - Copy from one pds to sequential ds (existing) - Job Card in Template
IEBCOPY (VSAM to QSAM
IEBDG - Generate test data
SORT (Existing datasets) - Basic SORT - input and output datasts exist
Snippet Name: Bold Font
Snippet Details: Normal Font
Columns
List
Icons Only
✓ Details
Cut
Copy
Paste
Layout
Use Large Icons
Customize...
Settings...

Snippets View - Customize Font from Settings…

You can customize the Snippet View's text font type and font size



Palette Settings

Font: Courier New-regular-10

Change...

Restore Default

Layout: ◯ Columns [✓ List] ◯ Icons only ◯ Details

List layout options: ☐ Use large icons

Drawer options: [✓ Always close] ◯ Close autom ◯ Never close

Font

Font:	Font style:	Size:
Courier New	Regular	10
Courier New	Regular	10
Fixedsys	Italic	11
Franklin Gothic	Bold	12
Freestyle Script	Bold Italic	14
French Script MT		16
Gabriola		18
Garamond		20

Effects: ☐ Strikeout ☐ Underline

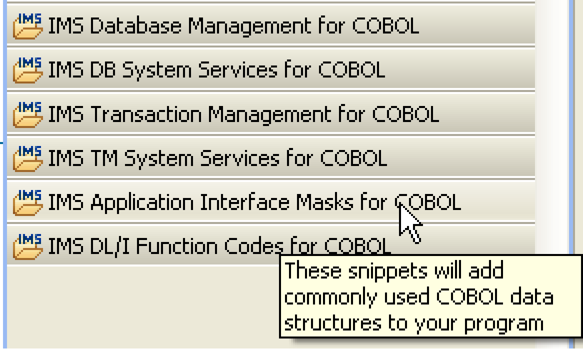
Color:

Sample: AaBbYyZz

Script:

```
COBOL Individual Statements
COBOL Batch Program Templates
COBOL Date/Time/Timestamp Handling
VSAM Statements
CICS APIs
CICS Program Templates
JCL and MVS Utilities
  ELAXFCOC
  Standard compile - with Job Card
  My Job Card
  FTP
  Batch Job Card
  IDCAMS Define VSAM
  IEBCOMPR
  IEBCOPY (From an ex
  IEBCOPY (From a Lib
```

