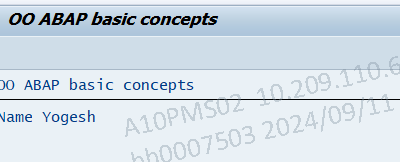
**OO ABAP Example programs**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Contents** | **Page NO** |
| **Class** | | | |
| A | Simple class and object creation program (Not in PDF) |  |
| B | Using static attributes and methods. (Not in PDF) |  |
|  |  |  |
| 1.1 | Access specifiers data access in main and sub class and outside the class. |  |
| 1.2 |  |  |
| 1.3 | Private and public variables can’t be used outside the class. |  |
| 1.4 | Common and class global data.. |  |
| 1.5 | Public section object only create in another class same private and protected section impossible... |  |
| 1.6 | How can refer to a class without defing the class but class has to be defined on. |  |
| 1.7 |  |  |
| 1.8 | FS can be used to contain value of any variable ie, instance and static variables.. |  |
| 1.9 | Static attributes of a class are retained throughout the entire runtime ie, variables did not change any class.. |  |
| 1.10 | Create and implement global class in se38 ref.1.10 |  |
| **Methods** | | |
| 2.1 | Different ways of calling a method which has only one import parameter |  |
| 2.2 | Parameters passed by value can be changed internally in a method but parameters pass by reference cannot changed in the method. |  |
| **S.No** | **Contents** | **Page NO** |
| 2.3 | Prefered parameter have more that one parameter that time get more preference compared to others parameter. |  |
| 2.4 | Exporting importing and changing parameter used in methods |  |
| 2.5 | Use internal table parameters in methods |  |
| 2.6 | Returning parameter pass by value 3 different methods |  |
| 2.7 | Declare static method |  |
| 2.8 | Static method only access class attributes & instance attributes can use both |  |
| 2.9 | How to raise exception in class. |  |
| 2.10 | Method call method itself |  |
| 2.11 | Access variable different in declaration class and methods |  |
| 2.12 | How to use the pointer table |  |
| 2.13 | Methods and class pass a dynamic value |  |
| 2.14 |  |  |
| 2.15 | Exception table used to handle the exceptions |  |
| **Constructor** | | |
| 3.1 | Instance constructor trigger method |  |
| 3.2 | Instance constructor with import method |  |
| 3.3 | Constructor methods have no export parameter |  |
| 3.4 | Constructor methods using import and exceptions |  |
| 3.5 | Constructor triggered when using static attribute |  |
| 3.6 | Static constructor can be triggered at the beginning of a processing block |  |
| **S.No** | **Contents** | **Page NO** |
| 3.7 | Static constructor of a class cannot have parameter or exceptions |  |
| **Inheritance** | | |
| 4.1 | Sub class can access public/protected component of super class |  |
| 4.2 | Sub class can re-implement inherited methods from super class |  |
| 4.3 | Abstract class can’t create object only derived class |  |
| 4.4 | Abstract class can’t implement in main class |  |
| 4.5 | Sub class can’t be inherit from final class |  |
| 4.6 | Final method in class cannot be redefine in subclass |  |
| 4.7 | Static attribute can change the outside the class also |  |
| 4.8 | Constructor of super class can be inherited by sub class also |  |
| 4.9 | Sub class can modify the constructor methods |  |
| 4.10 | When you called subclass first in static constructor system execute first super class |  |
| 4.11 | Static type reference variable point super class, dynamic type point one of it subclass |  |
| 4.12 | Reference variable not able to identify any new component in sub class, which not present in super class |  |
| 4.13 | Using private variable inherit from super class not from subclass even if sub class have private variable same name |  |
| 4.14 | Use of widening cast operator |  |
| **Interface** | | |
| 5.1 | Simple interface program |  |
| 5.2 | Interface only use public section |  |
| 5.3 | Interface all method should implement in program |  |
| 5.4 | Value for the interface attributes are assigned at the time of inclusion in a class |  |
| 5.5 | Use final method in interface |  |
| 5.6 | Use of abstract method from interface |  |
| 5.7 | Use of interface how to flow the reference variable study |  |
| 5.8 | Interface can be included another interface final interface containing all the interface inside the class |  |
| 5.9 | Using Aliases in interface |  |
| 5.10 | How method of an interface can be interpreted differently in different class |  |
| **Friends** | | |
| 6.1 | Friendship class grant permission to another class. |  |
| 6.2 | Sub class also friends with super class so super class friend also friends with sub class |  |
| 6.3 | Friend ship is one sided, class granting the friendship not automatically a friend of its friend |  |
| **Events** | | |
| 7.1 | Events in handler method in same class |  |
| 7.2 | Event with event handler method is in different class |  |
| 7.3 | More than one event handler method can exist for same event |  |
| 7.4 | Use of static events |  |
| 7.5 | Event with export parameter |  |
| **Class based exceptions** | | |
| 8.1 | How to avoid ABAP dump simple program |  |
| 8.2 | When both super and subclass are used to track error |  |
| 8.3 | Using subroutine in class based exceptions |  |
| 8.4 | Program can raise exceptions based on SAP standard exception classes |  |
| 8.5 | Object is created when trap a exception class |  |
| 8.6 | Demo on locally defined exception class |  |
| 8.7 | Outer nested try block also caught exceptions |  |
| 8.8 | Use clean up section |  |
| **BADI** | | |
|  |  |  |
|  |  |  |

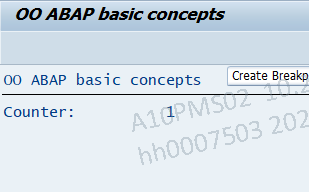
A. REPORT ZYP\_OO\_ABAP\_BASIC. *" Simple class & object creation program*  
  
CLASS LO\_CLASSA DEFINITION.  
  PUBLIC SECTION.  
    METHODS DISPLAY\_NAME.  
    DATA LV\_NAME TYPE STRING.  
ENDCLASS.  
\*&---------------------------------------------------------------------\*  
\*& CLASS (IMPLEMENTATION) LO\_CLASSA  
\*&---------------------------------------------------------------------\*  
\*&  
\*&---------------------------------------------------------------------\*  
CLASS LO\_CLASSA IMPLEMENTATION.  
  METHOD DISPLAY\_NAME.  
    WRITE: / 'NAME', LV\_NAME.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
DATA: LO\_OBJECT TYPE REF TO LO\_CLASSA.  
CREATE OBJECT LO\_OBJECT.  
LO\_OBJECT->LV\_NAME = 'YOGESH'.  
LO\_OBJECT->DISPLAY\_NAME ( ).

