A guide to use E-resource

1. Introduction

This has been mentioned in the book's introduction, and we repeat it here. You should perform the hands-on exercises as you read a chapter and encounter a hands-on exercise. You should not defer the performance of hands-on exercises to the time after complete reading of the chapter or the book. The book has been written assuming the reader will follow this procedure: perform the hands-on exercises as they are encountered. The objective of the hands-on exercises is to convey more than what the textual explanations and descriptions can convey.

To aid you in the performance of the ABAP programming hands-on exercises as they are encountered in the chapters, we have provided you with the ABAP program source lines in the E-Resource.

2. E-Resource - structure and contents

In the E-Resource, the source programs (*executable programs*, *include programs*, *function modules* and *methods* of classes) are available as text files. The E-Resource contains all hands-on exercise source programs demonstrated in the book and a few more not demonstrated but mentioned in the book. Most programs in the E-Resource are either fully or partially listed in the book as well. The E-Resource though contains only the source lines of programs, other components of program like text elements etc. have to be created manually or copied from previous program if available. Other workbench objects used by programs like DDIC objects, messages, interfaces, screens etc. have to be created manually. The directions to create the workbench objects used by a hands-on exercise program are contained in the book with the appropriate screen shots. The source program text files are organized chapter wise, with a separate folder for every chapter like SOURCE_CHAPTER04, SOURCE_CHAPTER05 and so on.

Each *executable* or *include program* or *function module* or *method* of class in the folders is a separate text file which can be opened with any text editor – in Microsoft windows note pad if you are operating on Microsoft windows. The text file names are same as the program names, for instance the program YCL_CH06_ITAB05 in chapter 6 has program source lines in a text file with the same name.

Following is a summary of chapter wise number of source programs:

Chapter No.	**Number of Programs**
04	**17**
05	**11**
06	**16**
07	**15**
08	**19**
09	**16**
10	**6**
11	**15**
12	**3**
13	**4**
14	**6**
Total	**128**

Following is a detailed list of chapter wise source programs:

Serial No.	Chapter No.	Program Name	Comments
01	04	YCL_CH04_01_HELLO_WORLD	
02		YCL_CH04_02_LIST_SYS_FIELDS	
03		YCL_CH04_03_ELEM_DATA_OBJECTS	
04		YCL_CH04_04_CONSTANTS	
05		YCL_CH04_05_TRANSF_DATE_MOVE	
06		YCL_CH04_06_STRU_DATA_OBJ	
07		YCL_CH04_07_STRU_DATA_TYP_OBJ1	
08		YCL_CH04_08_STRU_DATA_TYP_OBJ2	
09		YCL_CH04_09_STRU_MOVE_CORR	
10		YCL_CH04_10_DATE_ARITHM	
11		YCL_CH04_11_TRANSLATE	
12		YCL_CH04_12_CONCATENATE	
13		YCL_CH04_13_SPLIT	
14		YCL_CH04_14_STRLEN	
15		YCL_CH04_15_CONDENSE	
16		YCL_CH04_16_SY_INDEX	
17		YCL_CH04_17_DESCRIBE_FIELD	
18	05	YCL_CH05_01_LIST_SYNTAX	
19		YCL_CH05_02_CUST_LIST01	
20		YCL_CH05_03_CUST_LIST02	
21		YCL_CH05_04_CUST_LIST03	
22		YCL_CH05_05_CUST_LIST04	
23		YCL_CH05_06_CUST_LIST05	
24		YCL_CH05_07_CUST_LIST06	
25		YCL_CH05_08_BILL_DOCS_LIST	
26		YCL_CH05_09_RESERVE_LINES	
27		YCL_CH05_10_LEFT_SCROLL_BNDRY	
28		YCL_CH05_11_MISC	
29	06	YCL_CH06_DATA_DEC_REFS	
30		YCL_CH06_ITAB01	
31		YCL_CH06_ITAB02	
32		YCL_CH06_ITAB03	
33		YCL_CH06_ITAB04	

Serial No.	Chapter No.	Program Name	Comments
34	06	YCL_CH06_ITAB05	
35		YCL_CH06_ITAB06	
36		YCL_CH06_ITAB07	
37		YCL_CH06_ITAB08	
38		YCL_CH06_ITAB09	
39		YCL_CH06_ITAB10	
40		YCL_CH06_ITAB11	
41		YCL_CH06_ITAB12	
42		YCL_CH06_ITAB13	
43		YCL_CH06_ITAB14	
44		YCL_CH06_ITAB15	
45	07	CONVERSION_EXIT_ICOMA_INPUT	Conversion Routine
46		CONVERSION_EXIT_ICOMA_OUTPUT	Conversion Routine
47		LYCL_CH07_GROUP02F01	Include Program (for Subroutines) in Function Group
48		YCL_CH07_01_DESCRIBE_FLD_MODUL	
49		YCL_CH07_02_DESCRIBE_FLD_SROUT	
50		YCL_CH07_03_ITAB_AS_PARM	
51		YCL_CH07_04_STRU_AS_PARM	
52		YCL_CH07_05_OUTPUT_INBOX_MACRO	
53		YCL_CH07_06_CALL_SPELL_AMOUNT	
54		YCL_CH07_07_CALL_SPLIT_STRING	
55		YCL_CH07_08_TEST_CONV_ROUT01	
56		YCL_CH07_09_TEST_CONV_ROUT02	
57		YCL_CH07_10_TEST_CONV_ROUT03	
58		YCL_CH07_11_ISSUE_MESSAGE	
59		YCL_CH07_SPLIT_STRING	Function Module
60	08	YCL_CH08_01_INCLUDE_HEADING	Include Program
61		YCL_CH08_02_PSIZE_INTO_TAB	
62		YCL_CH08_03_PSIZE_APPEND_TAB	
63		YCL_CH08_04_SELECT_DISTINCT	
64		YCL_CH08_05_AGGREGATE_FUNC	
65		YCL_CH08_06_AGGREGATE_FUNC_VR1	
66		YCL_CH08_07_AGGREGATE_FUNC_VR2	
67		YCL_CH08_08_HEADER_JOINS	

