

Examples on OOPS in ABAP



SUBHENDU MAJUMDAR

Technical Consultant, IBM

PREFACE

This documentation contains examples on Object Oriented Programming in ABAP. It covers the individual sections in detail and contains examples illustrating the important concepts. Examples are positive or negative - the positive examples demonstrates implementation of concepts in program , the negative examples create compilation errors which shows the do's and don't's while coding for a class/object in ABAP.

Care has been taken to use simple examples which spawns not more than one page. Discussion on an example is categorised under four major heads:-

- ❖ **Theme** :- This section explains what the example is going to demonstrate.
- ❖ **Program Description** :- This section introduces to the program - briefly detailing the components of the program and what it is trying to achieve.
- ❖ **Dump**:- Contains code dump.
- ❖ **Output** :- Shows the output of the program when executed or shows the compilation errors for negative examples.

The best way to learn anything is learning by examples. The entire content has been designed and documented in such a way that the reader can easily grasp the matter and implement it in his course of learning.

The author will remain grateful to the responsible readers if they can point out mistakes in the documentation and suggest further improvements on this effort.

- Subhendu Majumdar

INDEX

1	Class	5
1.1	Accessibility of different sections of a class	5
1.2	Subclass cannot access the private component of superclass.....	8
1.3	External users cannot access protected/private components of a class.....	9
1.4	Local Class can understand data and types in the global area of the program.....	10
1.5	Class can be instantiated within implementation of another class	12
1.6	Deferred Definition of a Class	13
1.7	Place to put non-declarative statements	14
1.8	Use of Field Symbols in Class	15
1.9	Use of Static Attributes	16
1.10	Creation of Global class and using it in a local program	17
2	Methods	22
2.1	Method with one import parameter/ only one non-optional parameter	22
2.2	Import parameters passed by ref. can't be changed inside the method	23
2.3	Use of PREFERRED PARAMETER in a method	24
2.4	Use of EXPORT and CHANGING parameters of a method	25
2.5	Method using Internal Table as one of the parameters	26
2.6	Use of RETURNING parameters in method	27
2.7	Demo on Static Method	28
2.8	Static methods can only use static attributes, instance methods use both	29
2.9	Method Raising Exceptions	30
2.10	Method can call itself	31
2.11	Use of ME in methods	32
2.12	Pointer Tables	33
2.13	Dynamic Method Calls	34
2.14	Use of parameter table	35
2.15	Use of Exception Table	36
3	Constructors	37
3.1	Instance Constructors get fired at the time of class instantiation	37
3.2	Instance Constructors can have import parameters	38
3.3	Constructors cannot have any export parameters	39
3.4	Instance Constructors can raise exceptions	40
3.5	Use of static constructor	41
3.6	Static constructor can be triggered at the beginning of a processing block(form /event/block/procedure).....	42
3.7	Static/Class constructors cannot have any interface.....	43
4	Inheritance	44
4.1	Subclass can access public/protected components of superclass	44
4.2	Subclass can re-implement inherited methods from superclass.....	46
4.3	Objects cannot be created from an abstract class	47
4.4	Abstract methods cannot be implemented in abstract class.....	48
4.5	Final classes cannot have any subclass	49
4.6	Final methods cannot be redefined in the subclasses	50
4.7	Static attributes exist only once per inheritance tree	51
4.8	Constructors of superclass flows down the chain	52
4.9	Subclass can have enhanced constructor than its superclass.....	53
4.10	Static constructor of a class is called only once per program.	55

4.11	Static type and Dynamic type of a variable	56
4.12	Static type should be more general than dynamic type of a reference variable ...	58
4.13	Method of a parent class, used from its subclass, uses attributes of the parent class only, if the method is not re-defined in subclass.	59
4.14	Demo on Widening Cast	60
5	Interface.....	61
5.1	Simple use of an interface.....	61
5.2	Interfaces can only be implemented in the public section of a class	62
5.3	A class with an interface should implement all the methods of that interface.....	63
5.4	Values for interface attributes are assigned at the time of inclusion in a class...	64
5.5	Use of FINAL methods from Interface.....	65
5.6	Use of Abstract methods from Interface	66
5.7	Use of Interface Reference Variable	67
5.8	Use of Nested Interface	69
5.9	Using ALIASES	70
5.10	Polymorphism via Interfaces	71
6	Friendship.....	72
6.1	Friendship between Classes	72
6.2	Subclasses of friends can also become friends.	73
6.3	Friendship is one sided	74
7	Events.....	75
7.1	Events with Handler Method in the same class	75
7.2	Event with event handler method in different class	76
7.3	More than one event handler method can exist for same event.....	77
7.4	Use of static event	79
7.5	Events with export parameters	80
8	Class-Based Exceptions.....	81
8.1	Using SAP provided exception class	81
8.2	When both superclass and subclass are used to track error	82
8.3	Propagation of Class-Based exceptions in procedures to the caller	83
8.4	Program can raise exceptions based on SAP standard exception-classes.....	84
8.5	Objects are created from exception classes when error is trapped	85
8.6	Demo on Locally Defined Exception-Class	86
8.7	Nested TRY...ENDTRY block	87
8.8	Use of CLEANUP section.....	88
9	BADIs (Business Add-Ins)	89
9.1	Single Implementation of BADI	89
9.2	Multiple Implementation.....	93
9.3	Searching for BADI in SAP Transaction and Implementing it	96
9.4	Menu Enhancements	98

1 Class

1.1 Accessibility of different sections of a class	
Theme	<p>From this program, one will learn:-</p> <ol style="list-style-type: none"> 1. How to define, implement and instantiate a class. 2. What are the different sections of visibility in a class. 3. How to define instance attributes and get them accessed by external users. <p>The following program will also show that :-</p> <ul style="list-style-type: none"> ❖ Data declared in public section can be accessed by the class itself, by its subclasses as well as by other users outside the class. ❖ Data declared in the protected section can be accessed by the class itself, and also by its subclasses but not by external users outside the class. ❖ Data declared in the private section can be accessed by the class only, but not by its subclasses and by external users outside the class.
Brief Description	<p>This program contains a class : parentclass with following attributes in different sections:-</p> <p>Commondata in public section Protectdata in private section Privatedata in private section</p> <p>The method showval in class : parentclass displays values of all the attributes. This demonstrates that class can access all its attributes.</p> <p>Class childclass is a subclass of class parentclass, which has a method : subval.</p> <p>It displays the value for the data : commondata and protectdata . Then, it changes the values for both and displays them again. This demonstrates that subclass can access/change public/protected attributes of superclass.</p> <p>In the START-OF-SELECTION event, object : parent is instantiated from class : parentclass and object : child is instantiated from class : childclass.</p> <p>Then , the method showval of parent(object of parentclass) and method subval of child(object of childclass) is called , which displays the values of different attributes.</p>

Then, the public attribute of object parent is changed and the changed value is displayed.

This demonstrates that external users can change/display public attributes of a class.

Dump of the program:-

REPORT YSUBDEL LINE-SIZE 120.

```
CLASS parentcl ass DEFINITION .
PUBLIC SECTION.
  DATA : commondata(30) type c value 'Accessible to all'.
  METHODS : SHOWVAL.
PROTECTED SECTION.
  DATA : protectdata(40) type c value 'Protected data'.
private section.
data : privatedata(30) type c value 'Private data'.
ENDCLASS.
```

```
CLASS parentcl ass IMPLEMENTATION.
METHOD : SHOWVAL.
  write:/5 'All data from parentcl ass shown: -'.
  write:/ sy-uline.
  WRITE:/5 COMMONDATA,
        /5 PROTECTDATA,
        /5 PRIVATEDATA.
endmethod.
endcl ass.
```

```
CLASS chi l dcl ass DEFINITION INHERITING FROM parentcl ass.
PUBLIC SECTION .
METHODS : subval .
ENDCLASS.
```

```
CLASS chi l dcl ass IMPLEMENTATION.
method : subval .
  skip 1.
  write:/5 'Data of parent shown from child-'.
  write:/5 sy-uline.
  WRITE:/5 COMMONDATA,
        /5 PROTECTDATA.
  Commondata = 'Public data changed in subclass'.
  Protectdata = 'Protected data changed in subclass'.
  write:/5 sy-uline.
  WRITE:/5 COMMONDATA,
        /5 PROTECTDATA.
endmethod.
endcl ass.
```

START-OF-SELECTION.

```
DATA : parent type ref to parentclass ,  
      child type ref to childclass .  
create object : parent ,  
              child .  
call method : parent->showval ,  
              child->subval .
```

skip 2.

```
parent->commondata = 'User changing public data'.  
write: /5 parent->commondata.
```

Output

All data from parentclass shown:-

Accessible to all

Protected data

Private data

Data of parent shown from child-

Accessible to all

Protected data

Public data changed in subclass

Protected data changed in subclass

User changing public data

1.2 Subclass cannot access the private component of superclass	
Theme	The program demonstrates that subclasses cannot access the private components of superclass.
Program description	<p>The program used here is similar to above with change in the method : subval of class : childclass . This method is now attempting to access the attribute : privatedata , which is a private attribute of its superclass : parentclass.</p> <p>On compilation, the program will give a compilation error. This demonstrates that private components of superclass cannot be accessed by subclasses.</p>
Program Dump	<p>Take the first program. Only change the method : subval of class : childclass as follows:-</p> <pre>method : subval. skip 1. write:/5 'All data from parent class shown by subclass'. write:/5 sy-uline. WRITE:/5 COMMONDATA, /5 PROTECTDATA, /5 privatedata. endmethod.</pre>
Output	<p>The program will not compile. It will show an error message:-</p> <pre>Program YSUBDEL The field "PRIVATEDATA" is unknown, but there is a field with the similar name "PROTECTDATA".</pre>

1.3 External users cannot access protected/private components of a class	
Theme	This program will demonstrate that external users cannot access the protected and private components of a class
Program Description	<p>In this program , class C1 has three attributes declared in different sections as follows:-</p> <ul style="list-style-type: none"> ❖ Commondata in public section ❖ Protectdata in private section ❖ Privatedata in private section <p>In the main program, an object , OBJ1 is created from class C1 and an incorrect attempt is made to display the protected and private attribute of class C1 using its object OBJ1.</p> <p>Compilation of this program produces an error. This demonstrates : protected and private components of a class cannot be accessed by external users.</p>
Dump	<pre>REPORT YSUBDEL LINE-SIZE 120. CLASS c1 DEFINITION . PUBLIC SECTION. DATA : commondata(30) type c value 'Accessible to all'. PROTECTED SECTION. DATA : protectdata(40) type c value 'Protected data'. private section. data : privatedata(30) type c value 'Private data'. ENDCLASS. CLASS c1 IMPLEMENTATION. endclass. START-OF-SELECTION. DATA : obj1 type ref to c1. create object : obj1. write: /5 obj1->protectdata , obj1->privatedata.</pre>
Output	On compilation, an error will be generated which will prove that protected and private components of a class cannot be accessed by external users.

1.4 Local Class can understand data and types in the global area of the program.	
Theme	<p>This program will demonstrate the following:-</p> <ul style="list-style-type: none"> ❖ Different attributes of a class can be constructed utilizing the data and types declared outside the class, in the global area of the program. ❖ Data declared in the global area of the program can be used directly in a class.
Program description	<p>The global section of this program contains a type : TYP_TAB and an integer variable , NUM1.</p> <p>These type and data are used while defining attributes L_NUM1(integer) and IT_TAB (internal table) for class C1 . Also, the global data L_NUM is used directly inside the program.</p> <p>This demonstrates the theme.</p>
Dump	<pre> REPORT YSUBDEL1 LINE-SIZE 120. TYPES : BEGIN OF TYP_TAB , NAME(15) TYPE C , AGE TYPE I , END OF TYP_TAB . DATA : num1 type i value 5 . CLASS c1 DEFINITION . public section. methods : meth1 . DATA : l_num like num1 , it_tab type standard table of typ_tab , w_tab like line of it_tab. ENDCLASS. CLASS c1 IMPLEMENTATION. method : meth1 . data : l_cnum(2) type c. l_num = 0. do 5 times. l_num = l_num + 1. l_cnum = l_num. concatenate 'Student-' l_cnum into w_tab-name. w_tab-age = num1 * l_num . append w_tab to it_tab. clear w_tab. enddo. loop at it_tab into w_tab. write:/5 w_tab-name , w_tab-age. endl oop. endmethod. endclass. START-OF-SELECTION. DATA : obj1 type ref to c1. create object : obj1. call method obj1->meth1. </pre>

Output	Student-1	5
	Student-2	10
	Student-3	15
	Student-4	20
	Student-5	25

1.5 Class can be instantiated within implementation of another class	
Theme	This program will demonstrate that an object can be created from a class(which was created with no CREATE PRIVATE PROTECTED option at the time of its definition) in the implementation section of another class.
Program Description	<p>This program contains two classes - CLASS1 and CLASS2 .</p> <p>Class CLASS1 contains method : METHOD1 which displays value of some integer variable.</p> <p>Class CLASS2 contains method : METHOD2 . In the method implementation , an object is created from class : CLASS1 and then that object is used to call method METHOD1.</p> <p>This demonstrates that object can be created from a class(CLASS1) within implementation section of another class(CLASS2).</p>
	<pre> REPORT YSUBOOPS17 . class class1 definition. public section. methods : method1 . endclass. class class2 definition. public section. methods : method2 . endclass. class class1 implementation. method :method1. data : i_num type i value 2. write:/5 i_num. endmethod. endclass. class class2 implementation. method : method2. data : obj1 type ref to class1. create object obj1. call method obj1->method1. endmethod. endclass. start-of-selection. data : my_obj type ref to class2. create object : my_obj. call method my_obj->method2. </pre>
Output	2

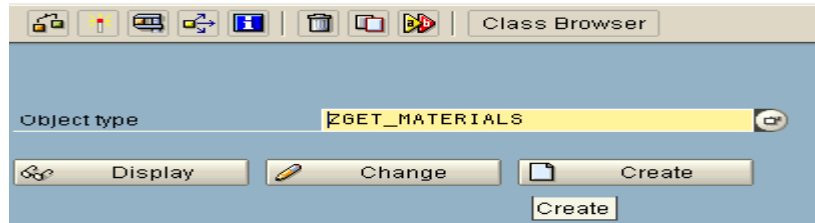
1.6 Deferred Definition of a Class	
Theme	This program will demonstrate how one can refer to a class without defining the class before that point. But, the class has to be defined later on.
Program description	<p>In this program , class C1 has an interface reference O2 declared with reference to class C2. But, before that, class C2 is not defined. It is defined later with a single public attribute , NUM .</p> <p>This demonstrates the theme.</p> <p>In the main section of the program, object : OBJ1 is created from class C1. Then, an object is created from the reference variable O2 in class C1. Finally, the attribute num of that object is displayed.</p>
Dump	<pre> report ysubdel 1. CLASS C2 DEFINITION DEFERRED. CLASS C1 DEFINITION. PUBLIC SECTION. DATA O2 TYPE REF TO C2. ENDCLASS. CLASS C2 DEFINITION. public section. data : num type i value 5. ENDCLASS. start-of-selection. data : obj1 type ref to C1. CREATE OBJECT obj1. create object obj1->o2. write: /5 obj1->o2->num . </pre>
Output	5

1.7 Place to put non-declarative statements		
Theme	For a class, the IMPLEMENTATION section can immediately follow the class DEFINITION section. If this is so, then all the non-declarative statements (viz., processing statements outside any class definition/ implementation) should be placed under a processing block, such as START-OF-SELECTION.	
Program description	<p>This program contains a class C1 with a method M1.</p> <p>In version 1, the IMPLEMENTATION part of the class follows the class definition section. But, the non-declarative statements are not placed under any block. This creates a compilation error.</p> <p>In version 2, the non-declarative statements are not placed under the block START-OF-SELECTION. It gets correctly compiled.</p> <p>This demonstrates the theme.</p>	
	<pre>REPORT YSUBDEL . class c1 definition. public section. methods : m1 . endclass. class c1 implementation. method : m1 . write:/5 'I am method m1'. endmethod. endclass. data : obj1 type ref to c1 . create object obj1. call method obj1->m1.</pre>	<pre>REPORT YSUBDEL . class c1 definition. public section. methods : m1 . endclass. class c1 implementation. method : m1 . write:/5 'I am method m1'. endmethod. endclass. START-OF-SELECTION. data : obj1 type ref to c1 . create object obj1. call method obj1->m1.</pre>
	Version 1: Incorrect	Version 2 : Correct
Output	Version 1 creates compilation error. Version 2 gets correctly compiled.	