Output:



B. REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS DEMO\_STATIC\_CLASS DEFINITION. " USING STATIC ATTRIBUTE AND METHODS  
  PUBLIC SECTION.  
    CLASS-DATA: COUNTER TYPE I.  
    CLASS-METHODS: INCREMENT\_COUNTER, DISPLAY\_COUNTER.  
ENDCLASS.  
“ If I am using instance attribute also same result. Note: Same method variable does not clear..  
CLASS DEMO\_STATIC\_CLASS IMPLEMENTATION.  
  METHOD INCREMENT\_COUNTER.  
    COUNTER = COUNTER + 1.  
  ENDMETHOD.  
  
  METHOD DISPLAY\_COUNTER.  
    WRITE: /'COUNTER:', COUNTER.  
    ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  
DEMO\_STATIC\_CLASS=>INCREMENT\_COUNTER( ).  
DEMO\_STATIC\_CLASS=>DISPLAY\_COUNTER( ).

Output:



1.1. REPORT ZYP\_OO\_ABAP\_BASIC.

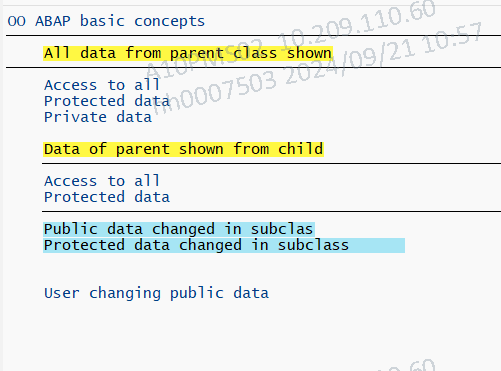
Private data access main class only.

Protected data access main class and sub class.

Public data access main class, sub class, and outside the class.  
CLASS PARENT\_CLASS DEFINITION.                                          *" PARENT CLASS*  
  PUBLIC SECTION.  
    DATA: COMMON\_DATA (30) TYPE C VALUE 'ACCESS TO ALL'.  
    METHODS SHOW\_VALUE.  
  PROTECTED SECTION.  
    DATA: PROTECTED\_DATA (40) TYPE C VALUE 'PROTECTED DATA'.  
  PRIVATE SECTION.  
    DATA: PRIVATE\_DATA (30) TYPE C VALUE 'PRIVATE DATA'.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) PARENT\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS PARENT\_CLASS IMPLEMENTATION.               *" PARENT CLASS IMPLEMENTATION CANN ACCESS ALL ACCESS DATA.*  
  METHOD SHOW\_VALUE.  
    WRITE: /5 'ALL DATA FROM PARENT CLASS SHOWN' COLOR 3.  
    WRITE: / SY-ULINE.  
    WRITE: /5 COMMON\_DATA,  
           /5 PROTECTED\_DATA,  
           /5 PRIVATE\_DATA.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS CHILD\_CLASS DEFINITION INHERITING FROM PARENT\_CLASS.            *" CHILD CLASS*  
  PUBLIC SECTION.  
    METHODS SUB\_VALUE.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CHILD\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CHILD\_CLASS IMPLEMENTATION.                                     *" INHERITANCE CLASS CAN'T ACCESS PRIVATE DATA.*  
  METHOD SUB\_VALUE.  
    SKIP 1.  
    WRITE:/5 'DATA OF PARENT SHOWN FROM CHILD' COLOR 3.  
    WRITE:/5 SY-ULINE,  
          /5 COMMON\_DATA,  
          /5 PROTECTED\_DATA.  
    COMMON\_DATA    = 'PUBLIC DATA CHANGED IN SUBCLASS'.  
    PROTECTED\_DATA = 'PROTECTED DATA CHANGED IN SUBCLASS'.  
    WRITE: /5 SY-ULINE.  
    WRITE: /5 COMMON\_DATA COLOR 1,  
           /5 PROTECTED\_DATA COLOR 1.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  
  DATA: PARENT TYPE REF TO PARENT\_CLASS,  
        CHILD  TYPE REF TO CHILD\_CLASS.  
  
  CREATE OBJECT : PARENT, CHILD.  
  
  CALL METHOD : PARENT->SHOW\_VALUE,  
                CHILD->SUB\_VALUE.  
  SKIP 2.  
  
  PARENT->COMMON\_DATA = 'USER CHANGING PUBLIC DATA'.

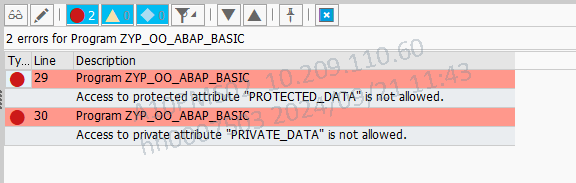
*" OUTSIDE CLASS CAN'T ACCESS PRIVATE ANS PROTECTED DATA.*  
  
  WRITE: /5 PARENT->COMMON\_DATA.

Output:



1.3.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS PARENT\_CLASS DEFINITION. *" Private and public variables can’t used in outside the class*  
  PUBLIC SECTION.  
    DATA COMMON\_DATA(30)    TYPE C VALUE 'ACCESS TO ALL'.  
  PROTECTED SECTION.  
    DATA PROTECTED\_DATA(30) TYPE C VALUE 'ONLY SUB CLASS'.  
  PRIVATE SECTION.  
    DATA PRIVATE\_DATA(30)   TYPE C VALUE 'ONLY MAIN CLASS'.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) PARENT\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS PARENT\_CLASS IMPLEMENTATION.  
    
ENDCLASS.  
  
START-OF-SELECTION.  
DATA: OBJECT\_1 TYPE REF TO PARENT\_CLASS.  
CREATE OBJECT OBJECT\_1.  
WRITE: /5 OBJECT\_1->COMMON\_DATA,  
          OBJECT\_1->PROTECTED\_DATA,  
          OBJECT\_1->PRIVATE\_DATA.

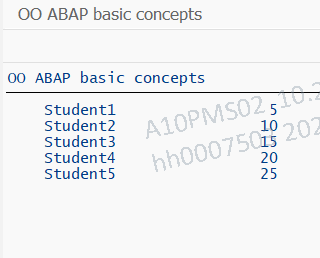
Output:



1.4.REPORT ZYP\_OO\_ABAP\_BASIC.

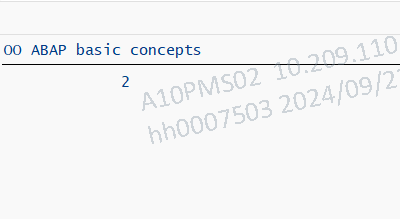
*" GLOBAL DATA OF THE CLASS AND OUTSIDE DATA OF THE CLASS DIRECTLY DECLERE A PROGRAM*  
  
TYPES: BEGIN OF TY\_TAB,  
         NAME(15) TYPE C,  
         AGE      TYPE I,  
       END OF TY\_TAB.  
  