Serial No.	Chapter No.	Program Name	Comments
68	08	YCL_CH08_09_INNER_JOIN	
69		YCL_CH08_10_OUTER_JOIN	
70		YCL_CH08_11_SUBQUERY	
71		YCL_CH08_12_TABULAR_CONDITION	
72		YCL_CH08_13_CUST_LIST_WITHSTRU	
73		YCL_CH08_14_CUST_LIST_WOUTSTRU	
74		YCL_CH08_15_READ_KEY_ASSIGN	
75		YCL_CH08_16_READ_INDEX_ASSIGN	
76		YCL_CH08_17_CHANGE_WOUT_MODIFY	
77		YCL_CH08_18_SRCE_AT_RUN_TIME	
78		YCL_CH08_19_RSWHOR_AT_RUN_TIME	
79	09	YCL_CH09_SELECTION_SCREEN01	
80		YCL_CH09_SELECTION_SCREEN01VR	
81		YCL_CH09_SELECTION_SCREEN02	
82		YCL_CH09_SELECTION_SCREEN02VR	
83		YCL_CH09_SELECTION_SCREEN03	
84		YCL_CH09_SELECTION_SCREEN04	
85		YCL_CH09_SELECTION_SCREEN05	
86		YCL_CH09_SELECTION_SCREEN05VR	
87		YCL_CH09_SELECTION_SCREEN06	
88		YCL_CH09_SELECTION_SCREEN07	
89		YCL_CH09_SELECTION_SCREEN08	
90		YCL_CH09_SELECTION_SCREEN09	
91		YCL_CH09_SELECTION_SCREEN10	
92		YCL_CH09_SELECTION_SCREEN11	
93		YCL_CH09_SELECTION_SCREEN12	
94		YCL_CH09_SELECTION_SCREEN13	
95	10	YCL_CH10_ILISTS01	
96		YCL_CH10_ILISTS02	
97		YCL_CH10_ILISTS03	
98		YCL_CH10_ILISTS04	
99		YCL_CH10_ILISTS05	
100		YCL_CH10_ILISTS06	

Serial No.	Chapter No.	Program Name	Comments
101	11	BEAD_STRINGS	Source for Method
102		SPLIT_STRING	Source for Method
103		YCL_CH11_01_BEAD_STRINGS_LCLAS	
104		YCL_CH11_02_BEAD_STRINGS_GCLAS	
105		YCL_CH11_03_SPLIT_STRING_GCLAS	
106		YCL_CH11_04_LOCAL_IF_CLASS	
107		YCL_CH11_04A_LOCAL_INH_CLASS	
108		YCL_CH11_05_USE_GCLASS_FES	
109		YCL_CH11_06_USE_CL_CTMENU	
110		YCL_CH11_07_USE_CL_CTMENU_V1	
111		YCL_CH11_08_TEST_CBE_ABAP_PG	
112		YCL_CH11_09_TEST_CBE_IN_FM	
113		YCL_CH11_10_TEST_CBE_IN_SUBR	
114		YCL_CH11_11_ABAP_TYPEDESCR	
115		YCL_CH11_FM_DEMO_CBE	Function Module
116	12	YCL_CH12_01_SIMPLE_ALV	
117		YCL_CH12_02_ALV_FCAT_ETC	
118		YCL_CH12_03_ALV_EVENTS	
119	13	YCL_CH13_01_USE_CL_SALV_TABLE	
120		YCL_CH13_02_SALV_EVENTS	
121		YCL_CH13_03_ANY_TABLE_ALV	
122		YCL_CH13_04_ANY_TABLE_SALV	
123	14	YCL_CH14_01_YCL_CH14_T005TDATA	
124		YCL_CH14_02_DEMO_SH_IMPORT	
125		YCL_CH14_03_SIMULATE_TABSTRIP	
126		YCL_CH14_04_TABSTRIP	
127		YCL_CH14_05_TABLE_CTRL	
128		YCL_CH14_06_TABSTRIP_SL_SCREEN	

The source program lines have to be transferred from E-resource text files into ABAP programs in the SAP environment.

3. Copy E-resource text files to ABAP programs in SAP environment

In the following sections: 3.1 to 3.4, the procedures for transfer of different program types: *executable*, *include*, *function modules*, *function module sub routines* and *methods* of classes has been described.