The IMS Code Snippets

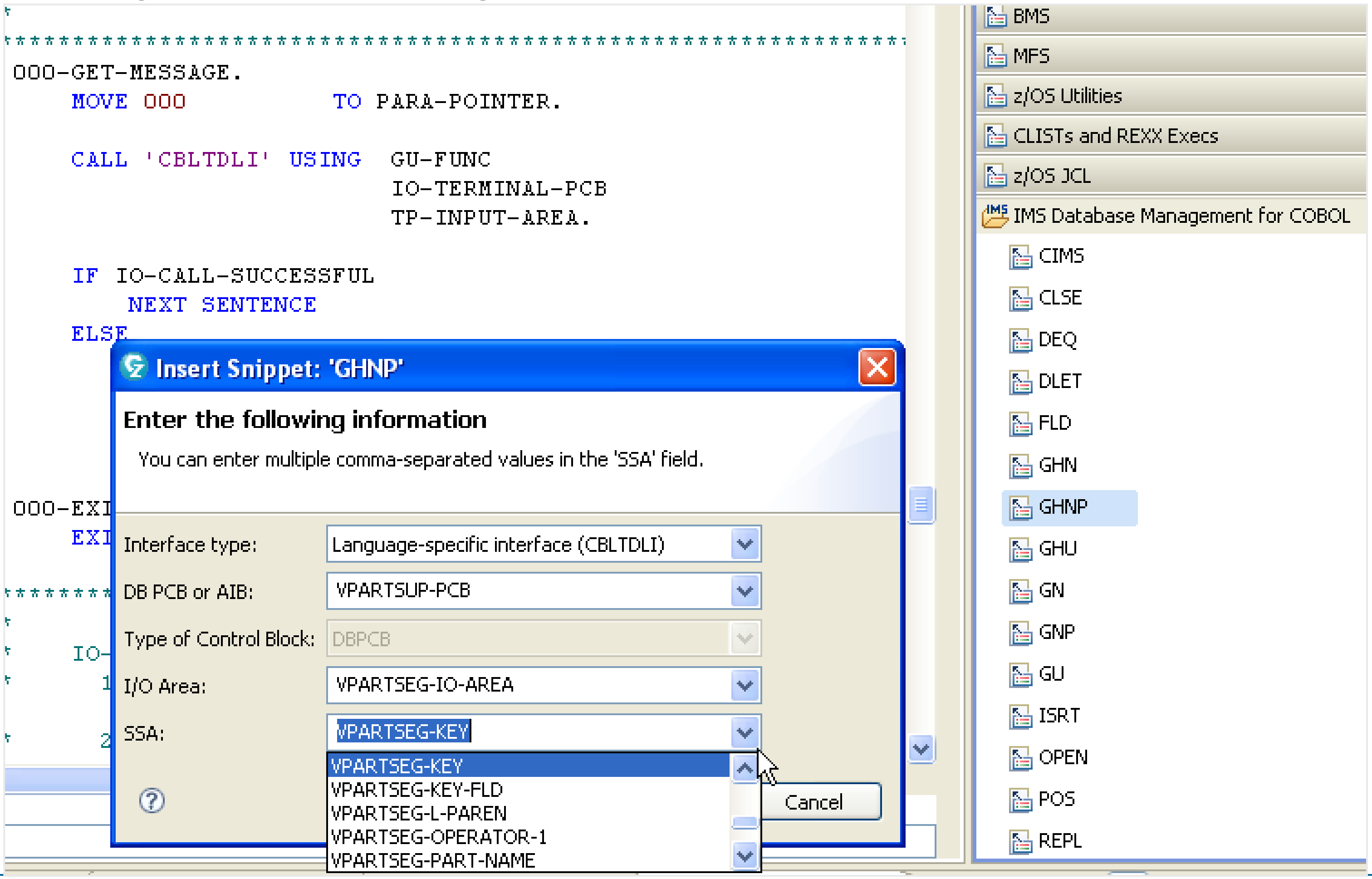


IMS Database Management for COBOL
IMS DB System Services for COBOL
IMS Transaction Management for COBOL
IMS TM System Services for COBOL
IMS Application Interface Masks for COBOL
IMS DL/I Function Codes for COBOL

These snippets will add commonly used COBOL data structures to your program

Starting with RDz v7.6, IBM shipped a number of useful IMS Code Snippets with the product ➔

These snippets go beyond simple text-based insertion to read your Data Division entries, and offer options for building statements using combo-boxes



```
      *
      *****************************************************************
       000-GET-MESSAGE.
           MOVE 000            TO PARA-POINTER.

           CALL 'CBLTDLI' USING  GU-FUNC
                                 IO-TERMINAL-PCB
                                 TP-INPUT-AREA.


           IF IO-CALL-SUCCESSFUL
              NEXT SENTENCE
           ELSE
```

Insert Snippet: 'GHNP'

Enter the following information

You can enter multiple comma-separated values in the 'SSA' field.

Interface type: Language-specific interface (CBLTDLI)
DB PCB or AIB: VPARTSUP-PCB
Type of Control Block: DBPCB
I/O Area: VPARTSEG-IO-AREA
SSA: VPARTSEG-KEY

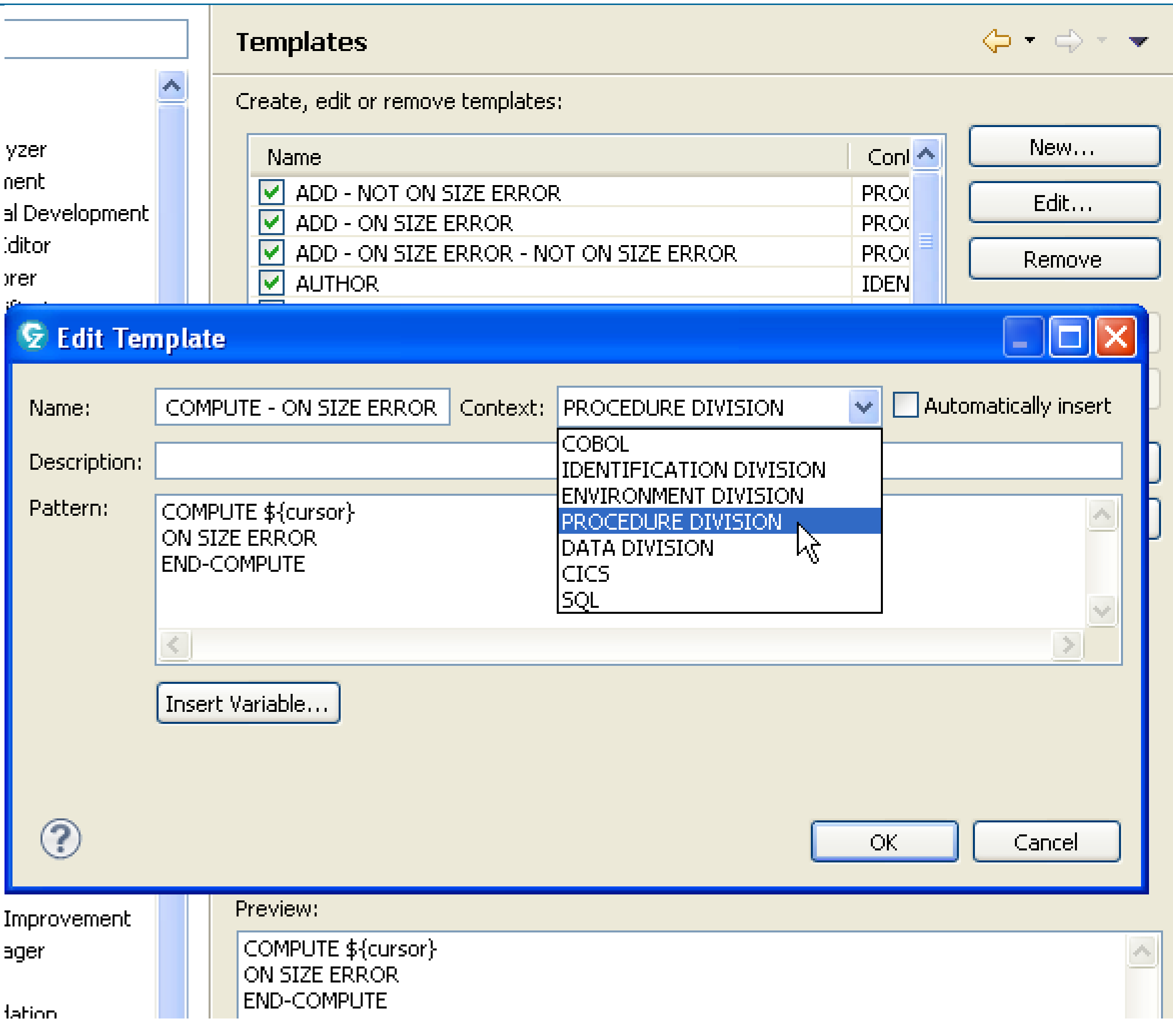
VPARTSEG-KEY
VPARTSEG-KEY-FLD
VPARTSEG-L-PAREN
VPARTSEG-OPERATOR-1
VPARTSEG-PART-NAME