DATA NUM1 TYPE I VALUE 5.  
  
CLASS PARENT\_CLASS DEFINITION.  
  PUBLIC SECTION.  
    METHODS METHOD1.  
    DATA: L\_NUM  LIKE NUM1,  
          LT\_TAB TYPE STANDARD TABLE OF TY\_TAB,  
          LS\_TAB LIKE LINE OF LT\_TAB.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) PARENT\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS PARENT\_CLASS IMPLEMENTATION.  
  METHOD METHOD1.  
    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 LS\_TAB-NAME.  
      LS\_TAB-AGE = NUM1 \* L\_NUM.  
      APPEND LS\_TAB TO LT\_TAB.  
      CLEAR LS\_TAB.  
    ENDDO.  
    LOOP AT LT\_TAB INTO LS\_TAB.  
      WRITE: /5 LS\_TAB-NAME,  
                LS\_TAB-AGE.  
    ENDLOOP.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: OBJ1 TYPE REF TO PARENT\_CLASS.  
  
  CREATE OBJECT: OBJ1.  
  CALL METHOD OBJ1->METHOD1.

Output:



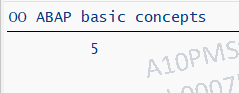
1.5.REPORT ZYP\_OO\_ABAP\_BASIC. *" PUBLIC SECTION OBJECT ONLY CREATE IN ANOTHER CLASS SAME PRIVATE AND PROTECTED SECTION IMPOSSIBLE*  
  
CLASS CLASS\_1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS METHOD\_1.  
ENDCLASS.  
  
CLASS CLASS\_2 DEFINITION.  
  PUBLIC SECTION.  
    METHODS METHOD\_2.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_1 IMPLEMENTATION.  
  METHOD METHOD\_1.  
    DATA I\_NUM TYPE I VALUE 2.  
    WRITE: /5 I\_NUM.  
  ENDMETHOD.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_2*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_2 IMPLEMENTATION.  
  METHOD METHOD\_2.  
    DATA OBJECT\_1 TYPE REF TO CLASS\_1.  
    CREATE OBJECT OBJECT\_1.  
    CALL METHOD OBJECT\_1->METHOD\_1.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA OBJECT\_2 TYPE REF TO CLASS\_2.  
  CREATE OBJECT OBJECT\_2.  
  CALL METHOD OBJECT\_2->METHOD\_2.

Output:



1.6.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS CLASS\_2 DEFINITION DEFERRED. *" HOW CAN REFER TO A CLASS WITHOUT DEFING THE CLASS BUT CLASS HAS TO BE DEFINED ON.*  
  
CLASS CLASS\_1 DEFINITION.  
  PUBLIC SECTION.  
    DATA LV\_OBJECT TYPE REF TO CLASS\_2.  
ENDCLASS.  
  
CLASS CLASS\_2 DEFINITION.  
  PUBLIC SECTION.  
    DATA LV\_NUM TYPE I VALUE 5.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA LO\_OBJECT\_1 TYPE REF TO CLASS\_1.  
  CREATE OBJECT LO\_OBJECT\_1.  
  CREATE OBJECT LO\_OBJECT\_1->LV\_OBJECT.  
  WRITE LO\_OBJECT\_1->LV\_OBJECT->LV\_NUM.

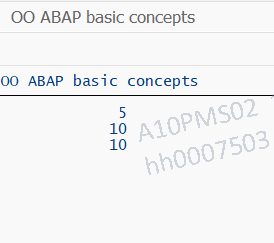
Output:



1.8.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
FIELD-SYMBOLS <FS> TYPE ANY.

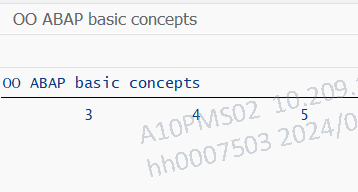
*" FS CAN BE USED TO CONTAIN VALUE OF ANY VARIABLE IE, INSTANCE AND STATIC VARIABLES*  
  
CLASS CLASS\_1 DEFINITION.  
  PUBLIC SECTION.  
    DATA: I\_NUM TYPE I VALUE 5.       *" INSTANCE ATTRIBUTE*  
    CLASS-DATA O\_NUM TYPE I VALUE 10. *" STATIC ATTRIBUTE*  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_1 IMPLEMENTATION.  
  
ENDCLASS.  
  
START-OF-SELECTION.  
  
  DATA O\_REF1 TYPE REF TO CLASS\_1.  
  CREATE OBJECT O\_REF1.  
  *" ASSIGN INSTANCE ATTRIBUTE TO FS*  
  ASSIGN O\_REF1->I\_NUM TO <FS>.  
  WRITE /5 <FS>.  
  
  *" ASSIGN STATIC ATTRIBUTE TO FS*  
  ASSIGN O\_REF1->O\_NUM TO <FS>.  
  WRITE /5 <FS>.  
  ASSIGN CLASS\_1=>O\_NUM TO <FS>.  
  WRITE /5 <FS>.

Output:



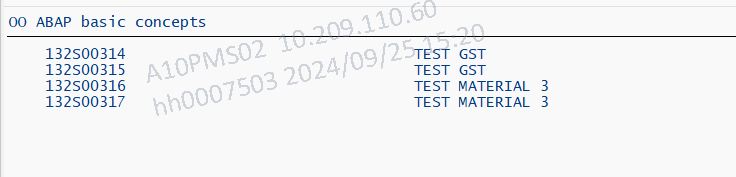
1.9.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS CLASS\_1 DEFINITION. *" STATIC ATTRIBUTES OF A CLASS ARE RETAINED THROUGHOUT THE ENTIRE RUNTIME IE, VARIABLES DID NOT CHANGE ANY CLASS*  
  PUBLIC SECTION.  
    CLASS-DATA NUM TYPE I.  
    METHODS METHOD\_1.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_1 IMPLEMENTATION.  
  METHOD METHOD\_1.  
    NUM = NUM + 1.  
    WRITE: NUM.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  
  CLASS\_1=>NUM = 3.  
  WRITE CLASS\_1=>NUM.  
  
  DATA: O\_REF1 TYPE REF TO CLASS\_1,  
        O\_REF2 TYPE REF TO CLASS\_1.  
  
  CREATE OBJECT: O\_REF1, O\_REF2.  
  CALL METHOD O\_REF1->METHOD\_1.  
  CALL METHOD O\_REF2->METHOD\_1.