3.1. Copy source lines from text files to ABAP programs - executable and include:

You can adopt the following procedure to transfer source program (*executable* or *include*) lines from the text files into an ABAP programs in SAP environment. Navigate to the ABAP editor opening screen – transaction code SE38. Enter the name of the program to be created like for example YCL_CH06_ITAB05. Click the create button or the function key F5. The program attributes screen appears. Enter a suitable *title* for the program, select the program type as *executable program* or *include program*, click on the *save* button. Being prompted for a package, assign the package as *$TMP (local object*). The ABAP editor screen will appear as shown in figure 1:

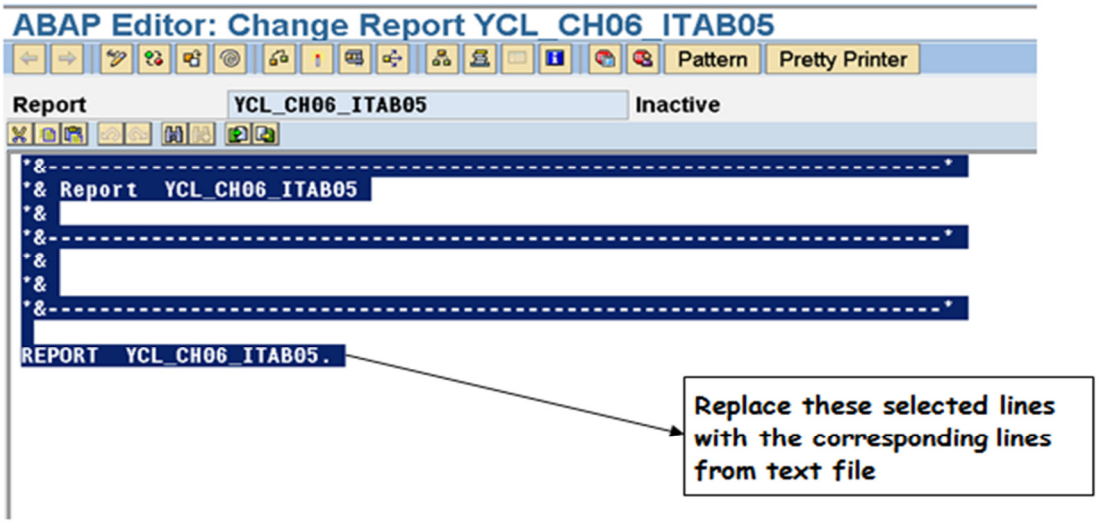

Figure 1 - ABAP Editor Screen: 7 Lines of Comments and REPORT statement

The ABAP editor system automatically generates 7 lines of comments and the REPORT statement for program type *executable*. These 7 lines of comments and the REPORT statement need to be replaced by the corresponding lines from the text file. Alternatively, you can replace only the REPORT statement of ABAP program with the corresponding REPORT statement from the text file. The idea is that the REPORT statement must originate from the text file. This is because, more frequently, the REPORT statement has additions like LINE-SIZE…, LINE-COUNT… etc.

For *include programs*, the issue of REPORT statement does not exist, the creation of *include programs* does not generate the REPORT statement.

The rest of the lines (other than the REPORT statement) from the corresponding text file can be copied into the ABAP program. Rather, to undertake the easiest course, you can replace the generated lines of the ABAP program with all of the lines from the corresponding text file. But a better alternative will be to manually enter the lines into the ABAP program using the keyboard. In this manner, you will become familiar with the syntaxes and structures of ABAP statements. After copying/manually entering all lines, you can perform syntax checking, activate the program, test etc.

You repeat this same procedure for every *executable* or *include* program.

3.2. Copy source lines from text files to function modules:

Before setting out to transfer source lines from a text file to a *function module*, you should create and activate a *function group* using the transaction code SE37. You will assign this *function group* to the *function module* you intend to create and copy source lines from text file to the *function module*. The creation of *function group* is described in chapter 7.

After creation and activation of the appropriate *function group*, on the opening screen of the transaction code SE37, enter the name of the *function module*. (Remember the name of the *function module* is same as the name of the text file) Click the create button or the function

key F5. You will be prompted for the *function group* and short text. Enter or assign the *function group* name, enter the short text and click on the save/continue button.

You will have to enter the parameters (four tabs) and exceptions. (one tab)

The parameters and exceptions (names) can be picked up from the source lines of the text file. The parameters and exceptions (names) are available in the comment lines following the statement: FUNCTION in source lines text file. For the example being demonstrated here, this is as shown in figure 2:

```
FUNCTION YCL_CH07_SPLIT_STRING.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(STRING_TO_SPLIT)
*"  TABLES
*"      STABLE
*"  EXCEPTIONS
*"      IMPORT_PARAMETER_TYPE_INVALID
*"      RETURN_TABLE_ELEMENT_NOT_TYPEC

ENDFUNCTION.
```

IMPORT Parameter: STRING_TO_SPLIT (Pass by Value)

TABLES Parameter: STABLE

Exceptions

Figure 2 – Create Function Module – Parameters & Exceptions in Comment Lines

Enter the parameters – *import*, *export*, *changing* and *tables*. There is a tab for each of the parameter types: *import*, *export*, *changing* and *tables*. When entering the parameters, remember to enable the check box for *pass by value* or *pass by value and result* if so required. Also, enable the check box for the parameters to be optional if so required. Enter the default value/s for parameter/s if required.