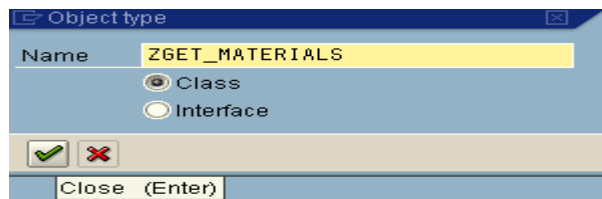
1.8 Use of Field Symbols in Class	
Theme	Field Symbols can be used to contain value of any variable in a class.
Program Description	The program uses a field symbol, <FS>. It handles the values of instance attribute, inum and static attribute , onum .
Dump	<p>REPORT YSUB_ASSIGN_FS.</p> <p>FIELD-SYMBOLS : <FS> TYPE ANY .</p> <p>class c1 definition . public section . * Instance attribute : inum declared below data : inum type i value 5 . * static attribute onum declared below class-data : onum type i value 10 . endclass.</p> <p>class c1 implementation. endclass.</p> <p>start-of-selection. data : oref1 type ref to c1 . create object oref1. * Assigning instance attribute to field symbol <fs> assign oref1->inum to <fs> . write:/5 <fs> . * Assigning static attribute to field symbol assign oref1->onum to <fs> . write:/5 <fs> . assign c1=>onum to <fs> . write:/5 <fs> .</p>
Output	5 10 10

1.9 Use of Static Attributes	
Theme	This program will demonstrate that : Static sttributes of a class are retained throughout the entire runtime. All the objects within a class can access its static attributes.
Program Description	<p>The program contains a class C1 with static attribute : NUM . The method : M1 increments the static attribute by 1 and displays the value each time it is called.</p> <p>In the main START-OF-SELECTION portion, two objects : OBJ1 and OBJ2 are created from class C1.</p> <p>First, static attribute : NUM is changed and accessed outside the class using the class component selector , '=>'. Then, both objects OBJ1 and OBJ2 are used to call method : M1 which shows the new value of static attribute : NUM .</p> <p>That the value of the static attribute gets incremented each time when the method M1 of different objects is called shows that this variable is able to retain its value through the entire runtime.</p>
Dump	<pre> report ysubdel . CLASS c1 DEFINITION . PUBLIC SECTION. CLASS-DATA : NUM TYPE I . METHODS : M1. ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD m1 . num = num + 1. write: /5 num . ENDMETHOD. ENDCLASS. START-OF-SELECTION. c1=>num = 3. write: /5 c1=>num . DATA : OREF1 TYPE REF TO C1 , OREF2 TYPE REF TO C1 . CREATE OBJECT : OREF1 , OREF2 . CALL METHOD OREF1->M1 . CALL METHOD OREF2->M1. </pre>
Output	3 4 5

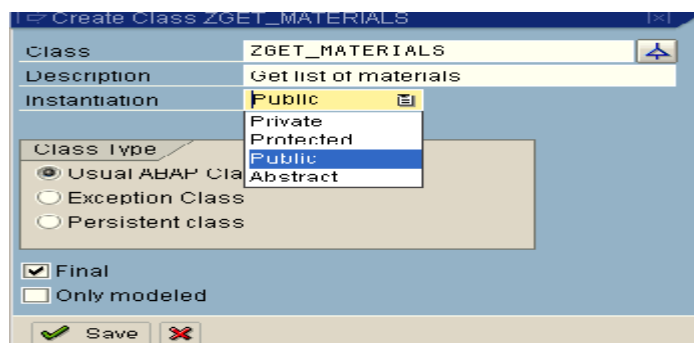
1.10 Creation of Global class and using it in a local program	
Theme	This example will show you how to create a class globally and use it in your local program
Program Descr.	<p>There is a demand to create a global class which will furnish a list of materials along with their descriptions based on the material type.</p> <p>Global class ZGET_MATERIALS will be created. Method LIST_MATERIALS will belong to this class which will take material type as an input and will furnish a list of material codes and their descriptions belonging to that material type.</p>
Steps	Follow the steps outlined below to perform the task

Step 1. Create the class from SE24:-**Class Builder: Initial Screen**

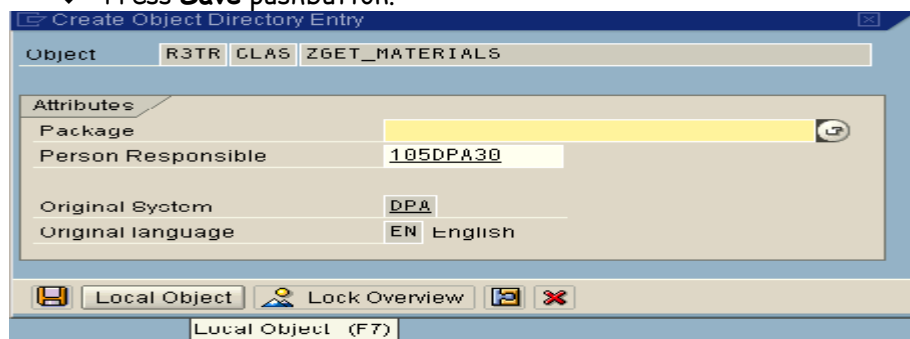
- ❖ Go to transaction SE24.
- ❖ Enter the name of the global class you want to create, with 'Y' or 'Z' at the beginning.
- ❖ Press Create pushbutton.



- ❖ A dialog window shown above will appear. Check the radiobutton : **Class**.
- ❖ Press Enter.



- ❖ Another dialog window shown above will appear. Enter the description for the class.
- ❖ Select from the **Instantiation** listbox whether you want to create the class as PUBLIC/PROTECTED/PRIVATE/ABSTRACT.
- ❖ Check the radiobutton for **Usual ABAP Class**.
- ❖ Check the checkbox for **Final**.
- ❖ Press **Save** pushbutton.

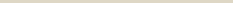


Enter the **package name** or **save** it as Local object.

Step 2 : Create the method : LIST_MATERIALS

Methods	Level	Vis...	Mo...	M...	Description
LIST_MATERIALS	Insta...	Pub...	<input type="checkbox"/>		Provide list of materials with material code and name

- ❖ Go to the tab-page : **Methods**.
- ❖ Enter the details for the method - mention name, type of method(instance/static), in which visibility section the method will reside and a short description of the method.
- ❖ Check -off/uncheck the checkbox to ensure that the method will be implemented.

Parameters Exceptions  Filter

Methods	Level	Vis...	Mo...	M...	Description
Params					
LIST_MATERIALS	Insta...	Pub...	<input type="checkbox"/>		Provide list of materials with material code and name

Click the pushbutton for **Parameters** to navigate to the screen to enter parameters for the method.

Parameter	Type	P...	O...	Typing ...	Associated Type	Default value	Description
L_MTART	Import...	<input type="checkbox"/>	<input type="checkbox"/>	Type	MTART		Material Type
MATERIAL_LIST	Export...	<input type="checkbox"/>	<input type="checkbox"/>	Type	TABLE		List of material code and names

There will be one importing parameter : **L_MTART** and one exporting internal table : **MATERIAL_LIST**. Create entries for them as shown above.

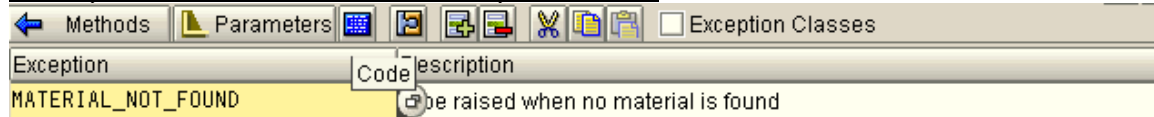
Parameter	Type	P...	O...	Typing ...	Associated Type	Default value	Description
_MTART	Import...	<input type="checkbox"/>	<input type="checkbox"/>	Type	MTART		Material Type

Click the pushbutton : **Exceptions** to make entry for Exceptions to be raised by the method.

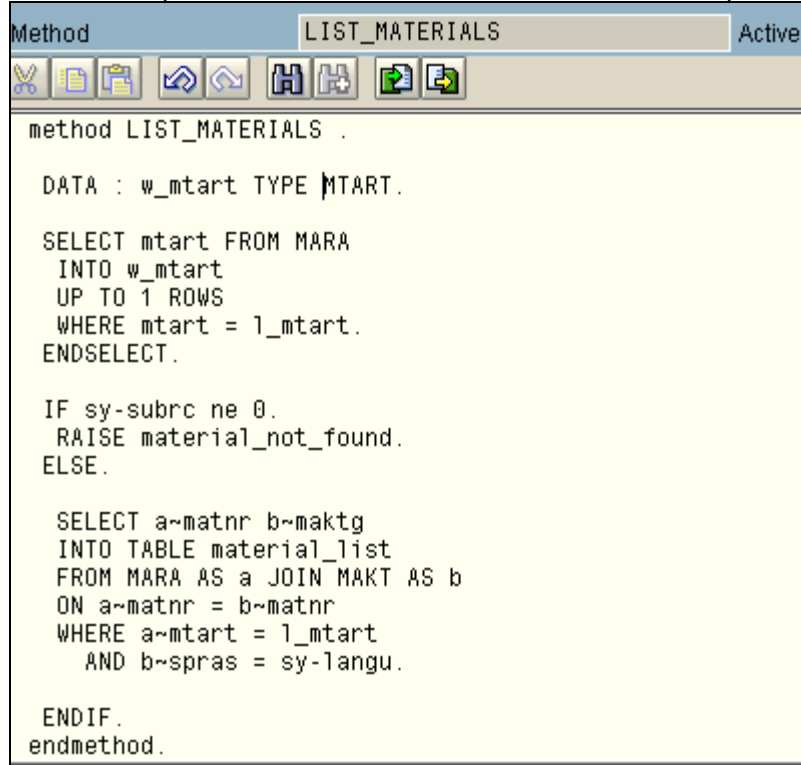
Exception	Description
material_not_found	to be raised when no material is found

Enter the name and description for the exception .

Then, **check** and **activate** the class.

Step 3: Write code for method implementation

Click on the pushbutton : Code(blue colored button) to implement the method.



An ABAP Editor will open up. Write the logic for code implementation.
Then, check and activate the code.

Your job of creating a global class is complete!!!

Step 4 : Use the global class created by you in a local program

```
REPORT YSUBDEL.
```

```
TYPES : BEGIN OF typ_mat ,
         matnr LIKE mara-matnr ,
         maktg LIKE makt-maktg ,
       END OF typ_mat .
```

```
DATA : it_mat TYPE STANDARD TABLE OF typ_mat ,
       x_mat LIKE LINE OF it_mat.
```

```
PARAMETERS : p_mtart LIKE mara-mtart OBLIGATORY.
```

```
START-OF-SELECTION.
```

* **Create object from the global class**

```
DATA : oref TYPE REF TO zget_materials.
CREATE OBJECT oref.
```

* **Call the method to get list of material code and name**

```
CALL METHOD oref->list_materials
  EXPORTING l_mtart      = p_mtart
  IMPORTING material_list = it_mat
  EXCEPTIONS
    material_not_found = 2.
```

```
if sy-subrc ne 0.
  write: /5 'Material not found'.
else.
```

* **Display the list**

```
  loop at it_mat into x_mat.
    write: /5 x_mat-matnr ,
           x_mat-maktg.
  endloop.
endif.
```

Output

Compile and run the program. There will be a parameter for material type in the selection screen. Enter a valid value and get the list of material codes and descriptions.

2 Methods

2.1 Method with one import parameter/ only one non-optional parameter	
Theme	<p>This program will demonstrate different ways of calling a method which has only one import parameter.</p> <p>This strategy is also valid for cases where a method has more than one import parameters, but only one of them being non-optional.</p>
Program	<p>This program has a class , C1 with a method : meth1. This method has only one import parameter(input1). Look at the method implementation for details. The main purpose of this program is to demonstrate the different ways of calling a method with single import parameter.</p>
Dump	<pre>REPORT YSUBDEL . CLASS C1 DEFINITION. PUBLIC SECTION. DATA : NUM TYPE I VALUE 5. METHODS : METH1 IMPORTING INPUT1 TYPE I . ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD : METH1. num = NUM * INPUT1 . WRITE: /5 NUM . num = 5. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1. CREATE OBJECT : OREF1. * Different ways of calling the method with one import parameter CALL METHOD OREF1->METH1 EXPORTING INPUT1 = 4. CALL METHOD OREF1->METH1(INPUT1 = 5). CALL METHOD OREF1->METH1(6).</pre>
Output	<p>20</p> <p>25</p> <p>30</p>

2.2 Import parameters passed by ref. can't be changed inside the method .	
Theme	<p>Parameters can be passed to a method as import parameters in two fashion:-</p> <ul style="list-style-type: none"> ❖ By reference ❖ By value. <p>Parameters passed by value can be changed internally in a method. But, parameters passed by reference cannot be changed in the method.</p>
Program description	<p>This program contains a class C1 with a method METH1. This method contains two input parameters : -</p> <p>INPUT1 : passed by reference INPUT2 : passed by value.</p> <p>The method METH1 attempts to change INPUT1. On compilation, an error is displayed. This establishes that input parameters passed by reference cannot be changed within the method.</p>
Dump	<pre> REPORT YSUBDEL . DATA : num TYPE I. CLASS C1 DEFINITION. PUBLIC SECTION. METHODS : METH1 IMPORTING INPUT1 TYPE I value(input2) type i . ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD : METH1. Input1 = 4. write:/5 input1. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1. CREATE OBJECT : OREF1. num = 3. CALL METHOD OREF1->METH1 EXPORTING INPUT1 = 4 input2 = num. </pre>
Output	<p>On compilation, an error message is generated.</p> <p>Now, instead of changing input1, change the import parameter input2 (passed by value) within the method. The program will get successfully compiled and executed.</p>

2.3 Use of PREFERRED PARAMETER in a method	
Theme	<p>If there are more than one OPTIONAL import parameters in a method and no non-optional import parameters without values, one can type in the clause PREFERRED PARAMETER after the list of import parameters to specify which of the optional parameters will get more preference compared to others when the method will be called using syntax :- CALL METHOD objref->meth(<val>).</p> <p>In other words, it decides which of the optional parameters will be assigned the value 'VAL'.</p>
Program Description	<p>This program contains a class C1 containing method METH1 which has two optional parameters , INPUT1 and INPUT2. Out of them, parameter INPUT2 is declared as preferred parameter. The method simply displays the value of two import parameters .</p> <p>Notice the last line of the program and see the output. The output will establish that the preferred parameter INPUT2 gets the value passed to the method when it is called using the syntax:- CALL METHOD objref->meth(<val>).</p>
Dump	<pre> REPORT YSUBDEL . CLASS C1 DEFINITION. PUBLIC SECTION. METHODS : METH1 IMPORTING INPUT1 TYPE I optional input2 TYPE I OPTIONAL PREFERRED PARAMETER INPUT2. ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD : METH1. write:/5 input1 , /5 input2 . ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1. CREATE OBJECT : OREF1. CALL METHOD : OREF1->METH1(input1 = 5 input2 = 3). skip 2. write:/5 'Next call'. call method oref1->meth1(10) . </pre>
Output	<pre> 5 3 Next call 0 10 </pre>

2.4 Use of EXPORT and CHANGING parameters of a method			
Theme	This program will demonstrate the use of EXPORTING and CHANGING parameters of a method.		
Program description	The program contains a method TAX_CALC belonging to the class CTAX . It receives GRADE as IMPORTING parameter and SALARY as CHANGING parameter. Based on the grade, the EXPORTING parameter ITAX is calculated and the CHANGING parameter, SALARY is modified by deducting tax from it.		
	<pre> REPORT YSUBDEL . DATA : w_tax type p decimals 2 , w_salary type p decimals 2 . CLASS CTAX DEFINITION. PUBLIC SECTION. METHODS : TAX_CALC IMPORTING grade TYPE C EXPORTING itax TYPE P CHANGING salary TYPE P . ENDCLASS. CLASS CTAX IMPLEMENTATION. METHOD : TAX_CALC. CASE grade. WHEN 'A01'. itax = salary * '0.2'. WHEN 'A02'. itax = salary * '0.1'. WHEN OTHERS. itax = salary * '0.15'. ENDCASE. salary = salary - itax. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : OREF1 TYPE REF TO CTAX. CREATE OBJECT : OREF1. w_salary = 30000. w_tax = 0 . write:/5 'Before method call, salary and tax are' , w_salary , w_tax . CALL METHOD OREF1->TAX_CALC EXPORTING grade = 'A01' IMPORTING itax = w_tax CHANGING salary = w_salary. write:/5 'After method call, salary and tax are' , w_salary , w_tax . </pre>		
Output	Before method call, salary and tax are	30,000.00	0.00
	After method call, salary and tax are	24,000.00	6,000.00

2.5 Method using Internal Table as one of the parameters	
Theme	This program demonstrates how an internal table can be used as one of the interface parameters of a method.
Program Description	The program contains a method : GETMARA in class : GET_MATERIALS . It accepts material group, MATGR as import parameter and details out the details of the materials belonging to that material group into I_TAB , which is an internal table used as EXPORTING parameter for the method.
	<pre> REPORT YSUBOOPS5 . types : begin of typ_tab , matnr like mara-matnr , meins like mara-meins , end of typ_tab . data : itab type standard table of typ_tab , x_tab LIKE LINE OF ITAB. CLASS get_materials DEFINITION. PUBLIC SECTION. METHODS : getmara IMPORTING matgr TYPE C EXPORTING I_tab TYPE ANY TABLE. endclass. CLASS get_materials IMPLEMENTATION. METHOD : getmara . SELECT matnr meins INTO TABLE I_tab FROM MARA WHERE MATKL = matgr. ENDMETHOD. ENDCLASS. PARAMETERS : p_matkl like mara-matkl . START-OF-SELECTION. DATA : w_mat TYPE REF TO get_materials. CREATE OBJECT : w_mat. CALL METHOD w_mat->getmara EXPORTING matgr = p_matkl IMPORTING I_tab = itab . LOOP AT ITAB INTO X_TAB. WRITE:/5 X_TAB-MATNR , X_TAB-MEINS. ENDLOOP. </pre>
Output	One/more than one records with material number and basic unit, depending on the material group entered in the selection-screen.