Cancel

BMS
MFS
z/OS Utilities
CLISTs and REXX Execs
z/OS JCL
IMS Database Management for COBOL
CIMS
CLSE
DEQ
DLET
FLD
GHN
GHNP
GHU
GN
GNP
GU
ISRT
OPEN
POS
REPL

IDz's Customize-able Content Assist Templates

- Finally - you can customize IDz's template "proposals" offered in the Content Assist
- You access this from:
  - **Window**
    - **Preferences**
      - **COBOL**
        - **Templates**
- Customization options include:
  - Modify (Edit…) an existing template
  - Add a (New…) template
  - Remove a template
  - Export all templates – so that other team members can share
  - Import…
  - Restore Removed (un-delete)
  - Revert to Default (un-modify)



Name	Cont
✓ ADD - NOT ON SIZE ERROR	PRO
✓ ADD - ON SIZE ERROR	PRO
✓ ADD - ON SIZE ERROR - NOT ON SIZE ERROR	PRO
✓ AUTHOR	IDEN

Name: COMPUTE - ON SIZE ERROR Context: PROCEDURE DIVISION

COBOL
IDENTIFICATION DIVISION
ENVIRONMENT DIVISION
PROCEDURE DIVISION
DATA DIVISION
CICS
SQL

Description:

Pattern:

```
COMPUTE ${cursor}
ON SIZE ERROR
END-COMPUTE
```

Preview:

```
COMPUTE ${cursor}
ON SIZE ERROR
END-COMPUTE
```

You can customize a template's:
- Content - Pattern - Context - where it's applicable - Description – hover help