Next, you will enter the exceptions. Having entered the parameters and exceptions of the *function module*, if you click on the *Source code* tab, the screen will look like in figure 3:

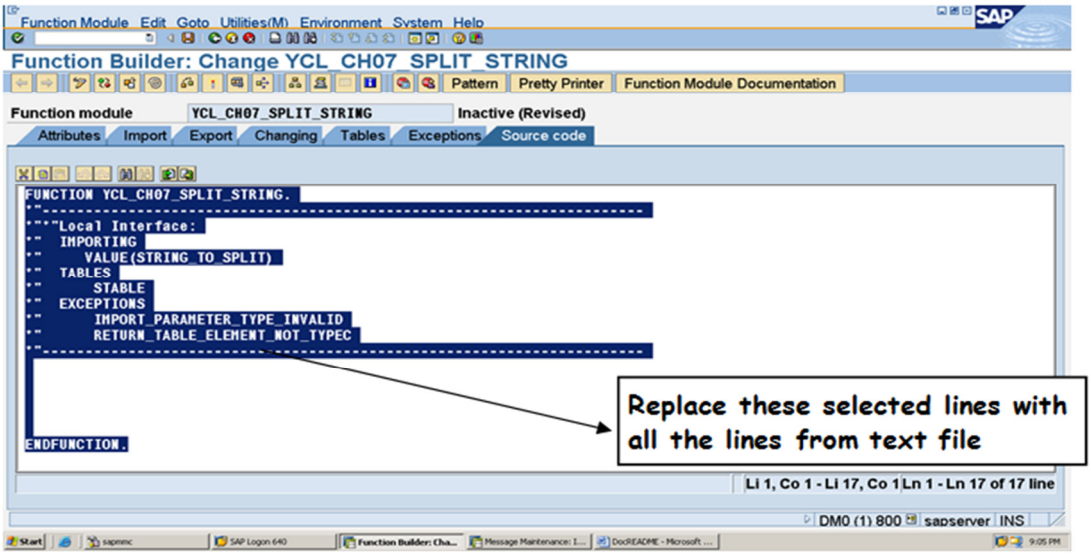

Figure 3 – Function Module – Statements FUNCTION…ENDFUNCTION with Comments

The *function module* shown in figure 3 as an example; has one *import* parameter, *passed by value* and one *tables* parameter.

The ABAP editor system generates the statements FUNCTION and ENDFUNCTION along with the comment lines as shown in figure 3. You will replace these lines with the source lines from text file. Ensure the parameter and exception names entered are identical to the names appearing in the comment lines of source text file. After copying the source lines from the text file into the *function module,* perform syntax check and activate the *function module*.

You can manually enter the source lines of the text file from the keyboard. This will be a drill to familiarize you with ABAP statements syntaxes and structures.

You can repeat this procedure for every *function module*.

Note: Locate the *function modules* CONVERSION_EXIT_ICOMA_INPUT and CONVERSION_EXIT_ICOMA_OUTPUT in the same *function group*. For all other *function modules*, use one *function group* to locate one *function module*.

3.3. Copy source lines from text file into function module sub routines

Sub routines called in *function modules* are ideally located in a separate *include program*. Sub routines created and located in an *include program* are global to the *function group*, that is, these sub routines can be called from any of the *function modules* in the *function group*. Sub routines can also be located in a *function module* following the statement ENDFUNCTION. The sub routines located following the statement ENDFUNCTION are local to the *function module* and can be called only from that *function module*.

In the present demonstrative example, we are locating the sub routines in an *include program*. In fact, we have only one *function module* invoking sub routines: CONVERSION_EXIT_ICOMA_OUTPUT. Before performing this step, you should have already created the *function modules*: CONVERSION_EXIT_ICOMA_OUTPUT and CONVERSION_EXIT_ICOMA_INPUT. You should have located these two *function modules* in the same *function group:* YCL_CH07_GROUP02.

To incorporate *function module* sub routines in an *include program*, perform the following procedure:

Navigate to the function builder opening screen – transaction code SE37. Enter the name of the *function module*: CONVERSION_EXIT_ICOMA_OUTPUT. Click the change button or the function key F6. Once, the *function module* is in change mode, make the following menu selection: Goto -> Main Program as shown in figure 4:

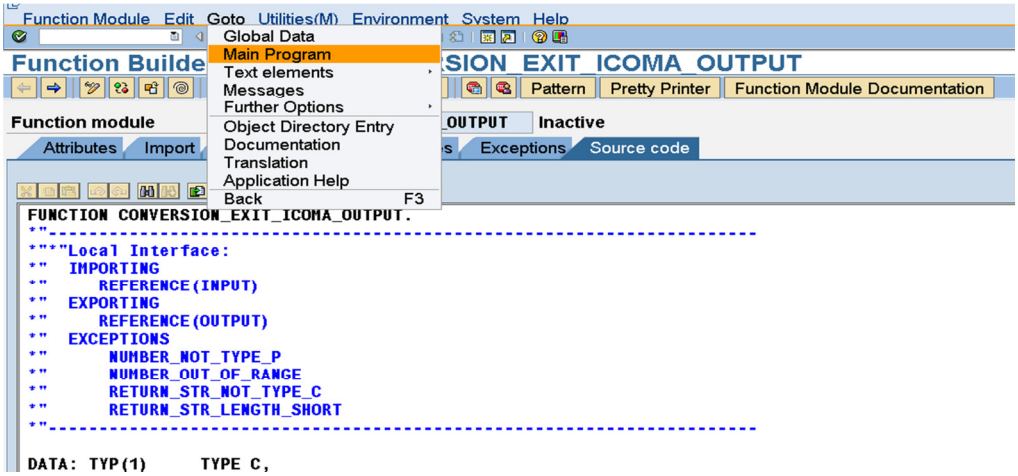

Figure 4 – Copy Sub Routines Called in Function Module I – Select Menu Option

Once you make the menu selection as shown in figure 4, you will navigate to the program SAPLYCL_CH07_GROUP02 or the main program of the *function group*. This is shown in figure 5.