2.6 Use of RETURNING parameters in method	
Theme	<p>To get some values from a method , one can use the EXPORTING, CHANGING or RETURNING parameters.</p> <p>If one uses RETURNING parameters, the following restrictions apply:-</p> <p>(1) No EXPORTING/CHANGING parameters can be used for the method.</p> <p>(2) Only one RETURNING parameter can be used.</p> <p>(3) RETURNING parameters are only passed by value.</p> <p>This program demonstrates the use of RETURNING parameters and the various ways to call a method with RETURNING parameter to get the value into some variable.</p>
Program Description	<p>Method M1 in class C1 have two input parameters(INPUT1 and INPUT2), which are used to derive value for RETURNING parameter , RESULT.</p> <p>The program demonstrates various syntaxes that can be used to call a method of this kind.</p>
	<pre> report ysubdel1 message-id 00. data : w_num type i. class c1 definition . public section. methods : m1 importing input1 type i input2 type i returning value(result) type i . endclass. class c1 implementation. method : m1. result = input1 * 2 + input2. endmethod. endclass. start-of-selection. data : obj1 type ref to c1 . create object obj1. * Syntax 1 call method obj1->m1 EXPORTING input1 = 5 input2 = 4 RECEIVING result = w_num. write:/5 w_num . * Syntax 2 w_num = obj1->m1(input1 = 10 input2 = 20). write:/5 w_num . * Syntax 3 move obj1->m1(input1 = 2 input2 = 3) to w_num . write:/5 w_num . </pre>
Output	14 40 7

2.7 Demo on Static Method	
Theme	This program will show how to declare and define a static method and how it can be called using class component selector.
	In the following program, method : TESTMETHOD is defined as static method and is called later using class component selector, '=>'. .
Dump	<pre> REPORT YSUBOOPS19 data : num type i. class testclass definition. public section. class-methods : testmethod. endclass. class testclass implementation. method : testmethod. num = 5. write:/5 num. endmethod. endclass. start-of-selection. call method testclass=>testmethod.</pre>
Output	5

2.8 Static methods can only use static attributes, instance methods use both

Theme	Static methods of a class can only use static attributes of that class. It cannot use instance attributes. But, instance methods can use both.		
Program Description	The following program contains a class C1 which contains the following:-		
	Component	Type	Static/Instance
	stnum	Data	static
	Instnum	Data	Instance
	Stmeth	Method	Static
	Instmeth	Method	Instance
Both the static and instance methods are attempting to display values of the static and instance attributes: STNUM and INSTNUM .			
On compilation, an error will be generated which will demonstrate that static method STMETH cannot work with instance attribute, INSTNUM .			
Dump	<pre>REPORT YSUBDEL. CLASS C1 DEFINITION. PUBLIC SECTION. CLASS-DATA : STNUM TYPE I VALUE 5. DATA : INSTNUM TYPE I VALUE 6 . CLASS-METHODS : STMETH . METHODS : INSTMETH . ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD : STMETH . WRITE: /5 STNUM . WRITE: /5 INSTNUM . ENDMETHOD. METHOD INSTMETH. WRITE: /5 STNUM . WRITE: /5 INSTNUM . ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1. CALL METHOD c1=>stmeth . CREATE OBJECT OREF1. CALL METHOD oref1->instmeth.</pre>		
Output	<p>On compilation,you get the following error:-</p> <p>Program YSUBDEL</p> <p>Within a static method, you can only access class attributes without further specifications.</p> <p>Remove the line in bold in the program and compile. It will get successfully compiled and executed.</p>		

2.9 Method Raising Exceptions	
Theme	Methods can raise exceptions like function modules which can be handled after calling the method, depending on various sy-subrc values. This program will demonstrate that.
Program description	<p>The program provides the user a selection-screen where the user enters a numeric value. If the user entry is <5, he gets an information message 'Should be >=5'. Else, five times of the value entered is displayed by the program on execution.</p> <p>The class C1 in this program contains method M1 which imports value for NUM1 , and returns five times of it through the export parameter, NUM2. However, if the value passed to NUM1 is lesser than 5, it raises an exception E1 with some error message.</p> <p>The method M1 is called after creating an object from the class. User-entered value in the parameter field : P_NO is passed to importing parameter NUM1.</p>
Dump	<pre> report ysubdel1 message-id 00. class c1 definition . public section. methods : m1 importing num1 type i exporting num2 type i exceptions e1. endclass. class c1 implementation. method : m1. if num1 lt 5 . message i398(00) with 'Should be >=5' raising e1. else . num2 = num1 * 5 . endif. endmethod. endclass. parameters : p_no type i . start-of-selection. data : obj1 type ref to c1 . create object obj1. call method obj1->m1 exporting num1 = p_no importing num2 = p_no exceptions e1 = 1. IF sy-subrc <> 0. MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4. ELSE. write:/5 p_no . ENDIF. </pre>
Output	The program provides the user a selection-screen where the user enters a numeric value. If the user entry is <5, he gets an information message 'Should be >=5'. Else, five times of the value entered is displayed by the program on execution.

2.10 Method can call itself	
Theme	Method of a class can call itself .But, please do not forget to specify some exit point to the method. Else, it will create an infinite loop.
Program Descr.	The program following contains a method M1 in a class C1 . It increases the value of static variable, STATNUM by 10 and displays it. Then, if the value of STATNUM is <=100 , it calls itself again.
Dump	<pre> report ysubdel1 message-id 00. class c1 definition . public section. class-data : statnum type i . methods : m1 . endclass. class c1 implementation. method : m1. statnum = statnum + 10. if statnum gt 100. exit. endif. write: /5 statnum . call method m1. endmethod. endclass. start-of-selection. data : obj1 type ref to c1 . create object obj1. call method obj1->m1 . </pre>
Output	<pre> 10 20 100 </pre>

2.11 Use of ME in methods	
Theme	<p>A method can have a variable defined within it having the same name as one of the attributes of the class to which the method belongs to.</p> <p>To clearly identify the class level attribute, the selector ME is used.</p>
Program Description	<p>Class TESTCLASS contains method TESTMETHOD. There is a variable I_NUM declared as public attribute in the class as well as in the implementation part of the method.</p> <p>To access the variable I_NUM at the class level within the method, the selector ME is used. Please see the outputs of this program for better understanding.</p>
Dump	<pre>REPORT YSUBOOPS17 . class testclass definition. public section. data : i_num type i value 5. methods : testmethod . endclass. class testclass implementation. method : testmethod. data : i_num type i value 2. write:/5 me->i_num , " access variable of the class /5 i_num . " access variable of the method endmethod. endclass. start-of-selection. data : i_num type i. data : my_obj type ref to testclass. create object : my_obj. call method my_obj->testmethod.</pre>
Output	<pre>5 2</pre>

2.12 Pointer Tables	
Theme	This program will demonstrate the use of pointer tables
Program Description	The program below uses pointer table : MYOBJ_TAB .
	<pre> REPORT YSUB00PS19 class testclass definition. public section. methods : testmethod . class-data : num type i. endclass. class testclass implementation. method : testmethod. num = num + 5. write: /5 num. endmethod. endclass. start-of-selection. data : myobj type ref to testclass , myobj_tab type table of ref to testclass. do 5 times. create object myobj . append myobj to myobj_tab. enddo. loop at myobj_tab into myobj . call method : myobj ->testmethod. endloop. </pre>
Output	<pre> 5 10 15 20 25 </pre>

2.13 Dynamic Method Calls	
Theme	<p>One can call methods dynamically. Following restrictions apply:-</p> <ul style="list-style-type: none"> ❖ Name of the method can be dynamic for static/instance method. ❖ Name of the class while calling static method can be dynamic. ❖ Both the name of the static method and the class containing it can be dynamic. <p>While doing so, it is better to use uppercase to assign the name of the class/methods to the variables(which will be used for dynamic assignment)</p>
Program Description	The following program contains class C1 with static method(STATM) and dynamic method(INSTM). The program utilizes all the syntaxes to call these static/instance methods dynamically.
Dump	<pre>REPORT YSUBOOPS19 . data : f(6) type c , g(10) type c . class c1 definition. public section. class-methods : statm . methods : instm . endclass. class c1 implementation. method : statm . write:/5 'I am static method'. endmethod. method : instm. write:/5 'I am instant method'. endmethod. endclass. start-of-selection. data : oref type ref to c1. create object oref. * Name of instance method can be dynamic f = 'INSTM'. call method oref->(f). * Name of static method can be dynamic f = 'STATM'. call method oref->(f). * Name of the class can be dynamic for static method call f = 'C1'. call method (f)=>statm. * Name of the method can be dynamic for static method call f = 'STATM'. call method c1=>(f). * Both can be dynamic for static method call g = 'C1'. call method (g)=>(f).</pre>
Output	<pre>I am instant method I am static method I am static method I am static method I am static method</pre>

2.14 Use of parameter table	
Theme	For dynamic method call, one can use the concept of PARAMETER TABLE to include references of all interface parameters of the method (instead of mentioning them separately).
Prog. Descr.	<p>The program contains class C1 with a method M1. This method has one IMPORTING parameter, 'P1' and one EXPORTING parameter 'P3'. P3 is calculated within the method.</p> <p>Here, parameter table PTAB is used to dynamically call the method M1. Please note that, parameter tables work only when a method is called dynamically (i.e. to day, the name of the method/class is determined at runtime).</p>
Dump	<pre> REPORT YSUB00PS24 . DATA : i_result type i, i_num type i value 5 . DATA f(2) TYPE c VALUE 'M1' . CLASS cl_abap_objectdescr DEFINITION LOAD . DEFINE : poptable. ptab_line-name = &1. ptab_line-kind = CL_ABAP_OBJECTDESCR=>&2. GET REFERENCE OF &3 INTO ptab_line-value. INSERT ptab_line INTO TABLE ptab. IF sy-subrc ne 0. EXIT. ENDIF. END-OF-DEFINITION. CLASS c1 DEFINITION. PUBLIC SECTION. METHODS m1 IMPORTING p1 TYPE i EXPORTING p3 type i . ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD m1. p3 = p1 + 200. ENDMETHOD. ENDCLASS. DATA r TYPE REF TO c1. DATA: ptab TYPE abap_parmbind_tab, ptab_line LIKE LINE OF ptab. START-OF-SELECTION. poptable : 'P1' EXPORTING i_num , 'P3' IMPORTING i_result . CREATE OBJECT r TYPE c1. CALL METHOD r->(f) PARAMETER-TABLE ptab. write:/5 i_result . </pre>
Output	205

2.15 Use of Exception Table	
Theme	Instead of dealing with each and every exception and assigning it to different sy-subrc values, one can use exception table to handle the exceptions when a method is called dynamically.
	The class C1 contains method M1 which raises an exception. Exception table ETAB is used to handle the exception.
	<pre> REPORT YSUBOOPS25 . CLASS cl_abap_objectdescr DEFINITION LOAD. CLASS c1 DEFINITION. PUBLIC SECTION. METHODS m1 EXCEPTIONS exc. ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD m1. RAISE exc. ENDMETHOD. ENDCLASS. DATA r TYPE REF TO object. DATA f(3) TYPE c VALUE 'M1'. DATA: etab TYPE abap_excpcbind_tab, etab_line LIKE LINE OF etab. START-OF-SELECTION. etab_line-name = 'EXC'. etab_line-value = 4. INSERT etab_line INTO TABLE etab. IF sy-subrc ne 0. EXIT. ENDIF. CREATE OBJECT r TYPE c1. CALL METHOD r->(f) EXCEPTION-TABLE etab. WRITE sy-subrc. </pre>
Output	4

3 Constructors

3.1 Instance Constructors get fired at the time of class instantiation	
Theme	This simple program will show you that instance constructor methods of a class get triggered when an object is created from the class.
Program Description	This program contains a class C1 with a constructor method which writes out something to indicate that it is triggered. In the START-OF-SELECTION block, the class C1 is instantiated, which triggers the instance constructor method (as is evident by the output as report). This establishes the theme.
Dump	<pre> REPORT YSUB00PS1. CLASS C1 DEFINITION. PUBLIC SECTION. METHODS : CONSTRUCTOR . ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD constructor. WRITE:/5 'I am constructor'. skip 2. ENDMETHOD. ENDCLASS. ***** main program ***** START-OF-SELECTION. DATA: obj1 TYPE REF TO c1. CREATE OBJECT: obj1. </pre>
Output	I am constructor

3.2 Instance Constructors can have import parameters

Theme	Instance constructors can have import parameters. Values to them are passed at the time of <code>CREATE OBJECT</code> statement to create object from the class containing the constructor.
Program Description	The program contains a class C1 which has one instance constructor with one import parameter. The constructor gets fired at the time of <code>CREATE OBJECT</code> statement.
Dump	<pre> REPORT YSUB00PS2. CLASS c1 DEFINITION. PUBLIC SECTION. METHODS : CONSTRUCTOR importing today type d. ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD constructor. Write:/5 'Today is : ' , today dd/mm/yyyy. ENDMETHOD. ENDCLASS. ***** main program ***** START-OF-SELECTION. DATA: obj1 TYPE REF TO c1. CREATE OBJECT: obj1 exporting today = sy-datum. </pre>
Output	Today is 08.04.2004

3.3 Constructors cannot have any export parameters	
Theme	This program will demonstrate that constructor methods cannot have any export parameters
Program Description	This program attempts to create a constructor with export parameter, which is trapped and resisted at the time of compilation. This establishes the theme.
Dump	REPORT YSUB00PS2. CLASS c1 DEFINITION. PUBLIC SECTION. METHODS : CONSTRUCTOR exporting name type c. ENDCLASS.
Output	Compilation error is reported.