Output:



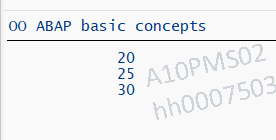
1.10.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
TYPES: BEGIN OF TY\_MATNR,           *" CREATE AND IMPLEMENT GLOBAL CLASS IN SE38 REF.1.10*  
         MATNR LIKE MARA-MATNR,  
         MAKTG LIKE MAKT-MAKTG,  
       END OF TY\_MATNR.  
  
DATA: LT\_MATNR TYPE STANDARD TABLE OF TY\_MATNR,  
      LS\_MATNR TYPE TY\_MATNR.  
  
PARAMETERS P\_MTART LIKE MARA-MTART OBLIGATORY.  
  
START-OF-SELECTION.  
  DATA O\_REF TYPE REF TO ZYP\_GET\_MATERIALS.  
  CREATE OBJECT O\_REF.  
  
  CALL METHOD O\_REF->LIST\_MATERIALS  
    EXPORTING  
      L\_MTART            = P\_MTART  
    IMPORTING  
      MATERIAL\_LIST      = LT\_MATNR  
    EXCEPTIONS  
      MATERIAL\_NOT\_FOUND = 1  
      OTHERS             = 2.  
  
  IF SY-SUBRC <> 0.  
    WRITE: /5 'MATERIALS NOT FOUND'.  
  ELSE.  
    LOOP AT LT\_MATNR INTO LS\_MATNR.  
      WRITE: /5 LS\_MATNR-MATNR,  
                LS\_MATNR-MAKTG.  
    ENDLOOP.  
  ENDIF.

Output:



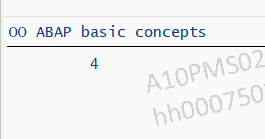
2.1.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS CLASS\_1 DEFINITION. *" Different ways of calling a method which has only one import parameter*  
  PUBLIC SECTION.  
    DATA: NUM TYPE I VALUE 5.  
    METHODS METHOD IMPORTING INPUT1 TYPE I.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_1 IMPLEMENTATION.  
  METHOD METHOD.  
    NUM = NUM \* INPUT1.  
    WRITE: /5 NUM.  
    NUM = 5.  
    ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
DATA: O\_REF1 TYPE REF TO CLASS\_1.  
CREATE OBJECT O\_REF1.  
  
CALL METHOD O\_REF1->METHOD EXPORTING INPUT1 = 4.  
CALL METHOD O\_REF1->METHOD( INPUT1 = 5 ).  
CALL METHOD O\_REF1->METHOD( 6 ).

Output:



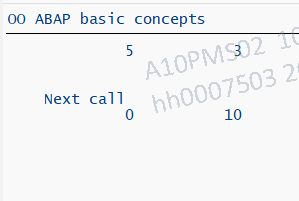
2.2.REPORT ZYP\_OO\_ABAP\_BASIC. *" PARAMETERS PASSED BY VALUE CAN BE CHANGED INTERNALLY IN A METHOD BUT PARAMETERS PASS BY REFERENCE CANNOT CHANGED IN THE METHOD.*  
  
DATA: NUM TYPE I.  
CLASS CLASS\_1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS METHOD\_1 IMPORTING INPUT\_1 TYPE I VALUE(INPUT\_2) TYPE I.  
ENDCLASS.  
  
CLASS CLASS\_1 IMPLEMENTATION.  
  METHOD METHOD\_1.  
    INPUT\_2 = 4.  
*\*    INPUT\_1 = 4. " YOU CANT GIVE THIS THIS IS PASS BY REFERENCE VALUE*  
    WRITE: INPUT\_1.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA O\_REF1 TYPE REF TO CLASS\_1.  
  CREATE OBJECT O\_REF1.  
  NUM = 3.  
  CALL METHOD O\_REF1->METHOD\_1  
    EXPORTING  
      INPUT\_1 = 4  
      INPUT\_2 = NUM.

Output



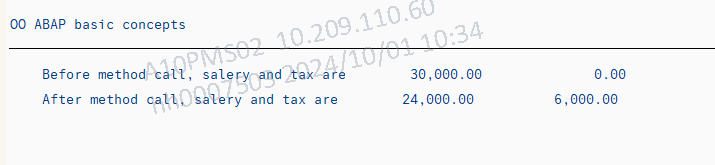
2.3.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS CLASS\_1 DEFINITION. *" PREFERED PARAMETER HAVE MORE THAT ONE PARAMETER THAT TIME GET MORE PREFERENCE  COMPARED TO OTHERS PARAMETER.*  
  PUBLIC SECTION.  
    METHODS METHOD\_1 IMPORTING INPUT\_1 TYPE I OPTIONAL INPUT\_2 TYPE I OPTIONAL PREFERRED PARAMETER INPUT\_2.  
ENDCLASS.  
  
CLASS CLASS\_1 IMPLEMENTATION.  
  METHOD METHOD\_1.  
    WRITE: /5 INPUT\_1, INPUT\_2.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  
  DATA: O\_REF1 TYPE REF TO CLASS\_1.  
  CREATE OBJECT O\_REF1.  
  CALL METHOD O\_REF1->METHOD\_1( INPUT\_1 = 5 INPUT\_2 = 3 ).  
  SKIP 2.  
  WRITE: /5 'NEXT CALL'.  
  CALL METHOD O\_REF1->METHOD\_1( 10 ).

Output:



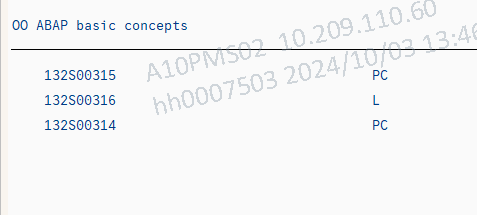
2.4.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
DATA: LV\_TAX TYPE P DECIMALS 2,  
      LV\_SAL TYPE P DECIMALS 2.  
  
CLASS C\_TAX DEFINITION. *" EXPORTING IMPORTING AND CHANGING PARAMETER USED IN METHODS*  
  PUBLIC SECTION.  
    METHODS TAX\_CALC IMPORTING GRADE  TYPE C  
                     EXPORTING I\_TAX  TYPE P  
                     CHANGING  SALERY TYPE P.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C\_TAX*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C\_TAX IMPLEMENTATION.  
  METHOD TAX\_CALC.  
    CASE GRADE.  
      WHEN 'A01'.  
        I\_TAX = SALERY \* '0.2'.  
      WHEN 'A02'.  
        I\_TAX = SALERY \* '0.1'.  
      WHEN OTHERS.  
        I\_TAX = SALERY \* '0.15'.  
    ENDCASE.  
    SALERY = SALERY - I\_TAX.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA O\_REF1 TYPE REF TO C\_TAX.  
  CREATE OBJECT O\_REF1.  
  LV\_SAL = 30000.  
  LV\_TAX = 0.  
  WRITE: /5 'BEFORE METHOD CALL, SALERY AND TAX ARE', LV\_SAL, LV\_TAX.  
  CALL METHOD O\_REF1->TAX\_CALC  
    EXPORTING  
      GRADE  = 'A01'  
    IMPORTING  
      I\_TAX  = LV\_TAX  
    CHANGING  
      SALERY = LV\_SAL.  
  WRITE: /5 'AFTER METHOD CALL, SALERY AND TAX ARE', LV\_SAL, LV\_TAX.