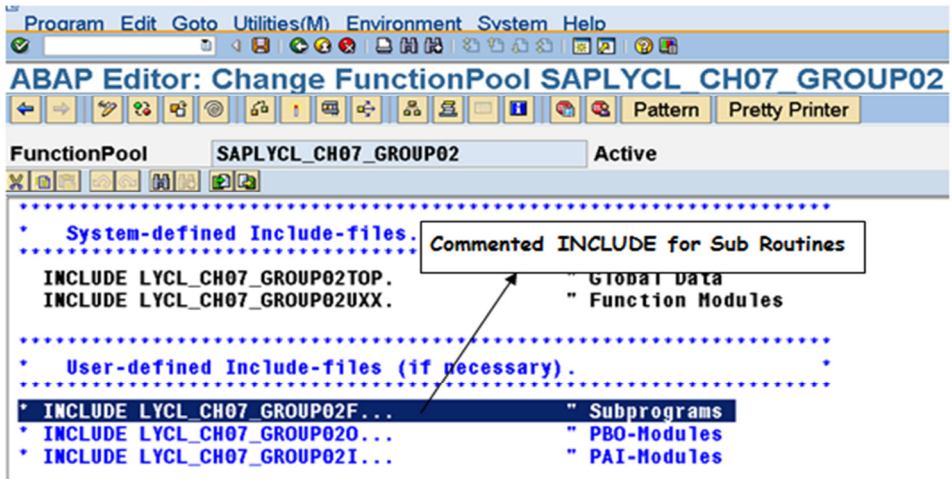

Figure 5 – Copy Sub Routines Called in Function Module II – Identify Commented Include

As shown in figure 5, there is already a provision for inclusion of an *include program* to contain the sub routines. But the statement is commented. So de-comment the statement:

- **`INCLUDE LYCL_CH07_GROUP02F…`**

The easiest way to create the *include program* is to double click on the program name. The double click will result in a prompt with the message:

Program names L… are reserved for function group includes

This message is not a pop-up; it appears on the status bar. This is shown in figure 6:

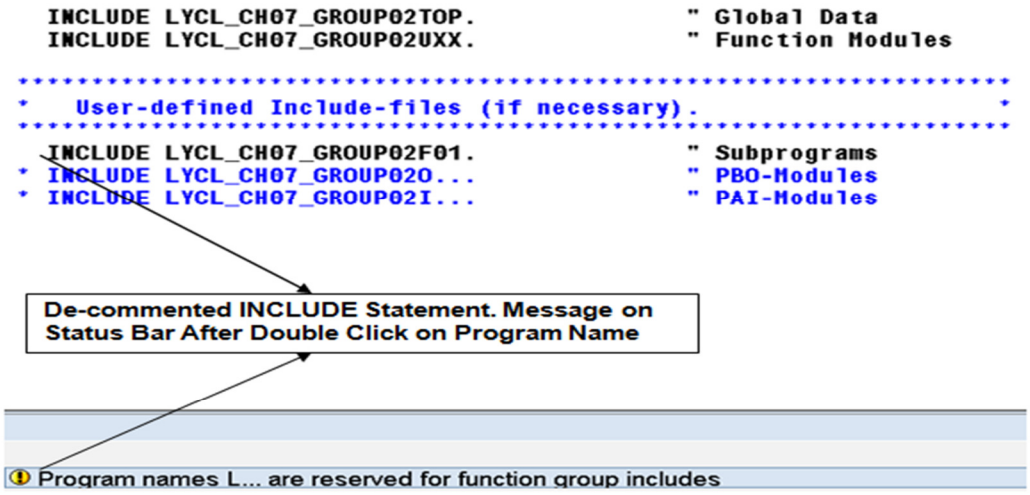

Figure 5 – Copy Sub Routines Called in Function Module III – De-comment Include etc.

Press <enter> key. The system will prompt with the pop-up message:

Include LYCL_CH07_GROUP02F01 does not exist.
Create Object?

This is shown in figure 6:

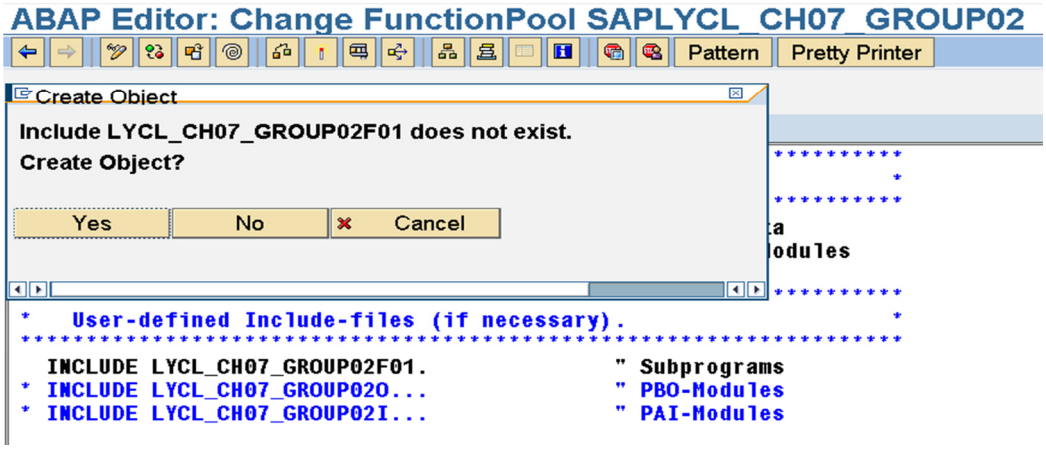

Figure 6 – Copy Sub Routines Called in Function Module IV – Prompt to Create Include Program