3.4 Instance Constructors can raise exceptions	
Theme	Instance Constructor methods can raise exceptions
Program Descriptions	This program contains a class C1 which contains an instance constructor method. It accepts an import parameter, NUM . If it is lesser than 7, an exception is raised, which is properly handled by specifying some sy-subrc value at the time of CREATE OBJECT statement and later handled properly.
Dump	<pre> REPORT YSUB00PS2. CLASS c1 DEFINITION. PUBLIC SECTION. METHODS : CONSTRUCTOR importing num type i exceptions e1 . ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD constructor. if num lt 7. raise e1. endif. ENDMETHOD. ENDCLASS. ***** main program ***** START-OF-SELECTION. DATA: obj1 TYPE REF TO c1. CREATE OBJECT: obj1 exporting num = 5 exceptions e1 = 2. if sy-subrc = 2. write: /5 'Exceptions raised' . endif. </pre>
Output	Exceptions Raised

3.5 Use of static constructor.

Theme	<p>There are two programs over here which will show you simple example on a static constructor. You will learn how to declare a static constructor. Static/class constructors get triggered before any of the following events:-</p> <ul style="list-style-type: none"> ❖ Generating an instance of a class using CREATE OBJECT obj, where obj has the data type REF TO class. ❖ Calling a static method using [CALL METHOD] class=>meth. ❖ Registering a static event handler method using SET HANDLER class=>meth for obj. ❖ Registering an event handler method for a static event of the class class. ❖ Addressing a static attribute with class=>a. <p>These two programs will show you that a class constructor gets fired before any of its static components are accessed, or an object is created from the class.</p>	
Dump	<pre>REPORT YSUB00PS2. CLASS c1 DEFINITION . PUBLIC SECTION. CLASS-DATA : NUM TYPE I VALUE 5. CLASS-METHODS : CLASS_CONSTRUCTOR. ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD CLASS_CONSTRUCTOR. WRITE: /5 'I am class constructor'. ENDMETHOD. ENDCLASS. START-OF-SELECTION. WRITE: /5 C1=>NUM.</pre>	<pre>REPORT YSUB00PS2. CLASS c1 DEFINITION . PUBLIC SECTION. CLASS-DATA : NUM TYPE I VALUE 5. CLASS-METHODS : CLASS_CONSTRUCTOR. ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD CLASS_CONSTRUCTOR. WRITE: /5 'I am class constructor'. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : OREF TYPE REF TO C1. CREATE OBJECT OREF.</pre>
	<p>Constructor is triggered when a static attribute is accessed</p>	<p>Constructor is fired when an object is created from the class</p>
Output	<p>For the first program(on the LHS):- I am class constructor 5 For the second program(on the RHS):- I am class constructor</p>	

3.6 Static constructor can be triggered at the beginning of a processing block(form /event/block/procedure)	
Theme	In the START-OF-SELECTION block of this program, static attribute of a class containing class constructor will be accessed. This will demonstrate that the the first thing which will get executed in the START-OF-SELECTION block is the class constructor method, irrespective of the point at which the static attribute is accessed.
Program Description	<p>The program contains a class C1 with a static constructor, which prints the statement:- " I am class constructor".</p> <p>This class also contains a static attribute , num of value = 5.</p> <p>The START-OF-SELECTION block in this program contains the following statements:-</p> <p>A write statement which will print :-"Hello"</p> <p>A call to access the static attribute(NUM) of class C1.</p> <p>On execution of this program, we will get the following output in the list:-</p> <p>I am class constructor</p> <p>Hello</p> <p>5</p> <p>instead of the expected output:-</p> <p>Hello</p> <p>I am class constructor</p> <p>5</p> <p>This demonstrates the theme.</p>
Dump	<pre> REPORT YSUB00PS2. CLASS c1 DEFINITION . PUBLIC SECTION. CLASS-DATA : NUM TYPE I VALUE 5. CLASS-METHODS : CLASS_CONSTRUCTOR. ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD CLASS_CONSTRUCTOR. WRITE: /5 'I am class constructor'. ENDMETHOD. ENDCLASS. START-OF-SELECTION. write: /5 'Hello' . write: /5 c1=>num. </pre>
Output	<p>I am class constructor</p> <p>Hello</p> <p>5</p>

3.7 Static/Class constructors cannot have any interface	
Theme	This program will show you that static constructors of a class cannot have any interface parameters and exceptions
Program Description	In this program, the class C1 contains a class constructor which is having an import parameter , NUM . The program could not be successfully compiled due to such attempt.
Dump	<pre>REPORT YSUBOOPS2. CLASS c1 DEFINITION . PUBLIC SECTION. CLASS-METHODS : CLASS_CONSTRUCTOR IMPORTING NUM TYPE C. ENDCLASS.</pre>
Output	Compilation of the program will fail with an error message:- "The method CLASS_CONSTRUCTOR may not have parameters or EXCEPTIONS" .

4 Inheritance

4.1 Subclass can access public/protected components of superclass

Theme	<p>This program will demonstrate:-</p> <ul style="list-style-type: none">❖ How to create a subclass from a superclass.❖ Subclass can access public/protected components(methods, attributes etc) of superclass.																												
Prog. Descr.	<p>This program contains superclass C1 and its subclass C2. Class C1 has the following components:-</p> <table><tr><th>Component</th><th>Nature</th><th>Section of Existence</th><th>Significance/action</th></tr><tr><td>NUM</td><td>Attribute</td><td>public</td><td>Value = 6</td></tr><tr><td>METH1</td><td>Method</td><td>Public</td><td>Displays value of NUM</td></tr><tr><td>METH2</td><td>Method</td><td>Protected</td><td>Displays:- "I am meth2"</td></tr><tr><td>NUM2</td><td>Attribute</td><td>Protected</td><td>Value = 7</td></tr></table> <p>Subclass of C1 is C2 which contains the following new components:-</p> <table><tr><th>Component</th><th>Nature</th><th>Section of Existence</th><th>Significance/action</th></tr><tr><td>M1</td><td>Method</td><td>public</td><td><ul style="list-style-type: none">➤ Calls method meth1.➤ Calls method meth2➤ Displays value of variable num2.</td></tr></table> <p>In the START-OF-SELECTION block, an object OREF is created from C2 and the method M1 is called.The output of the program shows that method M1 of class C2 calls method METH1, then METH2 and finally displays variable NUM2.</p> <p>This demonstrates that subclass C2 has access to the public and protected methods and attributes of superclass C1 and truly establishes the theme.</p>	Component	Nature	Section of Existence	Significance/action	NUM	Attribute	public	Value = 6	METH1	Method	Public	Displays value of NUM	METH2	Method	Protected	Displays:- "I am meth2"	NUM2	Attribute	Protected	Value = 7	Component	Nature	Section of Existence	Significance/action	M1	Method	public	<ul style="list-style-type: none">➤ Calls method meth1.➤ Calls method meth2➤ Displays value of variable num2.
Component	Nature	Section of Existence	Significance/action																										
NUM	Attribute	public	Value = 6																										
METH1	Method	Public	Displays value of NUM																										
METH2	Method	Protected	Displays:- "I am meth2"																										
NUM2	Attribute	Protected	Value = 7																										
Component	Nature	Section of Existence	Significance/action																										
M1	Method	public	<ul style="list-style-type: none">➤ Calls method meth1.➤ Calls method meth2➤ Displays value of variable num2.																										
Dump	<pre>REPORT YSUBDEL. CLASS C1 DEFINITION. PUBLIC SECTION. METHODS : METH1. DATA : NUM TYPE I VALUE 6. PROTECTED SECTION. DATA : num2 type i value 7. METHODS METH2. ENDCLASS. CLASS C1 IMPLEMENTATION . METHOD : METH1. WRITE:/5 num. endmethod. METHOD : METH2. WRITE:/5 ' I am meth2 '. ENDMETHOD. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. PUBLIC SECTION. METHODS : M1. ENDCLASS. CLASS C2 IMPLEMENTATION. METHOD M1. CALL METHOD : meth1, meth2. write:/5 num2. endmethod. endclass. START-OF-SELECTION. DATA : OREF TYPE REF TO C2. CREATE OBJECT OREF</pre>																												

	CALL METHOD : OREF->M1.
Output	6 I am meth2 7

4.2 Subclass can re-implement inherited methods from superclass	
Theme	Subclass can re-implement the inherited public and protected methods from superclass.
Program Descr.	<p>Class C1 contains method METH1(public) and METH2(protected), both of which are modified and re-implemented in its subclass C2. Objects are created out of both classes and the method METH1 for both objects are called.</p> <p>Output of the program demonstrates different behaviour for method METH1 of class C1 and C2.</p> <p>This demonstrates the theme.</p>
Dump	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION. PUBLIC SECTION. METHODS : METH1. PROTECTED SECTION. METHODS METH2. ENDCLASS. CLASS C1 IMPLEMENTATION . METHOD : METH1. WRITE:/5 'I am meth1 in class C1'. CALL METHOD METH2. ENDMETHOD. METHOD : METH2. WRITE:/5 'I am meth2 in class C1 '. ENDMETHOD. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. PUBLIC SECTION. METHODS : METH1 redefinition . PROTECTED SECTION. METHODS : METH2 redefinition. ENDCLASS. CLASS C2 IMPLEMENTATION. METHOD METH1. WRITE:/5 'I am meth1 in class C2'. call method meth2. endmethod. METHOD : METH2. WRITE:/5 'I am meth2 in class C2 '. ENDMETHOD. endclass. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1 , OREF2 TYPE REF TO C2. CREATE OBJECT : OREF1 , OREF2. CALL METHOD : OREF1->METH1 , OREF2->METH1. </pre>
Output	<p>I am meth1 in class C1 I am meth2 in class C1 I am meth1 in class C2 I am meth2 in class C2</p>

4.3 Objects cannot be created from an abstract class.		
Theme	Objects cannot be created from an abstract class. Only the subclasses of such class can be instantiated.	
Program Description	This program contains an abstract class C1 and its subclass C2 . Object cannot be created from class C1 , but possible from class C2 .	
Dump	<pre>REPORT YSUBDEL. CLASS C1 DEFINITION ABSTRACT. PUBLIC SECTION. ENDCLASS. CLASS C1 IMPLEMENTATION . METHOD : METH1. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. ENDCLASS. CLASS C2 IMPLEMENTATION. endclass. START-OF-SELECTION. data : OREF1 TYPE REF TO C1 , OREF2 TYPE REF TO C2. CREATE OBJECT oref1.</pre>	<pre>REPORT YSUBDEL. CLASS C1 DEFINITION ABSTRACT. PUBLIC SECTION. ENDCLASS. CLASS C1 IMPLEMENTATION . METHOD : METH1. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. ENDCLASS. CLASS C2 IMPLEMENTATION. endclass. START-OF-SELECTION. data : OREF1 TYPE REF TO C1 , OREF2 TYPE REF TO C2. CREATE OBJECT oref2.</pre>
	Instantiation of abstract class	Instantiation of subclass of an abstract class
Output	<p>Instantiation of abstract class will be resisted with error message at the time of compilation.</p> <p>Instantiation of subclass of an abstract class will be allowed.</p>	

4.4 Abstract methods cannot be implemented in abstract class		
Theme	Abstract methods cannot be implemented in that class. It has to be implemented in one of its subclass. To implement an abstract method in a subclass, one need to redefine this subclass using the REDEFINITION addition.	
Program Descr.	This program contains an abstract class C1 with abstract method METH1 , which is implemented in the same class. The program will not be compiled due to this. The program is then modified and the abstract method is implemented in class C2 , subclass of C1 . Now, the program gets successfully compiled.	
Dump	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION ABSTRACT. PUBLIC SECTION. METHODS : METH1 ABSTRACT. ENDCLASS. CLASS C1 IMPLEMENTATION . METHOD : METH1. WRITE:/5 'I am method : METH1 '. ENDMETHOD. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. ENDCLASS. CLASS C2 IMPLEMENTATION. endcl ass. START-OF-SELECTION. data : OREF2 TYPE REF TO C2. CREATE OBJECT oref2. </pre> <div>Incorrect</div>	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION ABSTRACT. PUBLIC SECTION. METHODS : METH1 ABSTRACT. ENDCLASS. CLASS C1 IMPLEMENTATION . ENDMETHOD. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. public section. methods : meth1 redefinition. ENDCLASS. CLASS C2 IMPLEMENTATION. METHOD : METH1. WRITE:/5 'I am method: METH1 '. ENDMETHOD. endcl ass. START-OF-SELECTION. data : OREF2 TYPE REF TO C2. CREATE OBJECT oref2. </pre> <div>Correct</div>

4.5 Final classes cannot have any subclass	
Theme	Subclasses cannot be inherited from a final Class. They can only be instantiated.
Program Descrip.	This program contains a final class C1 and a subclass C2 . This is not allowed and is resisted at the time of compilation. Hence, the theme is properly established.
Dump	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION FINAL. ENDCLASS. CLASS C1 IMPLEMENTATION . ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. ENDCLASS. CLASS C2 IMPLEMENTATION. endclass. START-OF-SELECTION. data : OREF2 TYPE REF TO C2. CREATE OBJECT oref2. </pre>
Output	Compilation error is generated:- The final class C1 cannot have any subclasses.

4.6 Final methods cannot be redefined in the subclasses	
Theme	Final method in a class can only be defined in that class. It cannot be redefined in any of its subclasses.
Program Descr.	<p>This program contains a class C1 which has a final method : METH1. Class C2 is subclass of C1 and tries to re-define the final method.</p> <p>Compilation error is generated , resisting successful compilation. This demonstrates the theme.</p>
Dump	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION . PUBLIC SECTION. METHODS : METH1 FINAL. ENDCLASS. CLASS C1 IMPLEMENTATION . method meth1. write:/5 'I am method meth1'. endmethod. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. PUBLIC SECTION. methods : meth1 redefi ni ti on. ENDCLASS. CLASS C2 IMPLEMENTATION. method : meth1. write:/5 ' I am meth1,modified in class C2'. Endmethod. endcl ass. START-OF-SELECTION. data : OREF2 TYPE REF TO C2. CREATE OBJECT oref2. </pre>
Output	<p>Program is not successfully compiled. The compilation error message is as follows:-</p> <p>The final method METH1 cannot be redefined.</p>

4.7 Static attributes exist only once per inheritance tree	
Theme	Static attributes only exist once in each inheritance tree. One can change them from outside the class using the class component selector with any class name, or within any class in which they are shared. They are visible in all classes in the inheritance tree.
Program Descr.	<p>Class C1 contains static attribute , NUM.</p> <p>Class C2 and C3 are subclasses of class C1.</p> <p>In the START-OF-SELECTION block, the static attribute , NUM is changed using reference of class C3.</p> <p>It gets changed with reference to class C2 also.</p> <p>Hence, static attribute, NUM, changed via class C3 is also changed with respect to C2 also. This demonstrates the theme.</p>
Dump	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION . PUBLIC SECTION. class-data : num type i. ENDCLASS. CLASS C1 IMPLEMENTATION . ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. ENDCLASS. CLASS C2 IMPLEMENTATION. endclass. CLASS C3 DEFINITION INHERITING FROM C1. ENDCLASS. START-OF-SELECTION. C3=>NUM = 10. WRITE: /5 C2=>NUM. </pre>
Output	10

4.8 Constructors of superclass flows down the chain	
Theme	Constructor of superclass is inherited by the subclass also.
Program Descr.	<p>Class C1 contains a constructor method, but its subclass C2 does not explicitly have its own.</p> <p>When object from class C2 is created, the constructor of class C1 is triggered.</p> <p>This establishes the theme.</p>
Dump	<pre>REPORT YSUB00PS18. CLASS C1 DEFINITION. PUBLIC SECTION. METHODS : CONSTRUCTOR . ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD constructor. WRITE: /5 'I am C1'. skip. ENDMETHOD. ENDCLASS. CLASS C2 DEFINITION INHERITING FROM C1. ENDCLASS. CLASS C2 IMPLEMENTATION. ENDCLASS. START-OF-SELECTION. DATA: obj type ref to C2. CREATE OBJECT: obj.</pre>
Output	I am C1.