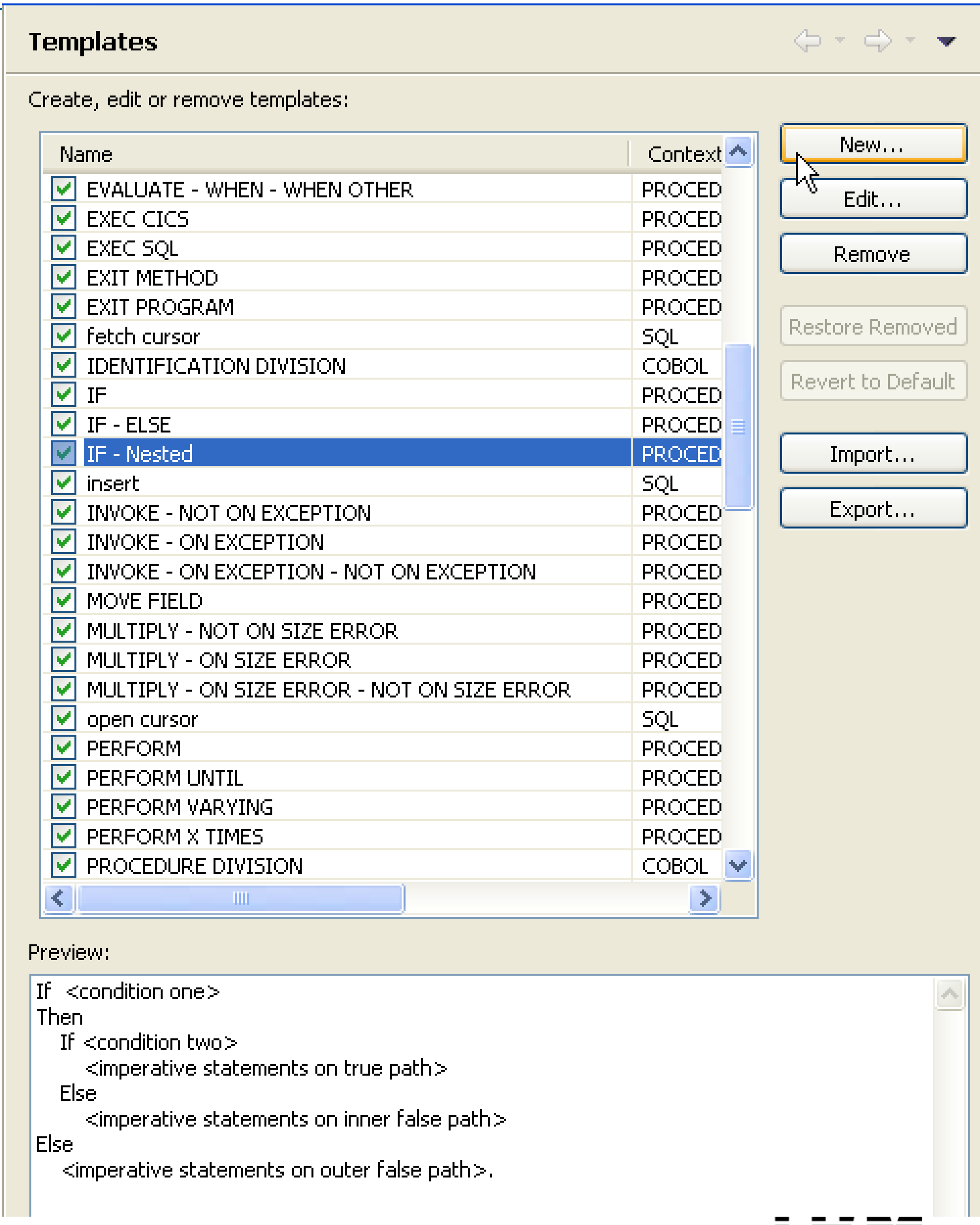
Steps – Customizing Template Proposals

- From **Window, Preferences, COBOL, Templates:**
  - Select one of the Template proposals and delete (**Remove**) it
  - Select a Template proposal and **Edit…** (change it) – something simple like changing the case to mixed-case, instead of all **UPPER** case
  - Add a **New…** proposal, as shown ➔

    You can copy and paste the this text.

```
If  <condition one>
Then
   If <condition two>
      <imperative statements on true path>
   Else
      <imperative statements on inner false path>
Else
   <imperative statements on outer false path>.
```

- Test your work out in one of your programs



Name	Context
✓ EVALUATE - WHEN - WHEN OTHER	PROCED
✓ EXEC CICS	PROCED
✓ EXEC SQL	PROCED
✓ EXIT METHOD	PROCED
✓ EXIT PROGRAM	PROCED
✓ fetch cursor	SQL
✓ IDENTIFICATION DIVISION	COBOL
✓ IF	PROCED
✓ IF - ELSE	PROCED
✓ IF - Nested	PROCED
✓ insert	SQL
✓ INVOKE - NOT ON EXCEPTION	PROCED
✓ INVOKE - ON EXCEPTION	PROCED
✓ INVOKE - ON EXCEPTION - NOT ON EXCEPTION	PROCED
✓ MOVE FIELD	PROCED
✓ MULTIPLY - NOT ON SIZE ERROR	PROCED
✓ MULTIPLY - ON SIZE ERROR	PROCED
✓ MULTIPLY - ON SIZE ERROR - NOT ON SIZE ERROR	PROCED
✓ open cursor	SQL
✓ PERFORM	PROCED
✓ PERFORM UNTIL	PROCED
✓ PERFORM VARYING	PROCED
✓ PERFORM X TIMES	PROCED
✓ PROCEDURE DIVISION	COBOL

```
If  <condition one>
Then
   If <condition two>
      <imperative statements on true path>
   Else
      <imperative statements on inner false path>
Else
   <imperative statements on outer false path>.
```

Creating New Programs Using Wizard

- There are several ways to create new programs from scratch
- The "Best Practice" method is to use IDz's COBOL Program Wizard

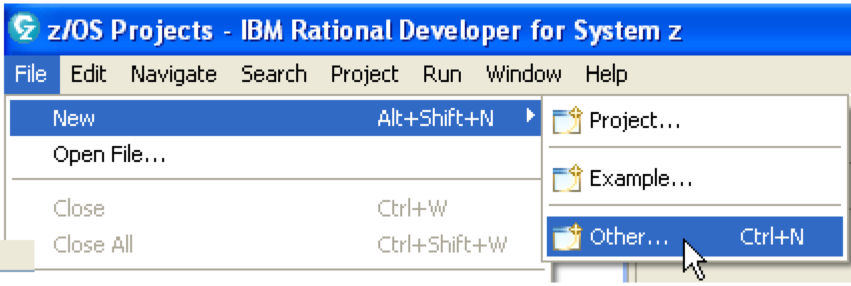
⏭ From **File, New > Other…**

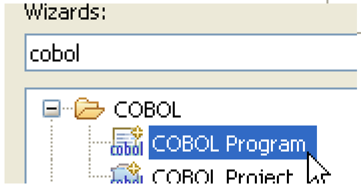
…in the Wizards panel,

- Type: **cobol**
- Select COBOL Program
- Click **Next >**

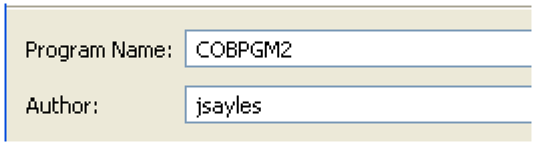
…in the COBOL Program panel,

- Name the Program
- Click **Next >**

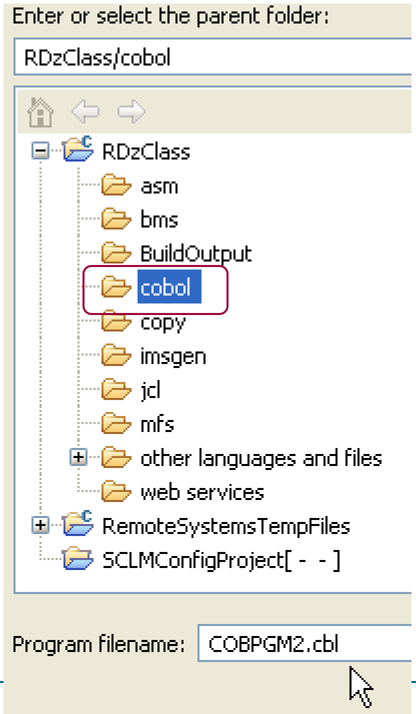



Wizards:

cobol


Program Name: COBPGM2

Author: jsayles


Enter or select the parent folder:

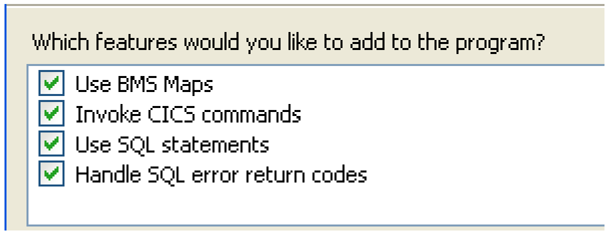
RDzClass/cobol

Program filename: COBPGM2.cbl

- Finally you specify which folder to create the program:
  - Select the **cobol** folder
  - Click **Next >**

Creating New Programs From Templates – continued

You can add CICS or DB2 template sample code to your new program:



Which features would you like to add to the program?

- [✓] Use BMS Maps
- [✓] Invoke CICS commands
- [✓] Use SQL statements
- [✓] Handle SQL error return codes

- Check the features you'd like
- Click **Finish**

- A few things happen:
  - Your new program is created ➔
  - The **Snippets** view is opened
    - Snippets information can be found in Appendix B of these slides

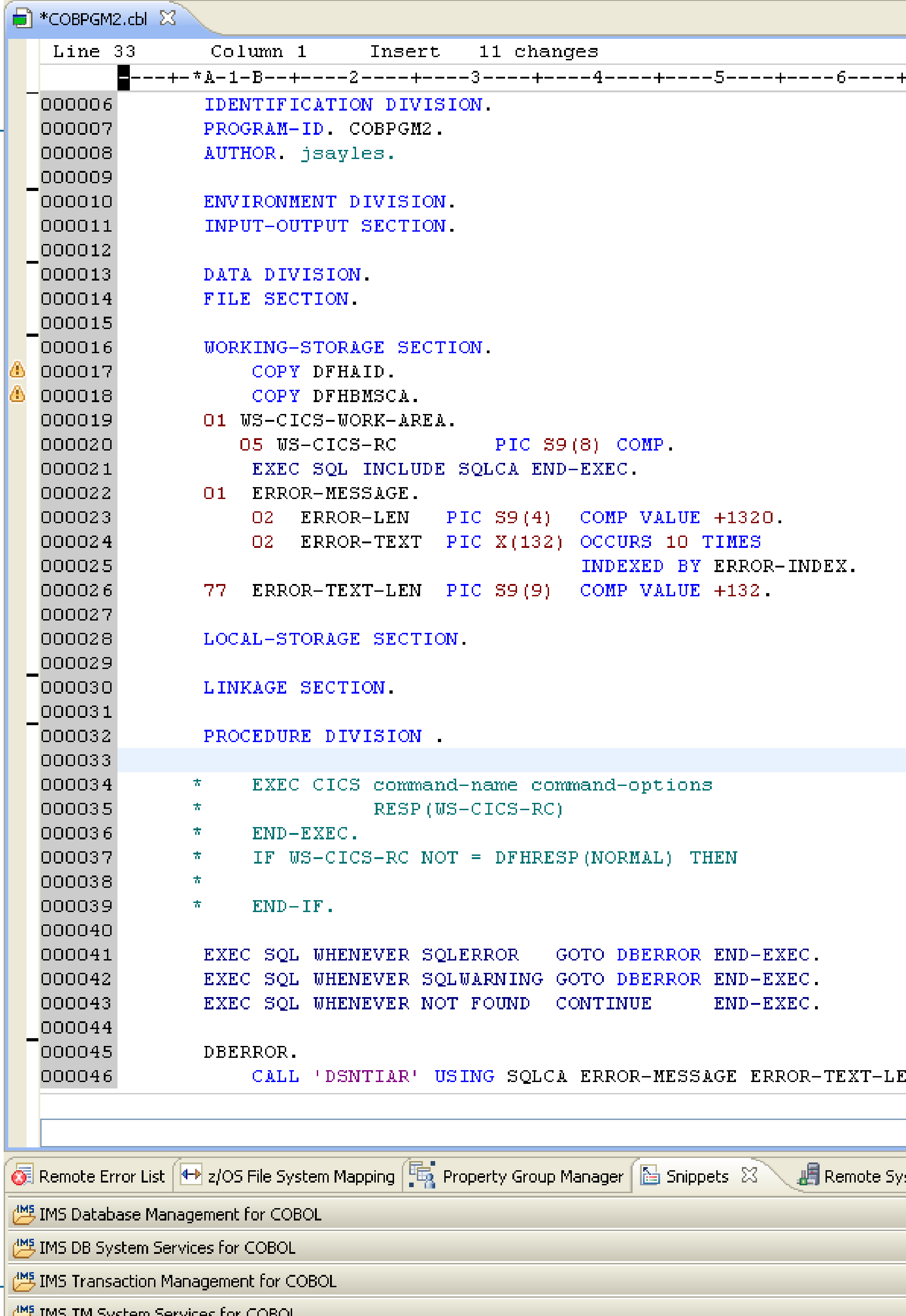
Note that you can customize the templates used to create new programs