Output:



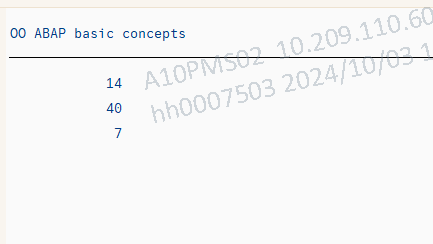
2.5.REPORT ZYP\_OO\_ABAP\_BASIC. *“Use internal table parameters in methods*  
  
TYPES: BEGIN OF TY\_TABLE,  
         MATNR LIKE MARA-MATNR,  
         MEINS LIKE MARA-MEINS,  
       END OF TY\_TABLE.  
  
  
DATA: LT\_TABLE TYPE STANDARD TABLE OF TY\_TABLE,  
      LS\_TABLE LIKE LINE OF LT\_TABLE.  
  
CLASS GET\_MATERIALS DEFINITION.  
  PUBLIC SECTION.  
    METHODS GET\_MARA IMPORTING MATGR TYPE C  
                     EXPORTING L\_TAB TYPE ANY TABLE.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) GET\_MATERIALS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS GET\_MATERIALS IMPLEMENTATION.  
  METHOD GET\_MARA.  
    SELECT MATNR MEINS INTO TABLE LT\_TABLE FROM MARA WHERE MATKL = MATGR.  
  ENDMETHOD.  
ENDCLASS.  
  
PARAMETERS P\_MATKL LIKE MARA-MATKL.  
  
START-OF-SELECTION.  
  
  DATA: LO\_MATNR TYPE REF TO GET\_MATERIALS.  
  CREATE OBJECT LO\_MATNR.  
  CALL METHOD LO\_MATNR->GET\_MARA  
    EXPORTING  
      MATGR = P\_MATKL  
    IMPORTING  
      L\_TAB = LT\_TABLE.  
  
  LOOP AT LT\_TABLE INTO LS\_TABLE.  
    WRITE:/5 LS\_TABLE-MATNR,  
             LS\_TABLE-MEINS.  
  ENDLOOP.

Output:



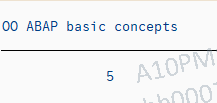
2.6.REPORT ZYP\_OO\_ABAP\_BASIC.     *" RETURNING PARAMETER PASS BY VALUE 3 DIFFERENT METHODS*  
  
*" IF USE RETURNING PARAMETER FOLLOWING RESTRICTIONS APPLY*  
*" NO EXPORTING / CHANGING PARAMETER CAN BE USED*  
*" ONLY RETURNING PARAMETER CAN BE USED*  
*" RETURNING PARAMETER ARE OLY PASSED BY VALUE*  
  
DATA W\_NUM TYPE I.  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS M1 IMPORTING INPUT\_1       TYPE I  
                         INPUT\_2       TYPE I  
               RETURNING VALUE(RESULT) TYPE I.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    RESULT = INPUT\_1 \* 2 + INPUT\_2.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_OBJECT\_1 TYPE REF TO C1.  
  CREATE OBJECT LO\_OBJECT\_1.  
  
  *"SYNTEX 1*  
  CALL METHOD LO\_OBJECT\_1->M1  
    EXPORTING  
      INPUT\_1 = 5  
      INPUT\_2 = 4  
    RECEIVING  
      RESULT  = W\_NUM.  
  
  WRITE: /5 W\_NUM.  
  
  *"SYNTEX 2*  
  W\_NUM = LO\_OBJECT\_1->M1( INPUT\_1 = 10 INPUT\_2 = 20 ).  
  WRITE:/5 W\_NUM.  
  
  *"SYNTEX 3*  
  MOVE LO\_OBJECT\_1->M1( INPUT\_1 = 2 INPUT\_2 = 3 ) TO W\_NUM.  
  WRITE: /5 W\_NUM.

Output:



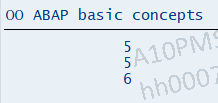
2.7. REPORT ZYP\_OO\_ABAP\_BASIC. *"DECLARE STATIC METHOD*  
  
DATA NUM TYPE I.  
  
CLASS CLASS\_A DEFINITION.  
  PUBLIC SECTION.  
    CLASS-METHODS C\_METHOD.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_A*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_A IMPLEMENTATION.  
  METHOD C\_METHOD.  
    NUM = 5.  
    WRITE: /5 NUM.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  
  CALL METHOD CLASS\_A=>C\_METHOD.

Output:



2.8.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION.   *" STATIC METHORD CAN'T ALLOW WITH INSTANCE ATTRIBUTE.*  
   *" STATIC METHODS OF A CLASS CAN ONLY USE STATIC ATTRIBUTES*  
   *" INSTANCE ATTRIBUTES CAN USE BOTH*  
  PUBLIC SECTION.  
    CLASS-DATA: ST\_NUM TYPE I VALUE 5.  
    DATA INST\_NUM TYPE I VALUE 6.  
    CLASS-METHODS ST\_METH.  
    METHODS INST\_METH.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
  METHOD ST\_METH.  
    WRITE: /5 ST\_NUM.  
*\*    WRITE: /5 INST\_NUM.    " STATIC METHOD ONLY ACCESS CLASS ATTRIBUTES*  
  ENDMETHOD.  
  
  METHOD INST\_METH.  
    WRITE:/5 ST\_NUM.  
    WRITE:/5 INST\_NUM.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: O\_REF1 TYPE REF TO C1.  
  CALL METHOD C1=>ST\_METH.  
  CREATE OBJECT O\_REF1.  
  CALL METHOD O\_REF1->INST\_METH.