4.9 Subclass can have enhanced constructor than its superclass.

Theme	<p>A subclass can modify the constructor method and add some extra functionalities. In the instance constructor method of the child class, the one for the superclass should be called first using :</p> <p>CALL METHOD super->CONSTRUCTOR statement and then additional statements can be added.</p> <p>Pl. note that REDEFINITION statement is not required to enhance constructors for a subclass.</p>
Program Descrip.	<p>This program contains three classes:-</p> <ul style="list-style-type: none"> ❖ GRANDFATHER at the top of the node with its own constructor method ❖ FATHER, subclass of GRANDFATHER with enhanced constructor ❖ SON, subclass of FATHER with its own enhanced constructor. <p>An object is created finally from the class SON, which triggers the constructor methods in the order: GRANDFATHER→FATHER→SON.</p>
	<pre> REPORT YSUBOOPS18. CLASS grandfather DEFINITION. PUBLIC SECTION. METHODS : CONSTRUCTOR . ENDCLASS. CLASS grandfather IMPLEMENTATION. METHOD constructor. WRITE:/5 'I am grandfather'. skip. ENDMETHOD. ENDCLASS. CLASS father DEFINITION INHERITING FROM GRANDFATHER. public section. METHODS : CONSTRUCTOR. ENDCLASS. CLASS father IMPLEMENTATION. METHOD constructor . call method super->constructor. WRITE:/5 'I am father'. skip. ENDMETHOD. ENDCLASS. CLASS son DEFINITION INHERITING FROM FATHER. public section. METHODS : CONSTRUCTOR. ENDCLASS. CLASS son IMPLEMENTATION. METHOD constructor . call method super->constructor. WRITE:/5 'I am son'. skip. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA: myson type ref to son. </pre>

	CREATE OBJECT: myson.
Output	I am grandfather I am father I am son

4.10 Static constructor of a class is called only once per program.	
Theme	The first when a subclass in a program is accessed, its static constructor is executed. But, before it can be executed, the static constructors of all of its superclasses must already have been executed. A static constructor may only be called once per program. Therefore, when one first address a subclass, the system looks for the next-highest superclass whose static constructor has not yet been executed. It executes the static constructor of that class, followed by those of all classes between that class and the subclass that is addressed.
Program Descr.	<p>This program contains three classes:-</p> <ul style="list-style-type: none"> ❖ FATHER with its own static constructor method ❖ SON, subclass of FATHER with its own static constructor method. <p>An object is created finally from the class SON, which triggers the constructor methods in the order: FATHER→SON. Now, an object is created from the class FATHER. But, that did not trigger constructor of class FATHER, because that had already been triggered by the program when an object was created from the class SON. This establishes the theme.</p>
Dump	<pre>REPORT YSUB00PS18. CLASS father DEFINITION. public section. class-METHODS : class_CONSTRUCTOR. ENDCLASS. CLASS father IMPLEMENTATION. METHOD class_constructor . WRITE: /5 'I am father' . skip. ENDMETHOD. ENDCLASS. CLASS son DEFINITION INHERITING FROM FATHER. public section. class-METHODS : class_CONSTRUCTOR. ENDCLASS. CLASS son IMPLEMENTATION. METHOD class_constructor . WRITE: /5 'I am son' . skip. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA: myson type ref to son. CREATE OBJECT: myson. data : myfather type ref to father. create object : myfather.</pre>
Output	<pre>I am father I am son</pre>

4.11 Static type and Dynamic type of a variable

Theme	Static type of a reference variable can point to a superclass; whereas its dynamic type can point to one of its subclasses. This program will show you various ways to do that.																																			
Program Descrip.	<p>This program contains class C1 with method M1. Class C2 is a subclass of C1 and contains redefined implementation of method M1. Reference variables are created in the program as follows:-</p> <table><tr><th>Reference Variable</th><th>Static type</th><th>Dynamic type</th><th>Dynamic type assigned by</th></tr><tr><td>OREF1</td><td>C1</td><td>C1</td><td>CREATE object oref1.</td></tr><tr><td>OREF11</td><td>C1</td><td>C2</td><td>CREATE OBJECT oref11 TYPE C2.</td></tr><tr><td>OREF111</td><td>C1</td><td>C2</td><td>CREATE OBJECT OREF111. OREF111 = OREF2.</td></tr><tr><td>OREF2</td><td>C2</td><td>C2</td><td>CREATE OBJECT oref2.</td></tr></table> <p>Finally, method M1 is called using all objects. The observations are as follows:-</p> <table><tr><th>Method call</th><th>Calls method</th><th>Reason</th></tr><tr><td>OREF1->M1</td><td>M1 of class C1.</td><td>Both Static & Dynamic type of OREF1 refers to C1.</td></tr><tr><td>OREF11->M1</td><td>M1 of class C2.</td><td>Static type of OREF11 refers to C1, dynamic type to C2.</td></tr><tr><td>OREF111->M1</td><td>M1 of class C2.</td><td>Static type of OREF111 refers to C1, dynamic type to C2.</td></tr><tr><td>OREF2->M1</td><td>M1 of class C2</td><td>Both Static & Dynamic type of OREF2 refers to C2.</td></tr></table>	Reference Variable	Static type	Dynamic type	Dynamic type assigned by	OREF1	C1	C1	CREATE object oref1.	OREF11	C1	C2	CREATE OBJECT oref11 TYPE C2.	OREF111	C1	C2	CREATE OBJECT OREF111. OREF111 = OREF2.	OREF2	C2	C2	CREATE OBJECT oref2.	Method call	Calls method	Reason	OREF1->M1	M1 of class C1.	Both Static & Dynamic type of OREF1 refers to C1.	OREF11->M1	M1 of class C2.	Static type of OREF11 refers to C1, dynamic type to C2.	OREF111->M1	M1 of class C2.	Static type of OREF111 refers to C1, dynamic type to C2.	OREF2->M1	M1 of class C2	Both Static & Dynamic type of OREF2 refers to C2.
Reference Variable	Static type	Dynamic type	Dynamic type assigned by																																	
OREF1	C1	C1	CREATE object oref1.																																	
OREF11	C1	C2	CREATE OBJECT oref11 TYPE C2.																																	
OREF111	C1	C2	CREATE OBJECT OREF111. OREF111 = OREF2.																																	
OREF2	C2	C2	CREATE OBJECT oref2.																																	
Method call	Calls method	Reason																																		
OREF1->M1	M1 of class C1.	Both Static & Dynamic type of OREF1 refers to C1.																																		
OREF11->M1	M1 of class C2.	Static type of OREF11 refers to C1, dynamic type to C2.																																		
OREF111->M1	M1 of class C2.	Static type of OREF111 refers to C1, dynamic type to C2.																																		
OREF2->M1	M1 of class C2	Both Static & Dynamic type of OREF2 refers to C2.																																		
Dump	<pre>REPORT YSUB00PS18. class c1 definition. public section. methods : m1. endclass. class c1 implementation. method m1 . write:/5 ' I am m1 of c1'. endmethod. endclass. class c2 definition inheriting from c1. public section. methods : m1 redefinition. endclass. class c2 implementation. method m1. write:/5 ' I am m1 of c2'. endmethod. endclass. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1, OREF11 TYPE REF TO C1, OREF111 TYPE REF TO C1, OREF2 TYPE REF TO C2 . CREATE OBJECT : OREF1 , OREF11 TYPE C2, OREF111 OREF2 . OREF111 = OREF2. CALL METHOD : OREF1->M1 , " Output : I am m1 of c1 OREF11->M1 , " Output : I am m1 of c2</pre>																																			

	OREF111->M1, " Output : I am m1 of c2 OREF2->M1 . " Output : I am m1 of c2
Output	I am m1 of c1 I am m1 of c2 I am m1 of c2 I am m1 of c2

4.12 Static type should be more general than dynamic type of a reference variable	
Theme	Static type of a reference variable can refer to a superclass, whereas its dynamic type can refer to a subclass of the superclass. In that case, the reference variable will identify all the common components of the superclass and subclass. It will not be able to identify any new components in the subclass, which are not present in its superclass.
Program Descr.	<p>This program contains class C1 and its subclass C2.</p> <p>Class C1 contains method M1, which is also redefined in class C2.</p> <p>Class C2 contains a new method M2.</p> <p>A reference variable OREF11 is created with static type of C1 and dynamic type of C2.</p> <p>Method M2 is attempted to be called using OREF11.</p> <p>This produces compilation error, establishing the theme.</p>
Dump	<pre> REPORT YSUB00PS18. class c1 definition. public section. methods : m1. endclass. class c1 implementation. method m1 . write:/5 ' I am m1 of c1' . endmethod. endclass. class c2 definition inheriting from c1. public section. methods : m1 redefinition. methods : m2 . endclass. class c2 implementation. method m1. write:/5 ' I am m1 of c2' . endmethod. method m2. write:/5 ' I am m2' . endmethod. endclass. START-OF-SELECTION. DATA : OREF11 TYPE REF TO C1. CREATE OBJECT : OREF11 TYPE C2. CALL METHOD : OREF11->M2 . </pre>
Output	Compilation error fails to identify method M2 in the last statement of the program.

4.13 Method of a parent class, used from its subclass, uses attributes of the parent class only, if the method is not re-defined in subclass.	
Theme	As long as a method(using private attributes) inherited from a superclass is not redefined, it still uses the private attributes of the superclass, not those of the subclass, even if the subclass has private attributes of the same name.
Program Descrip.	<p>Class C1 contains a method M1 in the public section and a private variable, NUM of value = 5. Method M1 in class C1 displays the value of private variable, NUM. Class C2 is a subclass of class C1. It does not redefine method M1. But, it has also a private variable , NUM with value = 6.</p> <p>An object is created from class C2 and the method M1 is called.</p> <p>The output shows that the variable NUM(as displayed by method M1) has been taken from class C1, not C2. This establishes the theme.</p>
Dump	<pre> report ysubdel . CLASS c1 DEFINITION. PUBLIC SECTION . METHODS : m1 . PRIVATE SECTION. DATA : num TYPE I VALUE 5 . ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD : m1 . write:/5 num . ENDMETHOD. ENDCLASS. CLASS c2 DEFINITION INHERITING FROM c1. PUBLIC SECTION . DATA : num TYPE I VALUE 6. ENDCLASS. CLASS c2 IMPLEMENTATION. ENDCLASS. START-OF-SELECTION. DATA : oref2 TYPE REF TO c2 . CREATE OBJECT : oref2 . CALL METHOD oref2->m1 . </pre>
Output	5

4.14 Demo on Widening Cast	
Theme	This program will show the use of widening cast operator
Program Description	Class C1 is superclass of C2 . Object of class C2 is assigned the object of class C1 using widening cast operator. This helps to avoid the compilation error, but generates the error 'cx_sy_move_cast_error trapped' at runtime, which is handled and reported properly in the program.
Dump	<pre> REPORT YSUBDEL . class c1 definition. public section. data : num type i value 5. endclass. class c1 implementation. endclass. class c2 definition inheriting from c1. public section. endclass. class c2 implementation. endclass. start-of-selection . data : obj1 type ref to c1 , obj2 type ref to c2 . create object : obj1 , obj2 . TRY. obj2 ?= obj1. CATCH cx_sy_move_cast_error. write:/5 'cx_sy_move_cast_error trapped'. ENDTRY. </pre>
Output	cx_sy_move_cast_error trapped

5 Interface

5.1 Simple use of an interface	
Theme	This program will show simple use of an interface with its own data and methods and how it is implemented in a class. It will also show that there can be methods of same name for an interface and the class implementing the interface.
Program Desc	<p>This program contains an interface I1 with attribute : NUM an method : METH1. This interface is implemented in a class : C1 which also has its own method METH1.</p> <p>An object OREF is created from class C1 and both the methods METH1 , one for class and another for interface is called using the object.</p>
Dump	<pre> report ysubdel . interface i1. data : num type i . methods : meth1. endinterface. class c1 definition. public section. methods : meth1. " class C1's own method interfaces : i1. endclass. class c1 implementation. method : meth1. write:/5 'I am meth1 in c1'. endmethod. method i1~meth1. write:/5 'I am meth1 from i1'. endmethod. endclass. start-of-selection. data : oref type ref to c1. create object oref. write:/5 oref->i1~num. call method oref->meth1. call method oref->i1~meth1. </pre>
Output	<pre> 0 I am meth1 in c1 I am meth1 from i1 </pre>

5.2 Interfaces can only be implemented in the public section of a class	
Theme	This program will show you that classes implementing an interface can only contain the features of the interface in its public section.
Program Description	In this program, class C1 is trying to accommodate interface I1 in its PRIVATE SECTION . This creates a compilation error, establishing the theme.
Dump	<pre> report ysubdel . interface i1. methods : meth1. endinterface. class c1 definition. protected section. interfaces : i1. endclass. </pre>
Output	Compilation error with error message :- INTERFACES may only be implemented in the public section.

5.3 A class with an interface should implement all the methods of that interface	
Theme	This program will show that a class containing an interface should implement all the methods of the interface in its implementation section.
Program Descrip	Class C1 implements interface I1 , which has got two methods , METH1 and METH2 . But, in the IMPLEMENTATION section of class C1 , only METH1 is implemented. This program will create a compilation error, establishing the theme .
Dump	<pre> report ysubdel . interface i1. methods : meth1 , meth2 . endinterface. class c1 definition. public section. interfaces : i1. endclass. class c1 implementation. method i1~meth1. write:/5 'I am meth1 from i1'. endmethod. endclass. start-of-selection. data : oref type ref to c1. create object oref. call method oref->i1~meth1. </pre>
Output	Compilation error with error message :- Implementation missing for method "I1~METH2"

5.4 Values for interface attributes are assigned at the time of inclusion in a class		
Theme	<p>One cannot specify values for attributes while declaring them in an interface like the following fashion : DATA : <var> TYPE <type> VALUE <val>.</p> <p>Instead of doing that, one has to specify the values for different attributes of interface at the point where the interface is declared in the public section of a class.</p>	
Program Descr.	<p>Interface I1 contains two numeric attributes , NUM1 and NUM2 .</p> <p>In version 1 of the program, attempt is made to specify the values while defining those attributes in the interface. This version does not get successfully compiled and thus establishes the theme.</p> <p>In version 2, values for interface attributes are specified at the time when I1 is included in the public section of class C1. This version gets successfully compiled and produces a result.</p>	
	<pre> report ysubdel . interface i1 . data : num1 type i value 5 , num2 type i value 6 . endinterface. class c1 definition. public section. interfaces : i1 . methods m1. endclass. class c1 implementation. method m1. write:/5 i1~num1, i1~num2. endmethod. endclass. start-of-selection. data : oref type ref to c1. create object oref. call method oref->m1. </pre>	<pre> report ysubdel . interface i1 . data : num1 type i , num2 type i . endinterface. class c1 definition. public section. interfaces : i1 DATA VALUES num1 = 5 num2 = 6 . methods m1. endclass. class c1 implementation. method m1. write:/5 i1~num1, i1~num2. endmethod. endclass. start-of-selection. data : oref type ref to c1. create object oref. call method oref->m1. </pre>
	Version 1	Version 2
Output	<p><u>Output of version 1:-</u> Compilation error with error message:- Within an interface, you cannot use VALUE with attributes(except constants)</p> <p><u>Output of version 2:-</u> 5 6</p>	

5.5 Use of FINAL methods from Interface	
Theme	This program will demonstrate how to create final method in a class from one of the methods of an interface.
Program Description	<p>Interface I1 contains two methods : M1 and M2.</p> <p>I1 is included and incorporated in class : C1 with M2 as a final method. Both the methods are implemented in class C1.</p> <p>Class C2 is a subclass of class C1. It redefines method : I1~M1 and re-implements it, but it does not do that for I1~M2 as that is declared as final method.</p> <p>In the START-OF-SELECTION block, object OREF1 is created from class C1 and OREF2 from class C2 and both the methods M1 and M2 are called using both the objects.</p>
Dump	<pre> report ysubdel . interface i1 . methods : m1 , m2 . endi nterface. class c1 defini ti on. public section. interfaces : I1 final methods m2 . endcl ass. class c1 implemen tation. method i1~m1. write:/5 'I am m1 in c1'. endmethod. method i1~m2. write:/5 'I am m2 in c1'. endmethod. endcl ass. class c2 defini ti on inheriting from c1. public section. methods : i1~m1 redefini ti on . endcl ass. class c2 implemen tation. method : i1~m1. write:/5 'I am m1 in c2'. endmethod. endcl ass. start-of-sele cti on. data : oref1 type ref to c1, oref2 type ref to c2 . create object : oref1 , oref2. call method : oref1->i1~m1 , " Output : I am m1 in c1 oref2->i1~m1 , " Output : I am m1 in c2 oref1->i1~m2 , " Output : I am m2 in c1 oref2->i1~m2 . " Output : I am m2 in c1 </pre>
Output	<pre> I am m1 in c1 I am m1 in c2 I am m2 in c1 I am m2 in c1 </pre>

5.6 Use of Abstract methods from Interface	
Theme	<p>This program will demonstrate the way by which a method from an interface can be used as an abstract method for a class.</p> <p>Abstract methods can only be implemented by the subclasses .</p> <p>A class containing an abstract method should be abstract itself.</p>
Program Desc.	<ul style="list-style-type: none"> ❖ This program contains an interface I1 with two methods , M1 and M2. ❖ Class C1 includes and implements methods of interface I1 , declaring method M2 as an abstract method. Hence, class C1 was also declared as an abstract class. ❖ Class C1 implements method I1~M1 , but not I1~M2 as this is an abstract method. ❖ Class C2 is a subclass of C1 , which defines I1~M2. ❖ In the START-OF-SELECTION block,object OREF2 is created from class C2(class C1 cannot be instantiated, as this is an abstract class) and both the methods : I1~M1 and I1~M2 are called.
Dump	<pre> report ysubdel . interface i1 . methods : m1 , m2 . endinterface. class c1 definition abstract. public section. interfaces : i1 abstract methods m2 . endclass. class c1 implementation. method i1~m1. write:/5 'I am m1 in c1'. endmethod. endclass. class c2 definition inheriting from c1. public section. methods : i1~m2 redefinition. endclass. class c2 implementation. method : i1~m2. write:/5 'I am m2 in c2'. endmethod. endclass. start-of-selection. data : oref2 type ref to c2 . create object : oref2. call method : oref2->i1~m1 , oref2->i1~m2. </pre>
Output	<pre> I am m1 in c1 I am m2 in c2 </pre>