From **Window, Preferences,** select:

COBOL

– **Code Templates**

– **Features**

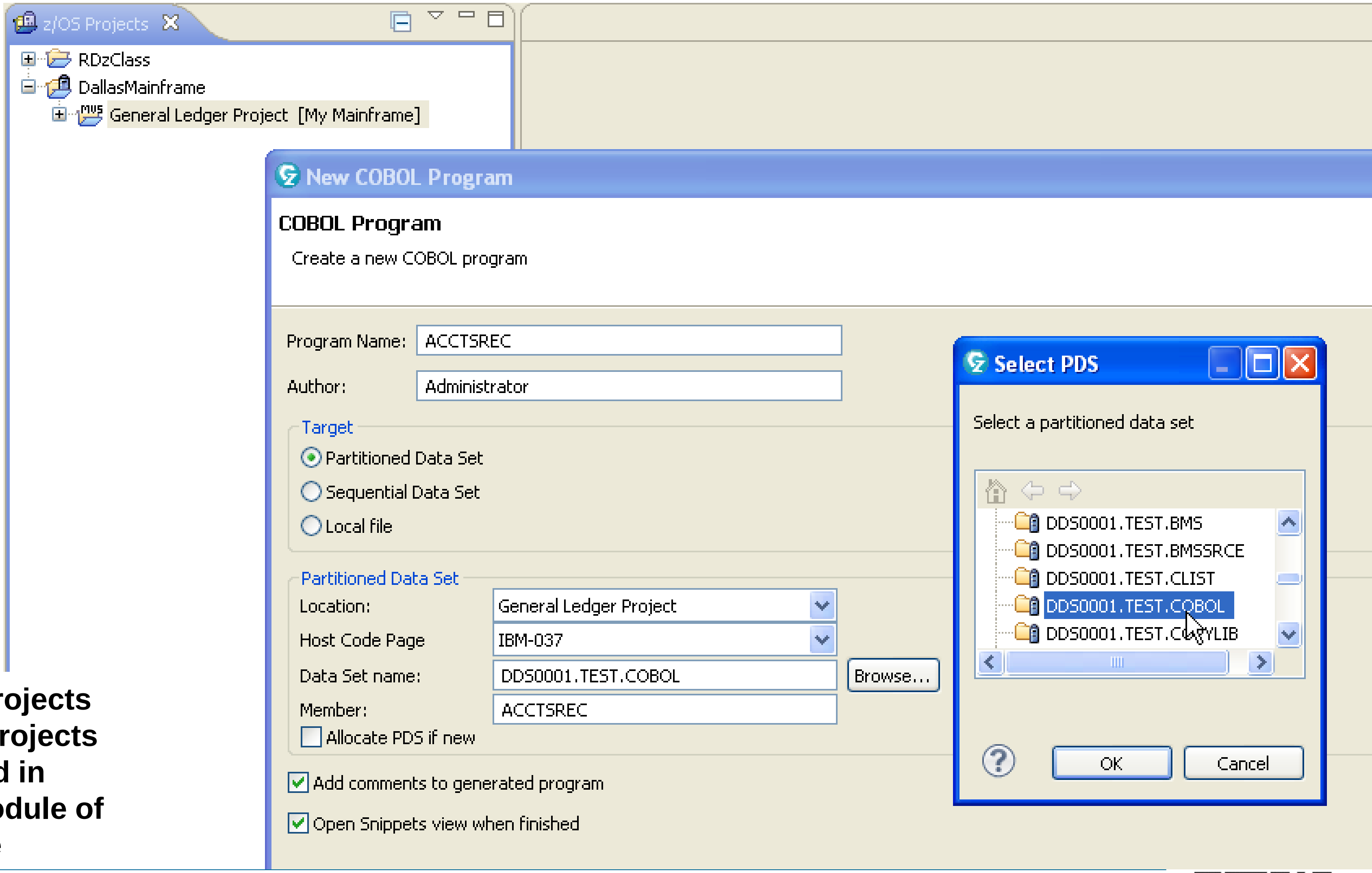


```
 Line 33        Column 1       Insert    11 changes
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+
000006        IDENTIFICATION DIVISION.
000007        PROGRAM-ID. COBPGM2.
000008        AUTHOR. jsayles.
000009
000010        ENVIRONMENT DIVISION.
000011        INPUT-OUTPUT SECTION.
000012
000013        DATA DIVISION.
000014        FILE SECTION.
000015
000016        WORKING-STORAGE SECTION.
000017            COPY DFHAID.
000018            COPY DFHBMSCA.
000019        01 WS-CICS-WORK-AREA.
000020           05 WS-CICS-RC          PIC S9(8) COMP.
000021            EXEC SQL INCLUDE SQLCA END-EXEC.
000022        01  ERROR-MESSAGE.
000023            02  ERROR-LEN    PIC S9(4)  COMP VALUE +1320.
000024            02  ERROR-TEXT   PIC X(132) OCCURS 10 TIMES
000025                                        INDEXED BY ERROR-INDEX.
000026        77  ERROR-TEXT-LEN   PIC S9(9)  COMP VALUE +132.
000027
000028        LOCAL-STORAGE SECTION.
000029
000030        LINKAGE SECTION.
000031
000032        PROCEDURE DIVISION .
000033
000034       *    EXEC CICS command-name command-options
000035       *              RESP(WS-CICS-RC)
000036       *    END-EXEC.
000037       *    IF WS-CICS-RC NOT = DFHRESP(NORMAL) THEN
000038       *
000039       *    END-IF.
000040
000041        EXEC SQL WHENEVER SQLERROR   GOTO DBERROR END-EXEC.
000042        EXEC SQL WHENEVER SQLWARNING GOTO DBERROR END-EXEC.
000043        EXEC SQL WHENEVER NOT FOUND  CONTINUE     END-EXEC.
000044
000045        DBERROR.
000046            CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LE
```