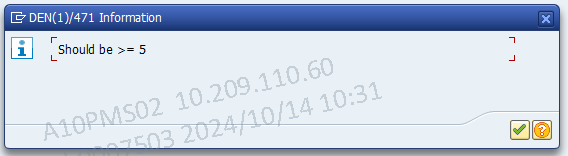
Output:



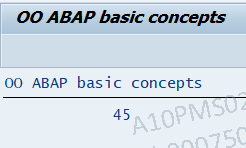
2.9.REPORT ZYP\_OO\_ABAP\_BASIC. *" HOW TO RAISE EXCEPTION IN CLASS.*  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS : M1 IMPORTING  NUM1 TYPE I  
                 EXPORTING  NUM2 TYPE I  
                 EXCEPTIONS E1.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    IF NUM1 LE 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.

Output:

Input 1

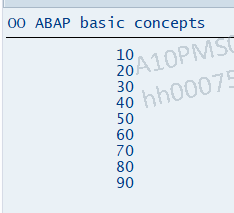


Input 9



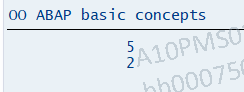
2.10.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION.  *" METHOD CALL METHOD ITSELF*  
  PUBLIC SECTION.  
    DATA: STATNUM TYPE I.  
    METHODS M1.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    STATNUM = STATNUM + 10.  
    IF STATNUM GE 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.

Output:



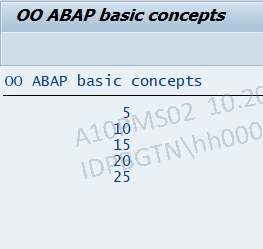
2.11.REPORT ZYP\_OO\_ABAP\_BASIC.     *" ACCESS VARIABLE DIFFERENT IN DECLERATION CLASS AND METHODS*  
  
CLASS TEST\_CLASS DEFINITION.  
  PUBLIC SECTION.  
    DATA: I\_NUM TYPE I VALUE 5.  
    METHODS: TEST\_METHOD.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) TEST\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS TEST\_CLASS IMPLEMENTATION.  
  METHOD TEST\_METHOD.  
    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 TEST\_CLASS.  
  CREATE OBJECT MY\_OBJ.  
  CALL METHOD MY\_OBJ->TEST\_METHOD.

Output:



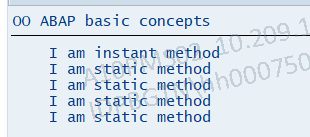
2.12.REPORT ZYP\_OO\_ABAP\_BASIC.  
*" THIS PROGRAM IS HOW TO USE THE POINTER TABLE*  
*" CREATE A OBJECT AND STORE A TABLE AND RETRIVE AGAIN*  
CLASS TEST\_CLASS DEFINITION.  
  PUBLIC SECTION.  
    METHODS: TEST\_METHOD.  
    CLASS-DATA NUM TYPE I.  
ENDCLASS.  
  
CLASS TEST\_CLASS IMPLEMENTATION.  
  METHOD TEST\_METHOD.  
    NUM = NUM + 5.  
    WRITE: /5 NUM.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA LO\_OBJECT TYPE REF TO TEST\_CLASS.  
  DATA LT\_OBJECT\_TABLE TYPE TABLE OF REF TO TEST\_CLASS.  
  
  DO 5 TIMES.  
    CREATE OBJECT LO\_OBJECT.  
    APPEND LO\_OBJECT TO LT\_OBJECT\_TABLE.  
  ENDDO.  
  
  LOOP AT LT\_OBJECT\_TABLE INTO LO\_OBJECT.  
    CALL METHOD LO\_OBJECT->TEST\_METHOD.  
  ENDLOOP.

Output:



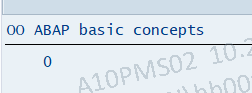
2.13.REPORT ZYP\_OO\_ABAP\_BASIC.  
  
DATA : F(6)  TYPE C,  
       G(10) TYPE C.  
*" METHODS AND CLASS PASS A DYNAMIC VALUE*  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    CLASS-METHODS : STATM .  
    METHODS : INSTM .  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD : STATM .  
    WRITE:/5 'I 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).

Output:



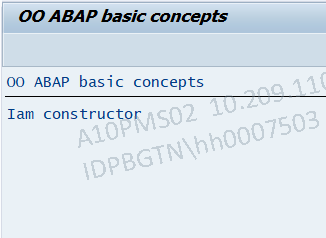
2.15.CLASS CL\_ABAP\_OBJECTDESCR DEFINITION LOAD.    *" TO LOAD THE CLASS OBJECTS*  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS M1 EXCEPTIONS EXC.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    RAISE EXC.  
  ENDMETHOD.  
ENDCLASS.  
  
DATA C TYPE REF TO OBJECT.  
DATA M(3) TYPE C VALUE 'M1'.  
DATA: LT\_TAB TYPE ABAP\_EXCPBIND\_TAB,     *" EXCEPTION TABLE*  
      LS\_TAB LIKE LINE OF LT\_TAB.  
  
START-OF-SELECTION.  
  LS\_TAB-NAME  = 'EXC'.  
  LS\_TAB-VALUE = '0'.  
  INSERT LS\_TAB INTO TABLE LT\_TAB.  
  
  IF SY-SUBRC <> 0.  
    EXIT.  
  ENDIF.  
  
  CREATE OBJECT C TYPE C1.  
    CALL METHOD C->(M) EXCEPTION-TABLE LT\_TAB.  
    WRITE SY-SUBRC.

Output:



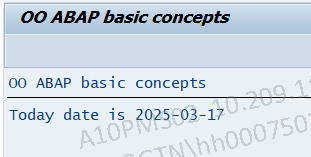
3.1. REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS CLASS\_A DEFINITION.  *"THIS CONSTRUCTOR TRIGGER WITHOUT METHOD CALL*  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_A*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_A IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    WRITE:'IAM CONSTRUCTOR'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
DATA LO\_CLASS\_A TYPE REF TO CLASS\_A.  
CREATE OBJECT LO\_CLASS\_A.

Output:



3.2. CLASS CLASS\_A DEFINITION.   *"CONSTRUCTOR USING IMPORT*  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR IMPORTING DATE TYPE D.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_A*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_A IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    WRITE: 'TODAY DATE IS', DATE DD/MM/YYYY.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
DATA: LO\_CLASS\_A TYPE REF TO CLASS\_A.  
CREATE OBJECT LO\_CLASS\_A EXPORTING DATE = SY-DATUM.

Output:



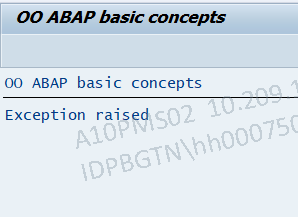
3.3.   
CLASS CLASS\_A DEFINITION.  
  PUBLIC SECTION.  
    METHODS METHOD\_A EXPORTING NAME TYPE C.  
ENDCLASS.

Output:

Compile error.