Press <enter> key again. The ABAP editor will generate the comment lines in the *include program* as shown in figure 7:

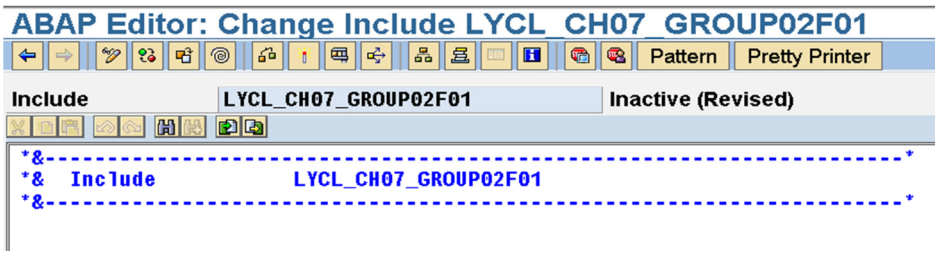

Figure 7 – Copy Sub Routines Called in Function Module V – Empty Include Program

Copy the lines from the text file into the *include program*: After the source lines have been copied from the text file, perform a syntax check and activate *the include program*. I*nclude program* after copying and activation is shown in Figure 8:

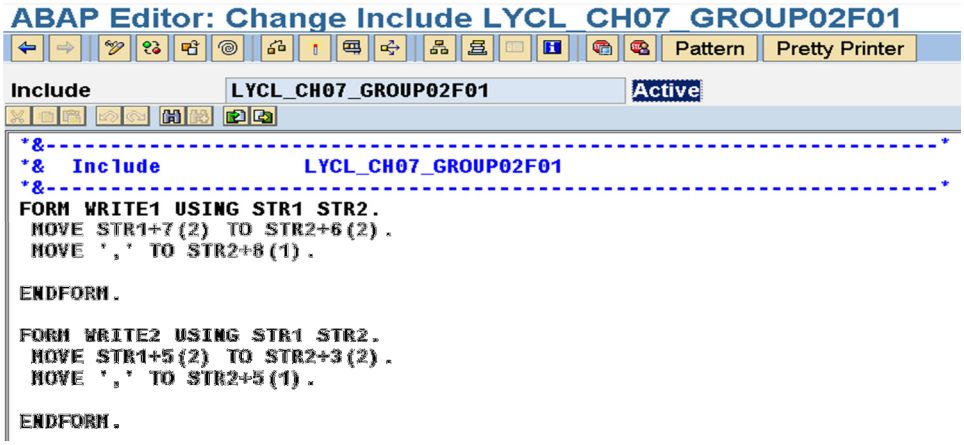

Figure 8 – Copy Sub Routines Called in Function Module VI – Lines Copied From Text File and Include Program Activated

After activation of the *include program*, press the back button to navigate back to the main program: LYCL_CH07_GROUP02. Perform syntax check and activate the main program LYCL_CH07_GROUP02. This is shown in figure 9:

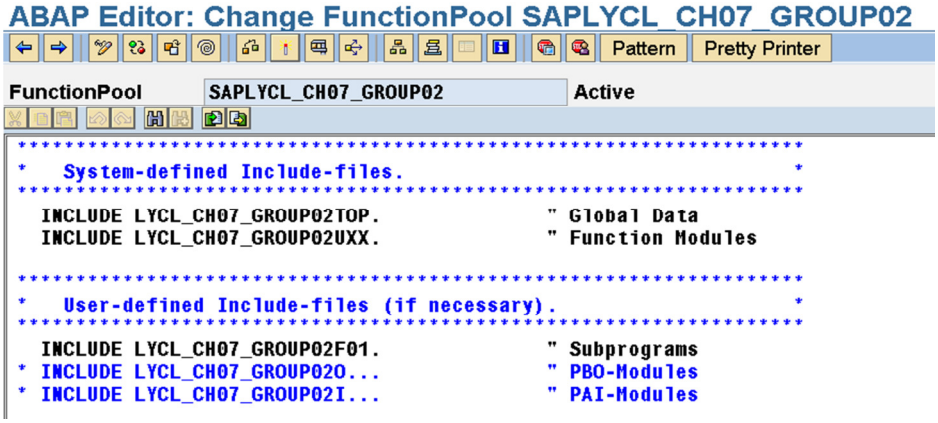

Figure 9 – Copy Sub Routines Called in Function Module VII – Main Program Activated

In the hands-on exercises, this is the only instance of sub routines called in a *function module*.

3.4. Copy source lines from text file to methods of a class

To copy source lines from text files into *methods*, you have to create the class first. The creation of the sole class YCL_CH11_CL_STRING_OPERATIONS in the class builder using the transaction code SE24 is described in chapter 11. Ensure this created class is active.