IMS Database Management for COBOL
IMS DB System Services for COBOL
IMS Transaction Management for COBOL
IMS TM System Services for COBOL

Create New Program in a z/OS LPAR

- You can create new programs using the New COBOL Program wizard, provided you are connected to a z/OS LPAR, and that you have created a z/OS Project/MVS Subproject (see Location: in the screen capture below).



z/OS Projects

- RDzClass
- DallasMainframe
  - General Ledger Project [My Mainframe]

New COBOL Program

COBOL Program

Create a new COBOL program

Program Name: ACCTSREC

Author: Administrator

Target

- [✓ Partitioned Data Set]
- Sequential Data Set
- Local file

Partitioned Data Set

Location: General Ledger Project

Host Code Page: IBM-037

Data Set name: DDS0001.TEST.COBOL

Member: ACCTSREC

- [ ] Allocate PDS if new
- [✓] Add comments to generated program
- [✓] Open Snippets view when finished

Select PDS

Select a partitioned data set

- DDS0001.TEST.BMS
- DDS0001.TEST.BMSSRCE
- DDS0001.TEST.CLIST
- DDS0001.TEST.COBOL
- DDS0001.TEST.COPYLIB

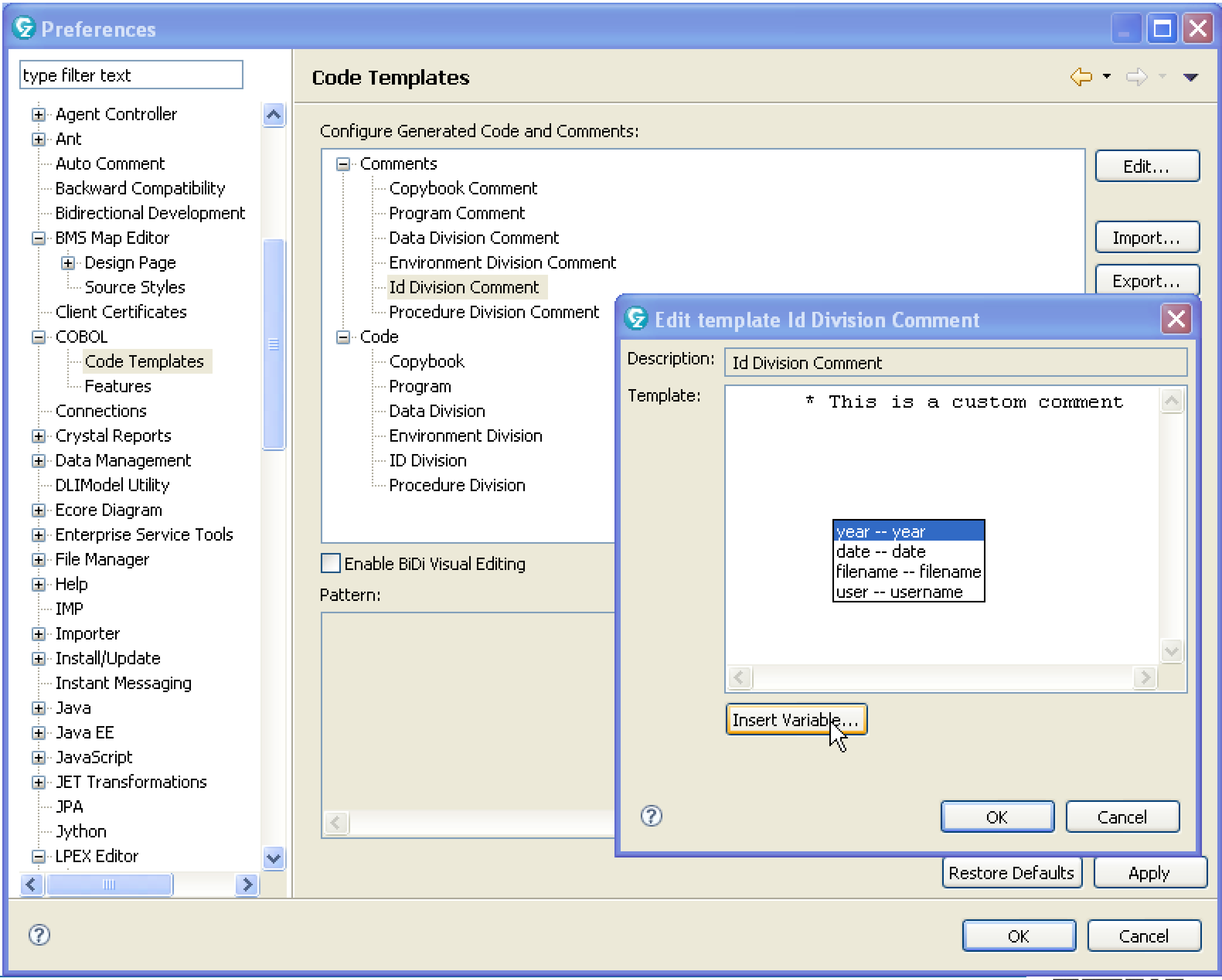
MVS SubProjects and z/OS Projects are covered in another module of this course

Customize the New Program Templates – Comments

You can create a custom Code Template for COBOL comments or the base program code itself.