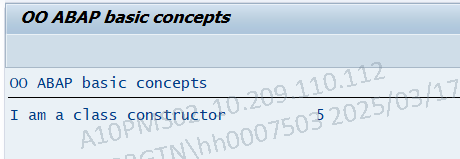
3.4. CLASS CLASS\_A DEFINITION.  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR IMPORTING NUM TYPE I EXCEPTIONS E1.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CLASS\_A*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CLASS\_A IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    IF NUM LT 7.  
      RAISE E1.  
    ENDIF.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  
  DATA: LO\_CLASS\_A TYPE REF TO CLASS\_A.  
  CREATE OBJECT: LO\_CLASS\_A EXPORTING NUM = 5 EXCEPTIONS E1 = 2.  
  IF SY-SUBRC = 2.  
    WRITE 'EXCEPTION RAISED'.  
  ENDIF.

Output:



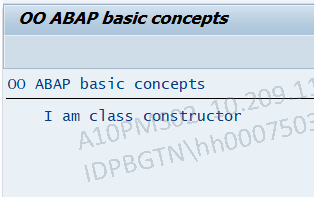
3.5.1 CLASS CLASS\_A DEFINITION. *"CONSTRUCTOR IS TRIGERED WHEN STATIC ATTRIBUTE IS ACCESS*  
  PUBLIC SECTION.  
    CLASS-DATA NUM TYPE I VALUE 5.  
    CLASS-METHODS CLASS\_CONSTRUCTOR.  
ENDCLASS.  
  
CLASS CLASS\_A IMPLEMENTATION.  
  METHOD CLASS\_CONSTRUCTOR.  
    WRITE 'I AM A CLASS CONSTRUCTOR'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  WRITE: CLASS\_A=>NUM.

Output:



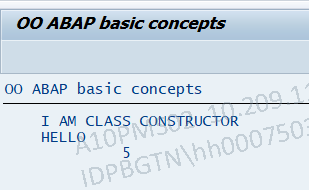
3.5.2 CLASS C1 DEFINITION. *"CONSTRUCTOR IS FIRED WHEN AN OBJECT IS CREATED FROM CLASS*  
  PUBLIC SECTION.  
    CLASS-DATA: NUM TYPE I VALUE 5.  
    CLASS-METHODS: CLASS\_CONSTRUCTOR.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD CLASS\_CONSTRUCTOR.  
    WRITE:/5 'I AM CLASS CONSTRUCTOR'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: OREF TYPE REF TO C1.  
  CREATE OBJECT OREF.

Output:



3.6. CLASS C1 DEFINITION.*" STATIC CONSTRUCTOR CAN BE TRIGGERED AT THE BEGINNING OF A PROCESSING BLOCK*  
  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.

Output:



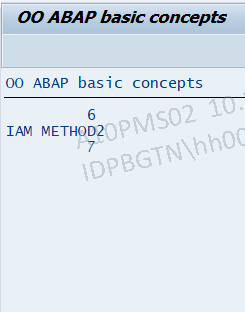
3.7. CLASS C1 DEFINITION. *"STATIC/CLASS CONSTRUCTORS CANNOT HAVE ANY INTERFACE*  
 PUBLIC SECTION.  
 CLASS-METHODS:CLASS\_CONSTRUCTOR IMPORTING NUM TYPE C.  
 ENDCLASS.

Output:

Static constructors of a class cannot have any interface parameters and exceptions.

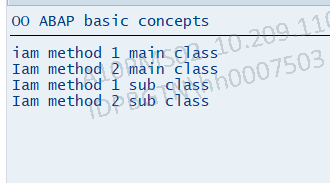
4.1. REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS MAIN\_CLASS DEFINITION.  
  PUBLIC SECTION.  
    DATA NUM TYPE I VALUE 6.  
    METHODS MC1.  
  PROTECTED SECTION.  
    DATA NUM2 TYPE I VALUE 7.  
    METHODS MC2.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) MAIN\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS MAIN\_CLASS IMPLEMENTATION.  
  METHOD MC1.  
    WRITE: / NUM.  
  ENDMETHOD.  
  
  METHOD MC2.  
    WRITE: / 'IAM METHOD2'.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS SUB\_CLASS DEFINITION INHERITING FROM MAIN\_CLASS.  
  PUBLIC SECTION.  
    METHODS SC1.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) SUB\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS SUB\_CLASS IMPLEMENTATION.  
  METHOD SC1.  
    CALL METHOD: MC1, MC2.  
    WRITE: / NUM2.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_CLASS\_B TYPE REF TO SUB\_CLASS.  
  CREATE OBJECT LO\_CLASS\_B.  
  CALL METHOD LO\_CLASS\_B->SC1.

Output:



4.2. REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS MAIN\_CLASS DEFINITION.  
  PUBLIC SECTION.  
    METHODS MC\_PU.  
  PROTECTED SECTION.  
    METHODS MC\_PR.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) MAIN\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS MAIN\_CLASS IMPLEMENTATION.  
  METHOD MC\_PU.  
    WRITE: /'IAM METHOD 1 MAIN CLASS'.  
    CALL METHOD MC\_PR.  
  ENDMETHOD.  
  
  METHOD MC\_PR.  
    WRITE: /'IAM METHOD 2 MAIN CLASS'.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS SUB\_CLASS DEFINITION INHERITING FROM MAIN\_CLASS.  
  PUBLIC SECTION.  
    METHODS MC\_PU REDEFINITION.  
  PROTECTED SECTION.  
    METHODS MC\_PR REDEFINITION.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) SUB\_CLASS*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS SUB\_CLASS IMPLEMENTATION.  
  METHOD MC\_PU.  
    WRITE: /'IAM METHOD 1 SUB CLASS'.  
    CALL METHOD MC\_PR.  
  ENDMETHOD.  
  METHOD MC\_PR.  
    WRITE: /'IAM METHOD 2 SUB CLASS'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_MAIN\_CLASS TYPE REF TO MAIN\_CLASS,  
        LO\_SUB\_CLASS  TYPE REF TO SUB\_CLASS.  
  
  CREATE OBJECT: LO\_MAIN\_CLASS, LO\_SUB\_CLASS.  
  
  CALL METHOD LO\_MAIN\_CLASS->MC\_PU.  
  CALL METHOD LO\_SUB\_CLASS->MC\_PU.

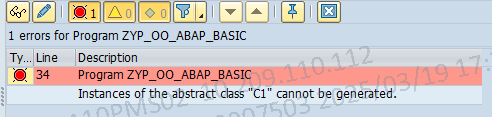
Output:



4.3.1 CLASS C1 DEFINITION ABSTRACT. *"Abstract class can't create object*

PUBLIC SECTION.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
  PUBLIC SECTION.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C2*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_CLASS\_A TYPE REF TO C1,  
        LO\_CLASS\_B TYPE REF TO C2.  
  