5.7 Use of Interface Reference Variable

Theme	This program will show the use of interface reference variable and how it can be used to access the components of an interface in a class(implementing that interface). Use of interface reference variable paves the way for polymorphism via interface.												
Program Descrip.	<p>Interface I1 , included/ implemented in class C1 has the following components:-</p> <table border="1"> <thead> <tr> <th>component</th><th>nature</th></tr> </thead> <tbody> <tr> <td>C_name</td><td>Constant with value =ABAP</td></tr> <tr> <td>inum</td><td>Instance attribute with value = 5</td></tr> <tr> <td>cnum</td><td>Static attribute of value 6</td></tr> <tr> <td>M1</td><td>Instance method</td></tr> <tr> <td>M2</td><td>Static method</td></tr> </tbody> </table> <p>Class C1 implements all the methods in its IMPLEMENTATION section. In the START-OF-SELECTION block, all the different attributes and methods are called using class reference variable OREF first. Then, interface reference variable IREF is used to do similar jobs, after the assignment IREF = OREF.</p>	component	nature	C_name	Constant with value =ABAP	inum	Instance attribute with value = 5	cnum	Static attribute of value 6	M1	Instance method	M2	Static method
component	nature												
C_name	Constant with value =ABAP												
inum	Instance attribute with value = 5												
cnum	Static attribute of value 6												
M1	Instance method												
M2	Static method												
Dump	<pre> report ysubdel . interface i1 . constants : c_name(4) type c value ' ABAP' . data : inum type i . class-data : cnum type i . methods : m1 . class-methods : m2. endi nterface. class c1 definition . public section. interfaces : I1 data values inum = 5 cnum = 6 . endcl ass. class c1 implementation. method i1~m1. write:/5 'I am m1 in c1'. endmethod. method i1~m2. write:/5 'I am class method m2 in c1'. endmethod. endcl ass. start-of-selection. data : iref type ref to i1 , oref type ref to c1 . create object : oref. write:/5 oref->i1~inum , oref->i1~cnum , c1=>i1~cnum . call method : oref->i1~m1 , oref->i1~m2 , c1=>i1~m2 . write:/5 sy-uline . iref = oref . write:/5 iref->inum , iref->cnum , i1=>c_name . call method : iref->m1 , </pre>												

	i ref->m2 .
Output	<div>5 6 6</div> <div>I am m1 in c1</div> <div>I am class method m2 in c1</div> <div>I am class method m2 in c1</div>
	<div>5 6 ABAP</div> <div>I am m1 in c1</div> <div>I am class method m2 in c1</div>

5.8 Use of Nested Interface	
Theme	This program will demonstrate how an interface can be included in another interface And the final interface containing all the interfaces can be used inside a class.
Program Descr.	<ul style="list-style-type: none"> ❖ Interface I1 contains method M1. ❖ Interface I2 includes interface I1. However, it also contains two methods M1 and M2 of its own. ❖ Class C1 contains interface I2 and implements methods : I1~M1 , I2~M1 and I2~M2. ❖ Object is created from class C1 and all the methods are called.
	<pre> report ysubdel . interface i1 . methods m1. endi nterface. interface i2. methods : m1 , m2. interfaces i1. endi nterface. class c1 defi nition. public section. interfaces : i2. endcl ass. class c1 implementation. method : i1~m1. write:/5 'I am m1 from i1'. endmethod. method : i2~m1. write:/5 'I am m1 from i2'. endmethod. method : i2~m2. write:/5 'I am m2 from i2'. endmethod. endcl ass. START-OF-SELECTION. data : oref type ref to c1. create object oref. call method : oref->i1~m1 , " Output : I am m1 from i1 oref->i2~m1 , " Output : I am m1 from i2 oref->i2~m2 . " Output : I am m1 from i2 </pre>
Output	<pre> I am m1 from i1 I am m1 from i2 I am m2 from i2 </pre>

5.9 Using ALIASES

Theme	<p>The full name of a component which an Interface adds to a class or another interface is intf~comp. Alias names can be substituted for this name when</p> <ul style="list-style-type: none"> -defining compound interfaces, or -declaring Interfaces in a class
Program Descr.	<p>Interface I1 contains method M1. Interface I2 contains method : M1 and M2 and interface I1. It aliases method M1 of I1 as METH1. Class C1 contains interface I2 and aliases method M2 of I2 as METH2. All the methods :- I1~M1 , I2~M1 and I2~M2 are implemented in class C1. In the START-OF-SELECTION block, object OREF is created from class C1 and the alias names are used to call the methods.</p>
Dump	<pre> report ysubdel . interface i1 . methods m1. endinterface. interface i2. methods : m1 , m2 . interfaces i1. aliases meth1 for i1~m1. endinterface. class c1 definition. public section. interfaces : i2. aliases meth2 for i2~m2. endclass. class c1 implementation. method i1~m1. write:/5 'I am m1 from i1'. endmethod. method : i2~m1. write:/5 'I am m1 from i2'. endmethod. method : i2~m2. write:/5 'I am m2 from i2'. endmethod. endclass. START-OF-SELECTION. data : oref type ref to c1. create object oref. call method : oref->i2~meth1. call method : oref->meth2 . </pre>
Output	<p>I am m1 from i1 I am m2 from i2</p>

5.10 Polymorphism via Interfaces

Theme	<p>This program will demonstrate how a method of an interface can be interpreted differently in different classes. At runtime, one can use different class reference variables to call same method of the interface to have different things done.</p> <p>Use of interface reference variable in this context further simplifies the work.</p>
Program Descrip.	<ul style="list-style-type: none"> ❖ Both the classes C1 and C2 , totally different from each other, contains interface I1. ❖ Interface I1 contains method M1 , which is implemented differently in C1 and C2. ❖ In the START-OF-SELECTION block, objects are created from class C1 and C2. An Interface reference variable IREF is also defined. ❖ By assigning the class reference variables(of C1 and C2 , one at a time) to interface reference variable IREF and then calling method M1 (using IREF) clearly demonstrates how polymorphism can be achieved using the concept of interface.
Dump	<pre> REPORT YSUBDEL. interface I1. METHODS : M1 . ENDINTERFACE. CLASS C1 DEFINITION. PUBLIC SECTION. INTERFACES : I1. ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD I1~M1. WRITE:/5 'I am method m1 in c1'. ENDMETHOD. ENDCLASS. CLASS C2 DEFINITION. PUBLIC SECTION. INTERFACES : I1. ENDCLASS. CLASS C2 IMPLEMENTATION. METHOD I1~M1. WRITE:/5 'I am method m1 in c2'. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1 , OREF2 TYPE REF TO C2 , IREF TYPE REF TO I1 . CREATE OBJECT : OREF1 , OREF2 . IREF = OREF1. CALL METHOD IREF->M1. IREF = OREF2. CALL METHOD IREF->M1. </pre>
Output	<pre> I am method m1 in c1 I am method m1 in c2 </pre>

6 Friendship

6.1 Friendship between Classes	
Theme	<p>A class can grant friendship to another class. By granting friendship , it allows another class to:-</p> <ul style="list-style-type: none"> ❖ Use its private components. ❖ Instantiate it, irrespective of the CREATE PRIVATE addition.
Program Descr.	<ul style="list-style-type: none"> ➤ Class C2 is created using CREATE PRIVATE option. That means, only the class itself and its friends can instantiate this class. ➤ Class C2 has a private method M2 and a private attribute , NUM. This means that these components can be accessed by class C2 itself and its friends. ➤ Now, C2 has granted friendship to class C1. ➤ So, methods of class C1 can access private components of C2 as well as can instantiate class C2. <p>This establishes the theme.</p>
Dump	<pre>REPORT YSUBDEL. CLASS C1 DEFINITION DEFERRED. CLASS C2 DEFINITION CREATE PRIVATE FRIENDS C1 . PROTECTED SECTION. DATA : NUM TYPE I VALUE 5. METHODS : M2. ENDClass. CLASS C2 IMPLEMENTATION. METHOD M2. WRITE:/5 'I am method m2 in C2' . ENDMETHOD. ENDCLASS . class c1 definition. public section . methods : m1. endclass. class c1 implementation. method m1. DATA : OREF2 TYPE REF TO C2. CREATE OBJECT OREF2. WRITE:/5 OREF2->NUM. CALL METHOD OREF2->M2. ENDMETHOD. endclass. START-OF-SELECTION. DATA : OREF1 TYPE REF TO C1. CREATE OBJECT OREF1. CALL METHOD OREF1->M1.</pre>
Output	<p>5</p> <p>I am method m2 in C2</p>

6.2 Subclasses of friends can also become friends.	
Theme	Subclasses of the friend class are also friends of the class granting friendship(to their super classes)
Program Descrip.	<ul style="list-style-type: none"> ❖ Class C2 has granted friendship to class C1.Hence, C1 is friend of class C2. ❖ Class C11 is a subclass of class C1. ❖ So, class C11 is also a friend of class C2.Class C11 can thus access the protected components of class C2. <p>This establishes the theme.</p>
	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION DEFERRED. CLASS C2 DEFINITION FRIENDS C1 . PROTECTED SECTION. DATA : NUM TYPE I VALUE 5. ENDCLASS. CLASS C2 IMPLEMENTATION. ENDCLASS . class c1 definition. public section . methods : m1. endclass. class c1 implementation. method m1. DATA : OREF2 TYPE REF TO C2. CREATE OBJECT OREF2. WRITE: /5 OREF2->NUM. ENDMETHOD. endclass. class c11 definition inheriting from c1. public section. methods : m11. endclass. class c11 implementation. method m11. DATA : OREF2 TYPE REF TO C2. CREATE OBJECT OREF2. WRITE: /5 OREF2->NUM. endmethod. endclass. START-OF-SELECTION. DATA : OREF11 TYPE REF TO C11. CREATE OBJECT OREF11. CALL METHOD OREF11->M11. </pre>
Output	5

6.3 Friendship is one sided		
Theme	In principle, granting of friendship is one-sided: A class granting a friendship is not automatically a friend of its friends. If the class granting the friendship wants to access the private components of a friend, then the latter has to explicitly grant friendship to the former.	
Program Descr.	<p>Class C2 grants friendship to class C1. Hence, class C1 can access protected attribute(num2) of C2.</p> <p>But, class C2 cannot access protected attribute(num1) of class C1. This is because friendship is one-sided.</p> <p>To allow C2 access protected attribute of C1, class C1 must also declare C2 as its friend.</p>	
	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION DEFERRED. CLASS C2 DEFINITION FRIENDS C1 . PROTECTED SECTION. DATA : NUM2 TYPE I VALUE 15. METHODS : M2. ENDCLASS. class c1 definition . public section. methods : methpub. private section . data : num1 type i value 10 . methods : m1. endclass. CLASS C2 IMPLEMENTATION. METHOD M2. data : oref1 type ref to c1. create object oref1. write: /5 oref1->num1. ENDMETHOD. ENDCLASS . class c1 implementation. method m1. DATA : OREF2 TYPE REF TO C2. CREATE OBJECT OREF2. WRITE: /5 OREF2->NUM2. ENDMETHOD. method methpub. call method m1. endmethod. endclass. START-OF-SELECTION. DATA : OREF TYPE REF TO C1. CREATE OBJECT OREF . CALL METHOD OREF ->METHPUB. </pre>	<pre> REPORT YSUBDEL. CLASS C1 DEFINITION DEFERRED. CLASS C2 DEFINITION FRIENDS C1 . PROTECTED SECTION. DATA : NUM2 TYPE I VALUE 15. METHODS : M2. ENDCLASS. class c1 definition friends c2. public section. methods : methpub. private section . data : num1 type i value 10 . methods : m1. endclass. CLASS C2 IMPLEMENTATION. METHOD M2. data : oref1 type ref to c1. create object oref1. write: /5 oref1->num1. ENDMETHOD. ENDCLASS . class c1 implementation. method m1. DATA : OREF2 TYPE REF TO C2. CREATE OBJECT OREF2. WRITE: /5 OREF2->NUM2. ENDMETHOD. method methpub. call method m1. endmethod. endclass. START-OF-SELECTION. DATA : OREF TYPE REF TO C1. CREATE OBJECT OREF . CALL METHOD OREF ->METHPUB. </pre>
	Incorrect	Correct

7 Events

7.1 Events with Handler Method in the same class	
Theme	<p>Event is a mechanism by which method of one class can raise method of another class, without the hazard of instantiating that class .</p> <p>The steps to be followed are as follows:-</p> <ul style="list-style-type: none"> ❖ Create an event in a class ❖ Create a triggering method in the same class which will raise the event. ❖ Create an event handler method for the event in same/other class. ❖ Register the event handler method in the program. <p>Now, your settings are complete. Create an object from the class containing the event and call the triggering method to raise the event.</p>
Program Descr.	<p>Class C1 contains an event E1, for which the triggering method is T1.</p> <p>Event handler method for event E1 is M1, placed in the same class C1.</p> <p>Registration is done at runtime for M1.</p> <p>Object is created from class C1 and the triggering method T1 is called, which raises the event and ultimately calls event handler method M1.</p>
Dump	<pre>REPORT YSUB00PS7 CLASS c1 DEFINITION. PUBLIC SECTION. *(1)Creating event : E1 EVENTS: E1. *(2) Creating an event handling method. This method can belong to * same or different class METHODS: M1 FOR EVENT E1 OF c1. * Method to raise the event METHODS : T1. ENDCLASS. CLASS c1 IMPLEMENTATION. * Method : M1 will be called when the event is raised METHOD : M1. write:/5 ' I am the event handler method' . ENDMETHOD. * Method : T1 will raise the event METHOD : T1. write:/5 'I am T1, going to raise event E1' . raise event E1. ENDMETHOD. ENDCLASS. Start-of-selection. Data: oref type ref to c1. Create object: oref . * Registering the event handler method SET HANDLER oref->M1 FOR oref . * Calling the event which will raise the event. call method oref->T1.</pre>
Output	<p>I am T1, going to raise event E1</p> <p>I am the event handler method</p>

7.2 Event with event handler method in different class	
Theme	Similar to above. Here, event handler method is in different class
	<p>Class C1 contains an event E1, for which the triggering method is T1. Event handler method for event E1 is M1, placed in the another class, C2. Registration is done at runtime for M1. Object is created from class C1 and the triggering method T1 is called, which raises the event and ultimately calls event handler method M1 in class C2.</p>
Dump	<pre> REPORT YSUB00PS7 CLASS c1 DEFINITION. PUBLIC SECTION. * Creating event : E1 EVENTS: E1. * Triggering method : T1 METHODS : T1. ENDCLASS. CLASS C2 DEFINITION. PUBLIC SECTION. * Creating an event handling method. METHODS: M1 FOR EVENT E1 OF c1. endclass. CLASS c1 IMPLEMENTATION. * Method : T1 will raise the event METHOD : T1. write:/5 'I am T1, going to raise event E1'. raise event E1. ENDMETHOD. ENDCLASS. class c2 implementation. * Method : M1 will be called when the event is raised METHOD : M1. write:/5 ' I am the event handler method in c2'. ENDMETHOD. endclass. Start-of-selection. Data: oref1 type ref to c1, oref2 type ref to c2. Create object: oref1 , oref2 . * Registering the event handler method SET HANDLER oref2->M1 FOR oref1 . * Calling the event which will raise the event. call method oref1->T1. </pre>
Output	<p>I am T1, going to raise event E1 I am the event handler method in c2</p>

7.3 More than one event handler method can exist for same event	
Theme	For an event in a class, there can be more than one event handler methods in same or different class. However, at runtime only one event handler method will be triggered at a time, based on the registration.
Program Descr.	<ul style="list-style-type: none"> ❖ Class C1 contains an event E1, for which the triggering method is T1 and the event handler methods are :- <ul style="list-style-type: none"> ➤ M1 in same class C1. ➤ M2 in another class C2. ❖ In the START-OF-SELECTION block, objects are created from class C1 and C2. ❖ First, registration is made using method M1 of class C1 as event handler method. ❖ Then, the event E1 is raised, calling method T1. This raises event handler method M1 of class C1. ❖ After that, the earlier registration is de-activated and new registration is made for method M2 of class C2 as event handler method. ❖ Event E1 is raised calling method T1. This raises event handler method M2 of class C2.
Dump	<pre> REPORT YSUB00PS7 CLASS c1 DEFINITION. PUBLIC SECTION. * Creating event : E1 EVENTS: E1. * Creating an event handling method. METHODS: M1 FOR EVENT E1 OF c1. * Method to raise the event METHODS : T1. ENDCLASS. CLASS C2 DEFINITION. PUBLIC SECTION. * Creating an event handling method. METHODS: M2 FOR EVENT E1 OF c1. endcl ass. CLASS c1 IMPLEMENTATION. * Method : T1 will raise the event METHOD : T1. write:/5 'I am T1, going to raise event E1'. raise event E1. ENDMETHOD. * Method : M1 will be called when the event is raised METHOD : M1. write:/5 ' I am the event handler method M1 in c1'. ENDMETHOD. ENDCLASS. class c2 implementation. * Method : M2 will be called when the event is raised METHOD : M2. write:/5 ' I am the event handler method M2 in c2'. ENDMETHOD. endcl ass. Start-of-selection. Data: oref1 type ref to c1,</pre>