To add or customize comments:

- Click the comment option you wish to modify
- Code an asterisk in position 7 (you'll have to space over 1-6)
- You can insert Variables that are filled in when new "templatized" programs are created



Description: Id Division Comment

Template:

```
      * This is a custom comment
```

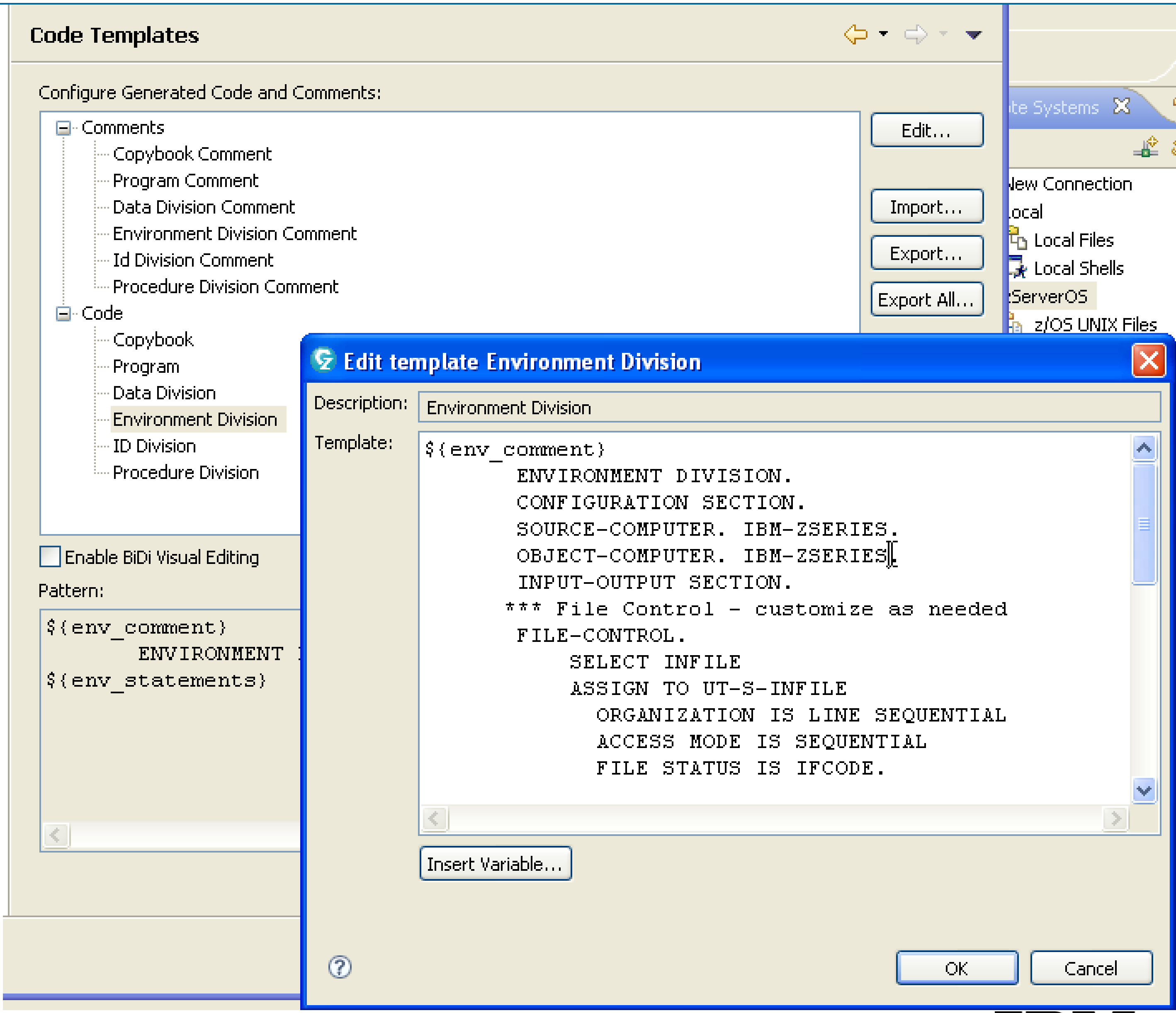
year -- year
date -- date
filename -- filename
user -- username

Customize the New Program Templates – Program Code

And you can add your own entries, common files, databases, variables, routines etc. to either:

- An entire program
- Separate program divisions

When a new program is created using the templates all of the custom comments and code are inserted.



Description: Environment Division

Template:

```
${env_comment}
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-ZSERIES.
       OBJECT-COMPUTER. IBM-ZSERIES.
       INPUT-OUTPUT SECTION.
      *** File Control - customize as needed
       FILE-CONTROL.
           SELECT INFILE
           ASSIGN TO UT-S-INFILE
             ORGANIZATION IS LINE SEQUENTIAL
             ACCESS MODE IS SEQUENTIAL
             FILE STATUS IS IFCODE.
```

Pattern:

```
${env_comment}
       ENVIRONMENT
${env_statements}
```