To copy lines from the text files into the *methods*, navigate to the class builder – transaction code SE24. Enter the name of the class: YCL_CH11_CL_STRING_OPERATIONS. Click the change button or press the function key F6. Click the *Methods* tab. The screen will look like in figure 10:

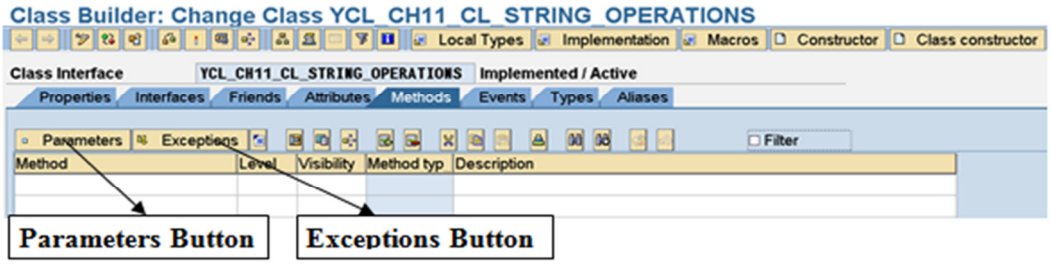

Figure 10 – Copy Text Lines into Method I – Methods Tab

Under the column *Method*, enter the *method* name as BEAD_STRINGS, under the column *Level*; enter Instance; under the column *Visibility*; enter Public. Enter a suitable short description under the column *Description*. The screen with the entered values is shown in figure 11:

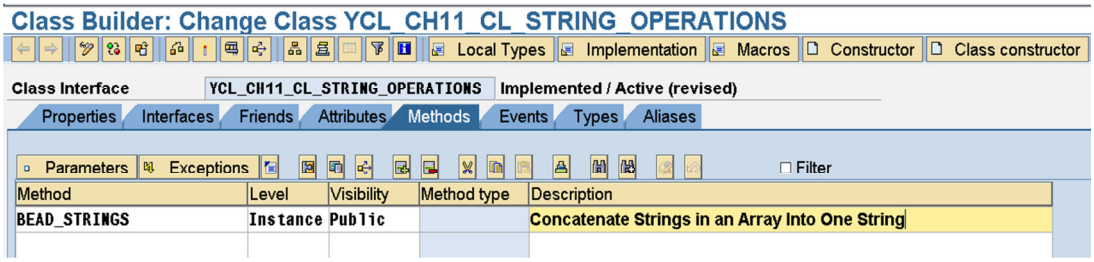

Figure 11 – Copy Text Lines into Method II – Entered Values for the Method

Next, click on the Parameters button to enter the *method*'s parameters. You can refer to the comments lines of the source text file for parameters and their particulars. Parameters for the *method* BEAD_STRINGS are to be entered as follows:

STR_TAB	**Importing**	**Pass by reference – check box off**	**TYPE**	**STRING_TABLE**
SEPARATOR	**Importing**	**Pass by value – check box on**	**TYPE**	**C**
STRNG	**Returning**	**pass by value & result – check box on**	**TYPE**	**STRING**

Note that the single output parameter of this *method* is a returning parameter, that is, the *method* is a *function method* returning a single value. The screen with the *method* parameters entered will look like in figure 12:

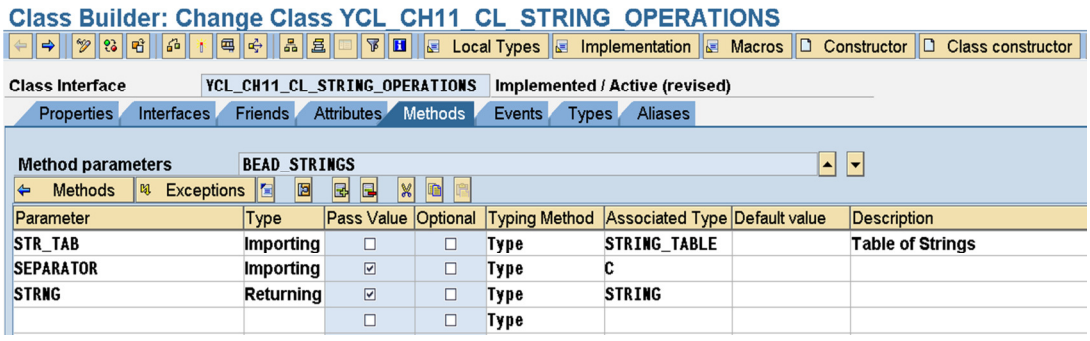

Figure 12 – Copy Text Lines into Method III – Entered Parameters for the Method

The exceptions need to be entered, so click on the exceptions button. Enter the name of the single exception NO_DATA as shown in the figure 13.

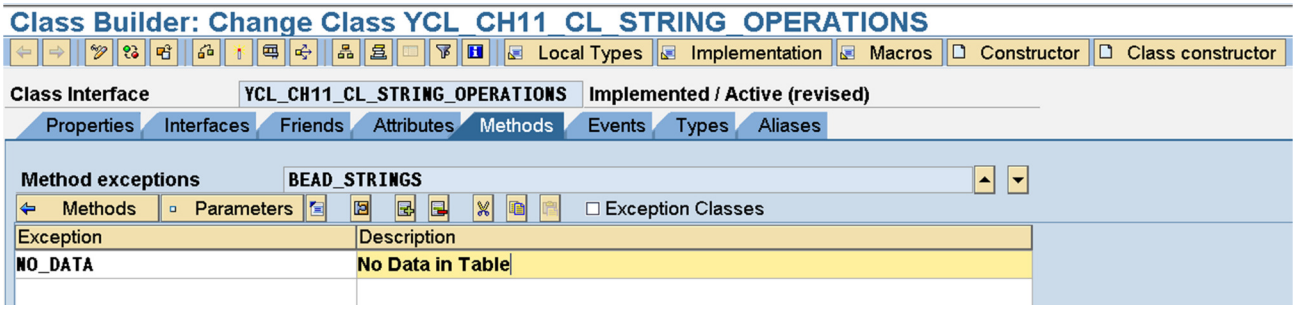

Figure 13 – Copy Text Lines into Method IV – Entered Exception for Method

After entering the exception; save the class. Click on the *Methods* tab. Double click on the *method* name BEAD_STRING. The ABAP editor will generate the *method* template as shown in figure 14:

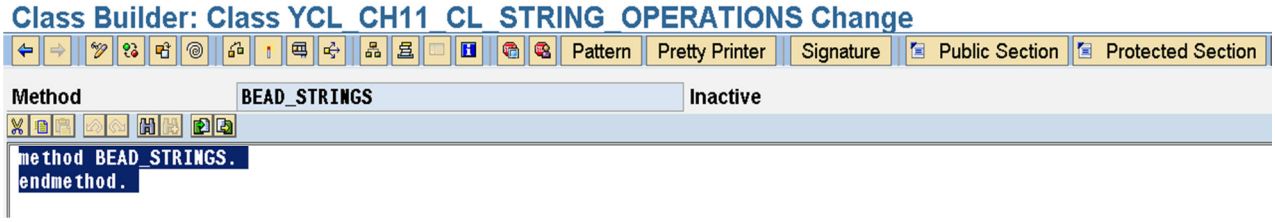

Figure 14 – Copy Text Lines into Method V – Generated Method Template

Replace the two lines of code from the lines of text file. Perform a syntax check. Activate the class YCL_CH11_CL_STRING_OERATIONS. The screen with lines transferred from text file into the *method* and the activated class is shown in figure 15:

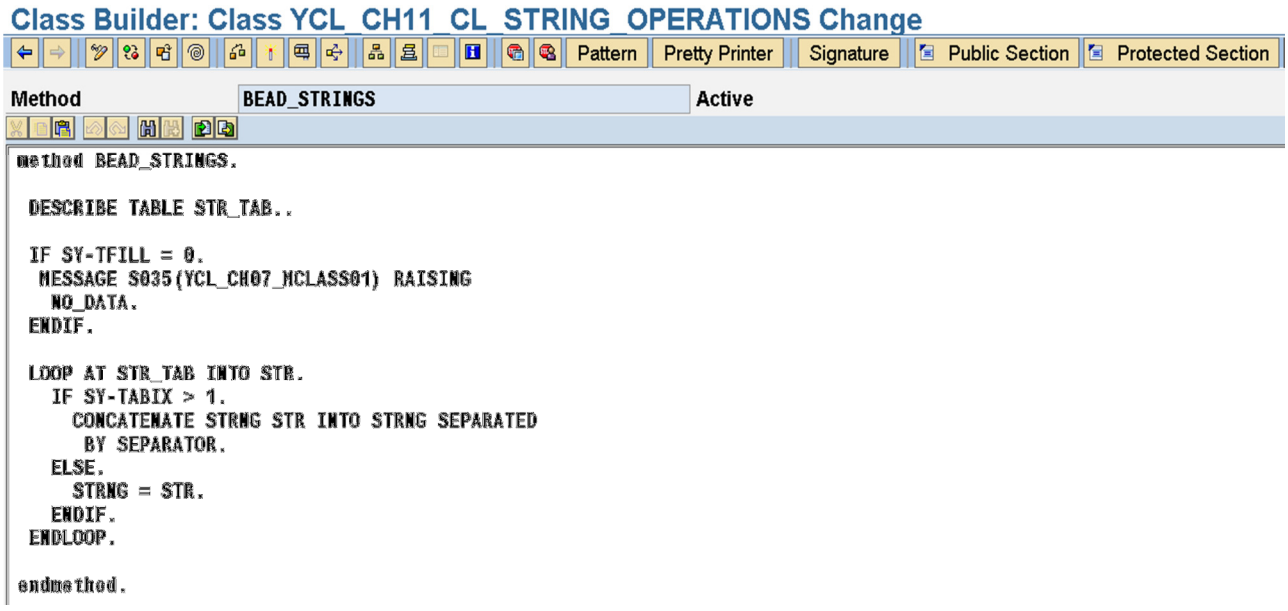

Figure 15 – Copy Text Lines into Method V – Lines Copied from Text File & Class Activated

You can repeat this procedure to transfer lines from text file into the other *method* SPLIT_STRING of the class YCL_CH11_STRING_OPERATIONS.

4. Conclusion

We could have provided a utility ABAP program to upload all the hands-on programs from the text files into ABAP programs in the SAP environment at one go, saving you of the effort of copying and pasting individually, the hundred odd programs. But that would make your

task frivolously simple. In fact, we recommend that you manually enter from the keyboard the individual lines of programs as far as possible. The process of manually entering from the keyboard, the program lines, will familiarize you with the ABAP syntaxes and statement structures.

One of the features of the book is that a number of programs created in earlier chapters are enhanced, additional features added to these programs in the succeeding chapters. So there will be situations where you are performing a hands-on exercise involving a program which is an extension of program created earlier. If you are manually entering the program from the keyboard, you can avoid manually re-entering from the keyboard the lines you entered earlier. You could just manually enter from the keyboard the lines you did not enter in the earlier program. You can copy and paste the lines from the earlier program. It is in this context that in the chapters, you are being instructed to copy lines from programs of earlier chapters.

We hope you are able to use the E-Resource to perform the hands-on exercises of the book quickly and effectively.