	<pre>oref2 type ref to c2. Create object: oref1 , oref2 . * Registering the event handler method SET HANDLER oref1->M1 FOR oref1 . * Calling the event which will raise the event. call method oref1->T1. * De-Registering the earlier event handler method SET HANDLER oref1->M1 FOR oref1 ACTIVATION space . * Registering the new event handler method SET HANDLER oref2->M2 FOR oref1 . * Calling the event which will raise the event. call method oref1->T1.</pre>
Output	<pre>I am T1, going to raise event E1 I am the event handler method M1 in c1 I am T1, going to raise event E1 I am the event handler method M2 in c2</pre>

7.4 Use of static event	
Theme	Static methods can only raise static events. The FOR...addition is not required to register for static events.
Program	Class C1 contains a static event E1 and static triggering method T1 . The event handler method M1 is in class C1 itself. At the time of registering event M1 as event handler method, the FOR ...addition is omitted.
Dump	<pre> REPORT YSUB00PS7 CLASS c1 DEFINITION. PUBLIC SECTION. * Creating event : E1 CLASS-EVENTS: E1. * Creating an event handling method. METHODS: M1 FOR EVENT E1 OF c1. * Method to raise the event CLASS-METHODS : T1. ENDCLASS. CLASS c1 IMPLEMENTATION. * Method : T1 will raise the event METHOD : T1. write:/5 'I am T1, going to raise event E1'. raise event E1. ENDMETHOD. * Method : M1 will be called when the event is raised METHOD : M1. write:/5 'I am the event handler method M1 in c1'. ENDMETHOD. ENDCLASS. Start-of-selection. Data: oref1 type ref to c1. Create object: oref1 . * Registering the event handler method SET HANDLER oref1->M1 . * Calling the event which will raise the event. call method oref1->T1. </pre>
Output	<pre> I am T1, going to raise event E1 I am the event handler method M1 in c1 </pre>

7.5 Events with export parameters	
Theme	Events can have export parameters, which it passes to its event handler method. The triggering method must pass values for all the exporting parameters of the event while raising the event using RAISE EVENT statement. The interface of an event handler method consists of a list of IMPORTING parameters, whose names are identical with those in the EXPORTING list and which are automatically created from the interface of the event. Each handler method can however specify which event parameters it wants to handle and which it does not.
Program Descr.	Class C1 contains event E1 which exports two parameters , NUM1 and NUM2 to its event handler method , M1 in class C1 . Method T1 is the triggering method for the event, which passes values to the EXPORTING parameters of the event at the time of RAISE EVENT statement.
Dump	<pre> REPORT YSUBDEL1. CLASS c1 DEFINITION. PUBLIC SECTION. EVENTS : E1 EXPORTING value(NUM1) TYPE I value(NUM2) TYPE I . METHODS : M1 FOR EVENT E1 OF C1 IMPORTING NUM1 NUM2 METHODS : T1. ENDCLASS. CLASS C1 IMPLEMENTATION. METHOD : M1. WRITE:/5 'First input ' , num1 . write:/5 'Second input ' , num2 . ENDMETHOD. METHOD T1. RAISE EVENT E1 exporting num1 = 2 num2 = 3. ENDMETHOD. ENDCLASS. START-OF-SELECTION. DATA : oref TYPE REF TO c1. CREATE OBJECT oref. SET HANDLER oref->M1 for oref. call method oref->T1. </pre>
Output	<pre> First input 2 Second input 3 </pre>

8 Class-Based Exceptions

8.1 Using SAP provided exception class

Theme	Errors in a program, which are detected at runtime and can be trapped, can be dealt with using SAP provided standard exception-classes.	
Program Descr.	This program makes a runtime error where a division by zero is observed. Let us take three different versions of the program and see the outputs	
Ver.No	Program	Output
1	<pre>REPORT YSUBCLASS_EXCEPTION. DATA: i TYPE i VALUE 1. START-OF-SELECTION. i = i / 0.</pre>	<p>Short Dump as follows:- Runtime errors COMPUTE_INT_ZERODIVIDE Exception CX_SY_ZERODIVIDE Occurred on 12.04.2004 at 17:02:18 Divide by 0 (type 1).</p>
2	<pre>REPORT YSUBCLASS_EXCEPTION. DATA: i TYPE i VALUE 1. START-OF-SELECTION. catch system-exceptions COMPUTE_INT_ZERODIVIDE = 2. i = i / 0. endcatch. if sy-subrc = 2. write:/5 'Division by zero!!!Check'. endif.</pre>	'Division by zero!!!Check'
3	<pre>REPORT YSUBCLASS_EXCEPTION. DATA: i TYPE i VALUE 1. START-OF-SELECTION. TRY. i = i / 0. CATCH cx_sy_zerodivide. write:/5 'Divide by zero caught'. ENDTRY.</pre>	Divide by zero caught

The three versions basically represent the same program, but shows how an error can be trapped using SAP provided standard exception class.

8.2 When both superclass and subclass are used to track error

Theme	SAP provided standard exception classes can reside in different levels of hierarchy tree; CX_ROOT being at the top. So, if both superclass and its subclass are used in a program to detect errors in TRY...ENDTRY block, the subclass should be used first, then the superclass.		
Program Descr.	<p>This program creates a division by zero problem. Here, both the superclass CX_ROOT and subclass CX_SY_ZERODIVIDE is used to trap errors.</p> <p>In version 1, superclass is used first to trap the error- which creates a compilation error.</p> <p>In version 2, subclass is used first - which gets compiled and executed successfully.</p>		
Dump	Version	Code	Output
	1	<pre>REPORT YSUBCLASS_EXCEPTION. DATA: i TYPE i VALUE 1. START-OF-SELECTION. TRY. i = i / 0. CATCH cx_root. write:/5 'Error trapped'. CATCH cx_sy_zerodivide. write:/5 'Div. by zero!'. ENDTRY.</pre>	Compilation error :- Exception in the CATCH clauses are not sorted in ascending order
	2	<pre>REPORT YSUBCLASS_EXCEPTION. DATA: i TYPE i VALUE 1. START-OF-SELECTION. TRY. i = i / 0. CATCH cx_sy_zerodivide. write:/5 'Div. by zero!'. CATCH cx_root. write:/5 'Error trapped'. ENDTRY.</pre>	Div. by zero!

8.3 Propagation of Class-Based exceptions in procedures to the caller	
Theme	Class-based exceptions in procedures can be propagated to the caller in the definition of the interface using the RAISING addition, if the exception is not to be handled in the procedure.
Program Descr.	In the following program, the piece of code which may contain error is inside a subroutine , SUB_CHECK_NO. But, the error is not handled in the procedure itself. So, a RAISING statement (with probable class-based exception that might be raised by the procedure) is specified at the definition of the procedure itself, and is later trapped using CATCH statement.
	<pre> REPORT YSUB00PS17 start-of-selection. try. perform sub_check_no using 5 . catch cx_sy_zerodivide. write: /5 'Hello' . endtry. FORM sub_check_no USING P_P_NO RAISING CX_SY_ZERODIVIDE. p_p_no = p_p_no / 0 . ENDFORM. " sub_check_no </pre>
Output	Hello

8.4 Program can raise exceptions based on SAP standard exception-classes	
Theme	The runtime environment only causes exceptions that are based on pre-defined classes, while in ABAP programs one can use raise pre-defined as well as user-specific exception classes.
Program Descr.	This program will show how exceptions based on SAP provided exception classes can be manually raised. Here, exception based on SAP exception-class CX_SY_ZERODIVIDE is raised manually.
Dump	<pre>REPORT YSUBOOPS17 . data : i num type i . try. raise exception type cx_sy_zerodivide. CATCH cx_sy_zerodivide. write: /5 'Exception caught'. endtry.</pre>
Output	Exception caught

8.5 Objects are created from exception classes when error is trapped	
Theme	<p>When a class-based exception is trapped using TRY...CATCH...ENDTRY statement, objects are created from the exception class. One can create the object using CATCH <exception name> INTO <exception class reference variable> statement.</p> <p>CX_ROOT is at the top of the inheritance tree for all SAP provided exception class and have some pre-defined methods available, which are adopted by all exception-classes.</p>
Program Descr.	<p>The program involves a division by zero error in the guarded section, which raises an exception on exception-class : CX_SY_ZERODIVIDE.</p> <p>A reference variable , EREF with static type referring to the exception class CX_SY_ZERODIVIDE is used to create an object while using the CATCH statement.</p> <p>Once the object is created, it can be used to manipulate some of the methods and attributes of the class CX_SY_ZERODIVIDE, which has been inherited by this class from CX_ROOT.</p>
Dump	<pre>REPORT YSUB00PS17 . data : inum type i value 5 , descrip type string , progame like sy-repid , lineno type i . data : eref type ref to cx_sy_zerodivide. try. inum = inum / 0. CATCH cx_sy_zerodivide into eref. * Utilizing methods/attributes using object of the exception classes call method eref->get_text receiving result = descrip. write: /5 'Name of the error trapped : ' , descrip. call method eref->get_source_position importing program_name = progame source_line = lineno . write: /5 'Error detected in program' , progame(15) , 'line number' , lineno. write: /5 eref->kernel_errid. endtry.</pre>
Output	<p>Name of the error trapped : Division by zero</p> <p>Error detected in program YSUBDEL line number 10</p> <p>COMPUTE_INT_ZERODIVIDE</p>

8.6 Demo on Locally Defined Exception-Class	
Theme	One can create his own exception-class locally in a program and raise exceptions related to his own class. This program will show how to do that.
Program Descrip.	In this program, an exception class CX_MY_EXCEPTION is defined locally, inheriting from standard exception class CX_STATIC_CHECK . Method M1 of class C1 raises it in the START-OF-SELECTION block, in the guarded section(between TRY.. and CHECK), which is trapped and dealt with.
Dump	<pre> REPORT YSUBCLASS_EXCEPTION_3. CLASS cx_my_exception DEFINITION INHERITING FROM CX_STATIC_CHECK. ENDCLASS. CLASS cx_my_exception IMPLEMENTATION. ENDCLASS. CLASS C1 DEFINITION. PUBLIC SECTION. METHODS: m1 raising cx_my_exception . ENDCLASS. CLASS c1 IMPLEMENTATION. METHOD m1. RAISE EXCEPTION TYPE cx_my_exception. ENDMETHOD. ENDCLASS. DATA: ex TYPE REF TO cx_my_exception, oref TYPE REF TO c1. START-OF-SELECTION. TRY. CREATE OBJECT oref. oref->m1(). CATCH cx_my_exception INTO ex. write: /5 'My Exception caught'. ENDTRY. </pre>
Output	My exception caught

8.7 Nested TRY...ENDTRY block	
Theme	Try...Endtry blocks can be nested. The inner Try...endtry block can be in the guarded section of outer Try...Endtry block.
Program Descr.	<p>The program shown below uses one nested try...endtry statement.</p> <p>For the inner block, the error for division by zero is properly caught from the system provided exception-class : CX_SY_ZERODIVIDE.</p> <p>But, there is a character to integer conversion statement specified in the inner block, which creates an error of exception class CX_SY_CONVERSION_NO_NUMBER trapped by the outer Try...Endtry block.</p>
	<pre> REPORT YSUBCLASS_EXCEPTION_3. START-OF-SELECTION. DATA : num type i value 5 . TRY. TRY. NUM = NUM / 0. CATCH cx_sy_ZERODIVIDE . WRITE: /5 'Division by 0 caught' . NUM = ' SUBHENDU' . ENDTRY. CATCH cx_sy_conversion_no_number. WRITE: /5 'Cannot be converted to number' . ENDTRY. </pre>
Output	<p>Division by 0 caught</p> <p>Cannot be converted to number</p>

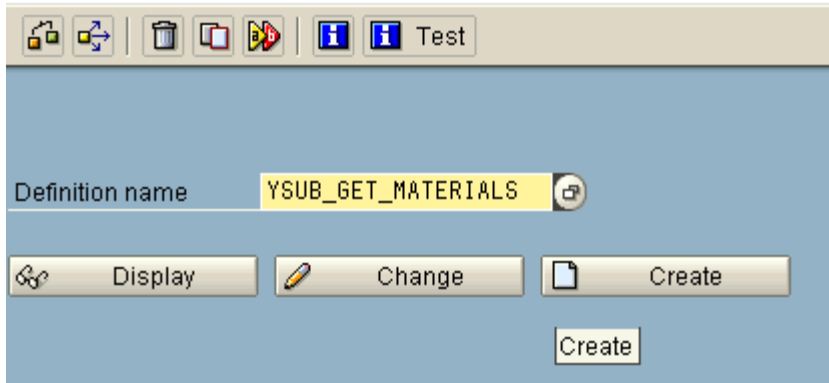
8.8 Use of CLEANUP section	
Theme	Clean up block is executed whenever an exception occurs within the TRY block and is not handled by a CATCH within the same TRY block, but is handled by an surrounding TRY block
Program Descr.	<p>This program uses a nested TRY...ENDTRY statement. In the inner block, there is an attempt to character-to-integer conversion, which raises an error of exception-class : CX_SY_CONVERSION_NO_NUMBER. But, the inner TRY block does not trap the error - rather, it is trapped by the outer TRY...ENDTRY block.</p> <p>Under such circumstances, the CLEANUP section of the inner block gets executed first - then the CATCH section of the outer block works.</p>
Dump	<pre>REPORT YSUBCLASS_EXCEPTION_3. START-OF-SELECTION. DATA : num type i value 5 . TRY. TRY. num = 'subhendu' . cleanup. write: /5 'In cleanup' . ENDTRY. CATCH cx_sy_conversion_no_number. WRITE: /5 'Cannot be converted to number' . ENDTRY.</pre>
Output	<p>In cleanup Cannot be converted to number</p>

9 BADIs (Business Add-Ins)

9.1 Single Implementation of BADI

Go to transaction SE18 and create a BADI Definition.

Business Add-Ins: Initial Definition Maintenance

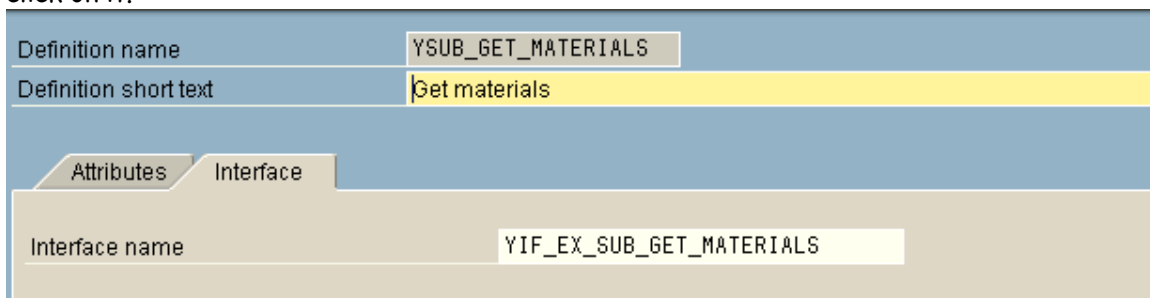


Definition name YSUB_GET_MATERIALS

Display Change Create

Create

Enter the description. The name of the interface will be automatically proposed. Double click on it.

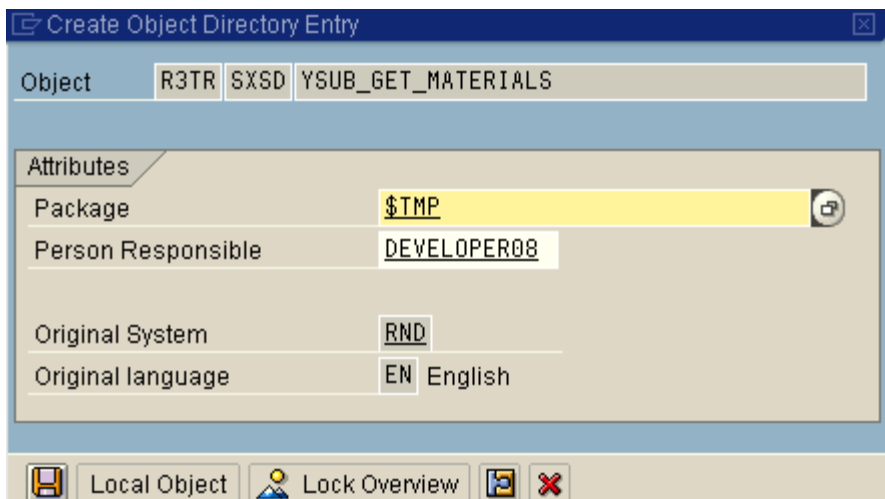


Definition name YSUB_GET_MATERIALS

Definition short text Get materials

Attributes Interface

Interface name YIF_EX_SUB_GET_MATERIALS



Create Object Directory Entry

Object R3TR SXSD YSUB_GET_MATERIALS

Attributes

Package \$TMP

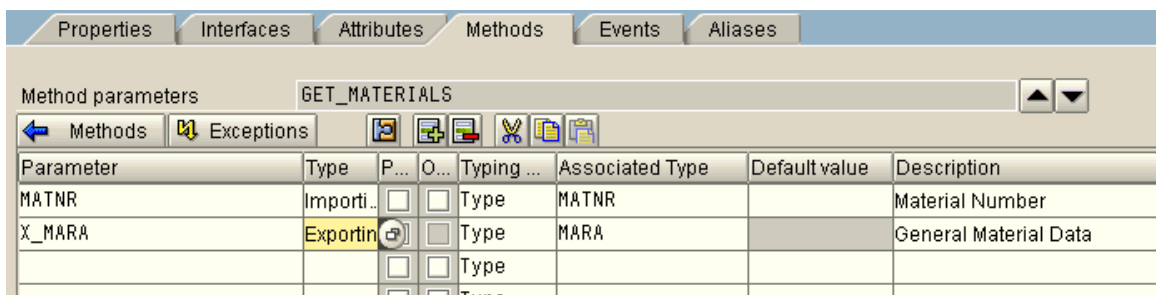
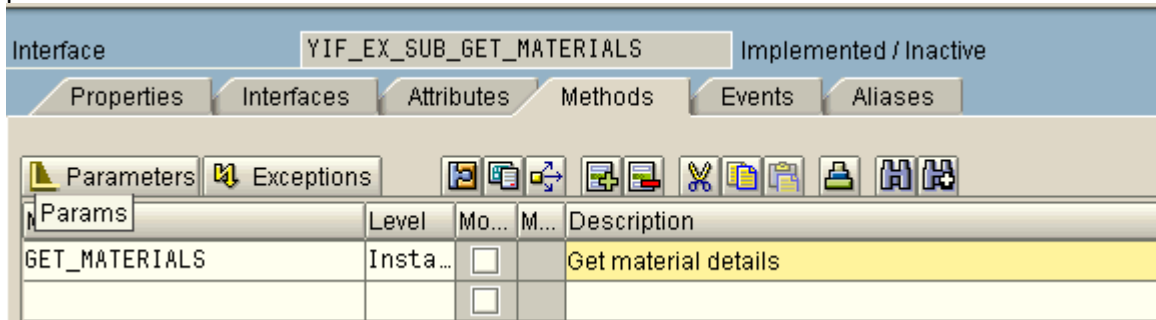
Person Responsible DEVELOPER08

Original System RND

Original language EN English

Local Object Lock Overview

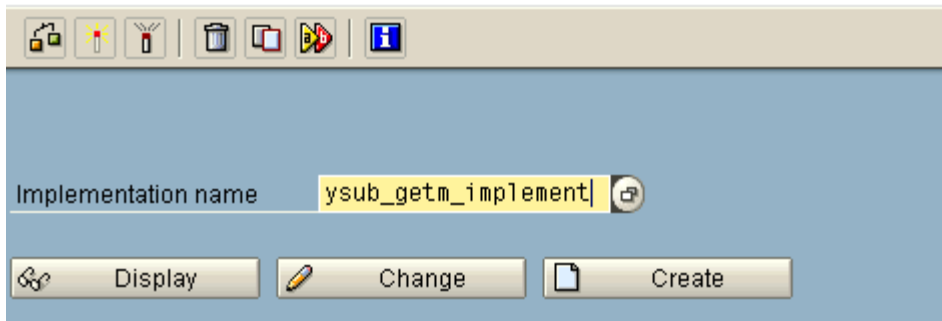
Enter the name of the method you want to create. Click the Parameters pushbutton to create parameters for the method.



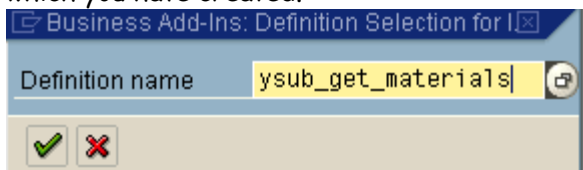
Save and activate it.

Go to transaction SE19 and create a BADI implementation.

Business Add-Ins: Initial Implementation Maintenance



A popup window will ask you for the definition name. Enter the name of the BADI definition which you have created.



Press Enter.

Enter the description for the implementation. Then save and activate it.

Business Add-In Builder: Change Implementation YSUB_GETM_IMPLME

The screenshot shows the 'Business Add-In Builder' window. At the top, there are icons for various functions and two tabs: 'Def. documentatn' and 'Documentation'. Below the icons, there are three input fields: 'Implementation name' with the value 'YSUB_GETM_IMPLMENT' and status 'Inactive', 'Implementation short text' with the value 'Implementation for BADI : ysub_get_materials', and 'Definition name' with the value 'YSUB_GET_MATERIALS'. Below these fields are two tabs: 'Attributes' and 'Interface'. The 'Interface' tab is selected, showing 'Interface name' as 'YIF_EX_SUB_GET_MATERIALS' and 'Name of implementing class' as 'YCL_IM_SUB_GETM_IMPLMENT'. At the bottom, there is a table with three columns: 'Method', 'Implement...', and 'Description'. The table contains one row: 'GET_MATERIALS', 'ABAP Co...', and 'Get material details'.

Method	Implement...	Description
GET_MATERIALS	ABAP Co...	Get material details

Double click on the method name.

The screenshot shows a 'Save' dialog box with a question mark icon. The text inside says: 'Business add-in implementation was changed: YSUB_GETM_IMPLMENT. Do you want to save?'. There are three buttons at the bottom: 'Yes', 'No', and 'Cancel'.

The screenshot shows the ABAP editor window. The title bar says 'Method YIF_EX_SUB_GET_MATERIALS~GET_MATERIALS Activ'. The editor contains the following code:

```

method YIF_EX_SUB_GET_MATERIALS~GET_MATERIALS .
  IF NOT MATNR IS INITIAL.
    SELECT SINGLE * FROM MARA INTO X_MARA
      WHERE MATNR = MATNR.
  endif.
endmethod.

```

Write the code. Save and activate it.

Then, create a code using the BADI

```
REPORT YSUB_GET_MATERIALS_PROGRAM.

CLASS cl_exithandler DEFINITION LOAD.
DATA : L_BADI_INSTANCE TYPE REF TO YIF_EX_SUB_GET_MATERIALS.
DATA : X_MARA LIKE MARA.

PARAMETERS : P_MATNR LIKE MARA-MATNR OBLIGATORY.

START-OF-SELECTION.

CALL METHOD CL_EXITHANDLER=>GET_INSTANCE
  EXPORTING EXIT_NAME = 'YSUB_GET_MATERIALS'
            NULL_INSTANCE_ACCEPTED = 'X'
  CHANGING INSTANCE = L_BADI_INSTANCE.

CALL METHOD L_BADI_INSTANCE->GET_MATERIALS
  EXPORTING MATNR = P_MATNR
  IMPORTING X_MARA = X_MARA.

WRITE:/5 X_MARA-MATNR ,
         X_MARA-MATKL ,
         X_MARA-MEINS .
```

9.2 Multiple Implementation

Create a BADI: YSUB_GET_MATERIALS_1 in a manner similar to that created above. But, check the checkbox for Multiple Use

Definition name: YSUB_GET_MATERIALS_1
 Definition short text: Get material details

Attributes Interface

General Data

Package: \$TMP Last changed by: DEVELOPER08
 Language: EN English Last change: 03.02.2004 11:27:51

Name of bus. add-in class: YCL_EX_SUB_GET_MATERIALS_1

Type

☒ Multiple use

Interface YIF_EX_SUB_GET_MATERIALS_1 Implemented / Active

Properties Interfaces Attributes Methods Events Aliases

Method parameters: GET_MATERIALS

Parameter	Type	P...	O...	Typing ...	Associated Type	Default value	Description
MATNR	Imported	<input type="checkbox"/>	<input type="checkbox"/>	Type	MATNR		Material Number
X_MARA	Changed	<input type="checkbox"/>	<input type="checkbox"/>	Type	MARA		General Material Data

Then, create one implementations for the same BADI:-

```
method YIF_EX_SUB_GET_MATERIALS_1~GET_MATERIALS .
  if not matnr is initial.
    select single * from mara into x_mara
      where matnr = matnr.
  endif.
endmethod.
```

(Code is similar to the previous one)

Then, create a program utilizing the BADI:-

```

REPORT YSUB_GET_MATERIALS_PROGRAM_1.

CLASS cl_exithandler DEFINITION LOAD.
DATA : L_BADI_INSTANCE TYPE REF TO YIF_EX_SUB_GET_MATERIALS_1.
DATA : X_MARA LIKE MARA.

PARAMETERS : P_MATNR LIKE MARA-MATNR OBLIGATORY.

START-OF-SELECTION.

CALL METHOD CL_EXITHANDLER=>GET_INSTANCE
  EXPORTING EXIT_NAME = 'YSUB_GET_MATERIALS_1'
  NULL_INSTANCE_ACCEPTED = 'X'
  CHANGING INSTANCE = L_BADI_INSTANCE.

CALL METHOD L_BADI_INSTANCE->GET_MATERIALS
  EXPORTING MATNR = P_MATNR
  CHANGING X_MARA = X_MARA.

WRITE:/5 X_MARA-MATNR ,
         X_MARA-MATKL ,
         X_MARA-MEINS .

```

Now, you want to create another implementation of the same BADI. Let us exemplify the concept. Say, you want that when the user will enter 'GARI' in the selection-screen, it will stand for 'CAR' internally and selection will be done out of MARA table based on material code : 'CAR'. O, you define another implementation of the same BADI from transaction : SE19.

Implementation name: YSUB_GETM_IMPL2

Buttons: Display, Change, Create

Implementation name: YSUB_GETM_IMPL2 Active

Implementation short text: material retrieval-2

Definition name: YSUB_GET_MATERIALS_1

Attributes Interface

Interface name: YIF_EX_SUB_GET_MATERIALS_1

Name of implementing class: YCL_IM_SUB_GETM_IMPL2

Method	Implement...	Description
GET_MATERIALS	ABAP Co...	get material details from mara

```
method YIF_EX_SUB_GET_MATERIALS_1~GET_MATERIALS .  
  data : l_matnr type matnr.  
  l_matnr = matnr.  
  if l_matnr = 'GARI'.  
    l_matnr = 'CAR'.  
  endif.  
  
  select single * from mara into x_mara  
    where matnr = l_matnr.  
  
endmethod.
```

Now, when you will execute the program and enter 'GARI' in the material code field in the selection-screen, it will get internally translated to 'CAR' when the second implementation will be active.

9.3 Searching for BADI in SAP Transaction and Implementing it

There is a business demand in ABC corporation . when the user will post goods receipt via transaction MIGO, he should enter same date in document date and posting date field. Else, an information message will ask the user to do that.

The Bill of Lading number should start with 'NP'

You, as a SAP Technical Consultant, is asked to translate this idea into the appropriate section of the code.

Your manager has asked you to use BADI instead of any user or field exits to implement the idea.

Go to the program behing MIGO and search for the phrase :

CL_EXITHANDLER=>GET_INSTANCE in the main program. This will show you the BADIs which can be implemented by you to incorporate the business demand.

Program	Found locs/short description
<input type="checkbox"/> LMIG06L2	<pre> 38 call method cl_exithandler=>get_instance EXPORTING exit_name = 'MB_MIGO_BADI' null_instance_accepted = x changing instance = if_badi. 59 call method cl_exithandler=>get_instance EXPORTING exit_name = 'MB_MIGO_ITEM_BADI' null_instance_accepted = x changing instance = if_badi_item. </pre>

So, you get to know now that there are two BADIs which can come to your use. They are:
MB_MIGO_BADI

Now, you have to go to transaction SE18 and explore each of the BADIs to find out the suitable one. In fact, the suitable one will have a method in it for which import/export parameters should have some reference to document/posting dates.

On investigation, you will find that in BADI : MB_MIGO_ITEM_BADI, there is a method : ITEM_MODIFY which uses : is_gohead as import parameter. This has the structure GOHEAD which contains the fields for document and posting date as columns.

Now, you will implement this BADI. Go to transaction SE19 and create an implementation for the BADI. In the code for the method, write the following:-

```
method IF_EX_MB_MIGO_ITEM_BADI~ITEM_MODIFY .

DATA : L_INI(2) TYPE C.

IF is_gohead-bldat ne is_gohead-budat.
  message i398(00) with 'Both posting and document dates should be same'.
ENDIF.

L_INI = IS_GOHEAD-FRBNR+0(2).
TRANSLATE L_INI TO UPPER CASE.
IF L_INI NE 'NP'.
  MESSAGE I398(00) WITH 'Bill of Lading should start with NP'.
ENDIF.

endmethod.
```

Then, save and activate it. Then, perform a transaction via MIGO. Your requirement will be fulfilled.

9.4 Menu Enhancements

SAP allows you to enhance menus in its user interfaces using function codes. These function codes must adhere to the form /namespace/+, just like in SMOD/CMOD enhancements. They are assigned to a specific enhancement and only appear in their corresponding menus once an implementation of this enhancement has been activated.

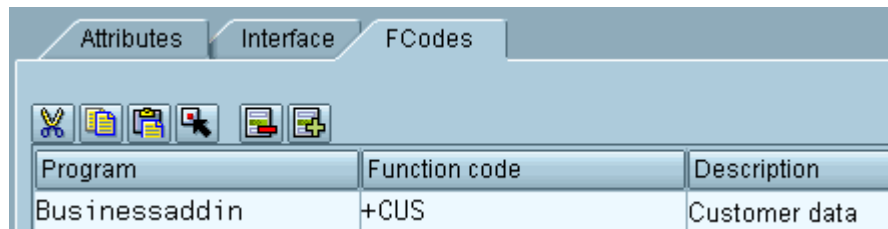
Application developers reserve specific function codes for customers when defining a Business Add-In. They use the Menu Painter to include these codes in the appropriate menu lists. Application developers must also ensure that these menu options are called in their applications and that the corresponding add-in methods are also retrieved. Customers can take advantage of menu enhancements by creating a new implementation, choosing a text for the menu option, and then programming the method used to determine what action is performed when the menu enhancement is called.



Menu enhancement is only possible using single use add-ins (not multiple use add-ins) that are not filter-dependent. Currently, menu enhancements can only be created in conjunction with program enhancements (interfaces).

To create a menu enhancement, proceed as follows:

1. Create an add-in and define its interface.
2. Choose *Fcodes* from the tabstrip.
3. Enter the name of your program, the function code, and a description.



Program	Function code	Description
Businessaddin	+CUS	Customer data

4. Call the Menu Painter or double-click on your program name or function code to branch to user interface maintenance in the Menu Painter. Enter your function code in the appropriate menu list. If you have accessed the Menu Painter directly during add-in definition, you can call your menu lists by choosing *Goto --> Object lists --> Menu list* instead.

Calling a Menu Enhancement from an Application Program

Your programming should look like this:

```
(...)  
case fcode.  
  when 'SAP'.  
    (...)
```

Implementing a Menu Enhancement

When implementing menu enhancements, proceed as follows:

1. Create an implementation and choose *Fcodes*. All data adopted from your Business Add-In's definition is displayed here. You can make entries for the implementation on the right. You can also double-click on the first input field. The following dialog box appears:

Program	Function code	Description	Function text
BUSINESSADDIN	+CUS	Display customer data	ICON_DISPLAY

Here you may enter a text for your function code, the name of an icon and a text for the icon, and a short informational text.

The actions that you want the system to perform after the pushbutton is chosen must be programmed in the appropriate method, either manually or using default source code that has been provided to you.

Menu enhancements only become visible after the implementation has been activated and the application program that calls the Business Add-In has been executed.