  CREATE OBJECT LO\_CLASS\_A.

Output:



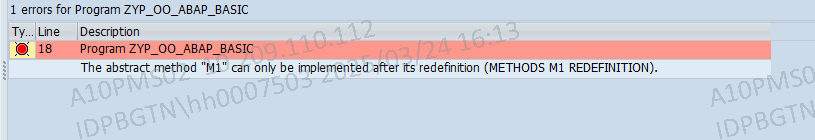
4.3.2 CLASS C1 DEFINITION ABSTRACT. *" DERIVED ABSTRACT CLASS CREATE OBJECT*  
  PUBLIC SECTION.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
  PUBLIC SECTION.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C2*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_CLASS\_A TYPE REF TO C1,  
        LO\_CLASS\_B TYPE REF TO C2.  
  
  CREATE OBJECT LO\_CLASS\_B.

Output:

Object created no error.

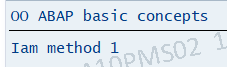
4.4.1 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION ABSTRACT. *" Abstract method can't be implement in class*  
  PUBLIC SECTION.  
    METHODS M1 ABSTRACT.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION. *" Only implement in sub class*  
 METHOD M1.  
    WRITE: 'IAM METHOD 1'.  
 ENDMETHOD.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
*" Implement in sub class need to redefine the sub class*  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
   
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_CLASS\_A TYPE REF TO C1.  
  CREATE OBJECT LO\_CLASS\_A.

Output:



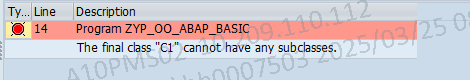
4.4.2 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION ABSTRACT. *" ABSTRACT METHORD CAN'T BE IMPLEMENT IN CLASS*  
  PUBLIC SECTION.  
    METHODS M1 ABSTRACT.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION. *" ONLY IMPLEMENT IN SUB CLASS*  
  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
*" IMPLEMENT IN SUB CLASS NEED TO REDEFINE THE SUB CLASS*  
  PUBLIC SECTION.  
  METHODS M1 REDEFINITION.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
 METHOD M1.  
    WRITE: 'IAM METHOD 1'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_CLASS\_B TYPE REF TO C2.  
  CREATE OBJECT LO\_CLASS\_B.  
  CALL METHOD LO\_CLASS\_B->M1.

Output:



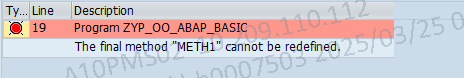
4.5. CLASS C1 DEFINITION FINAL. *" SUB CLASS CANNOT BE INHERIT FROM FINAL CLASS*  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_CLASS\_A TYPE REF TO C1.  
  CREATE OBJECT LO\_CLASS\_A.

Output:



4.6. CLASS C1 DEFINITION. *"FINAL METHOD IN CLASS CAN ONLY BEEN DEFINED IN THAT CLASS*  
  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. *"IT CANNOT BE REDEFINE IN ANY OF ITS SUB CLASS*  
  PUBLIC SECTION.  
    METHODS: METH1 REDEFINITION.  
ENDCLASS.  
CLASS C2 IMPLEMENTATION.  
  METHOD: METH1.  
    WRITE:/5 ' I AM METH1, MODIFIED IN CLASS C2'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: OREF2 TYPE REF TO C2.  
  CREATE OBJECT OREF2.

Output:



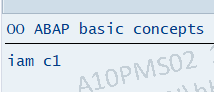
4.7. REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION.   
  PUBLIC SECTION.  
    CLASS-DATA: NUM TYPE I. *" STATIC ATTRIBUTES ONLY EXIST ONCE IN EACH INHERITANCE TREE*  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
CLASS C3 DEFINITION INHERITING FROM C1.  
ENDCLASS.  
  
START-OF-SELECTION.  
*" ONE CAN CHANGE THEM THE OUTSIDE THE CLASS USING CLASS COMPONENT SECTOR WITH ANY CLASS NAME OR*  
*" WITHIN ANY CLASS IN WHICH THEY ARE SHARED*  
  C3=>NUM = 10.  
  WRITE: C2=>NUM.

Output:



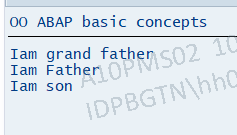
4.8. REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION. *”CONSTRUCTOR OF SUPER CLASS INHERITED BY SUB CLASS ALSO.*  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    WRITE 'IAM C1'.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_CLASS\_B TYPE REF TO C2.  
  CREATE OBJECT LO\_CLASS\_B.

Output:



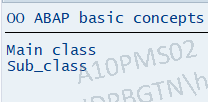
4.9 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS GRAND\_FATHER DEFINITION.  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR.  
ENDCLASS.  
  
CLASS GRAND\_FATHER IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    WRITE: / 'IAM GRAND FATHER'.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS FATHER DEFINITION INHERITING FROM GRAND\_FATHER.  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR.  
ENDCLASS.  
  
CLASS FATHER IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    *”INSTANCE CONSTRUCTOR MENTHODS OF CHILD CLASS ONE FOR SUPER CLASS SHOULD BE CALLED FIRST*  
    *" SUPER->CONSTRUCTOR*  
    CALL METHOD SUPER->CONSTRUCTOR.  
    WRITE: / 'IAM FATHER'.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS SON DEFINITION INHERITING FROM FATHER.  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR.  
ENDCLASS.  
  
CLASS SON IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    CALL METHOD SUPER->CONSTRUCTOR. *”THIS MUST BE DECLERE*  
    WRITE: / 'IAM SON'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_SON TYPE REF TO SON. *”SUB CLASS CAN MODIFY THE CONSTRUCTOR METHOD*  
  CREATE OBJECT LO\_SON.

Output:



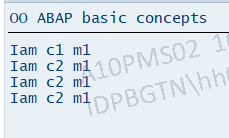
4.10 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS FATHER DEFINITION.  
  PUBLIC SECTION.  
    CLASS-METHODS CLASS\_CONSTRUCTOR. *" STATIC CONSTRUCTOR CALLED ONCE PER PROGRAM*  
ENDCLASS.  
  
CLASS FATHER IMPLEMENTATION.  
  METHOD CLASS\_CONSTRUCTOR.  
    WRITE: / 'MAIN CLASS'.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS SON DEFINITION INHERITING FROM FATHER.  
  PUBLIC SECTION.  
    CLASS-METHODS CLASS\_CONSTRUCTOR.  
ENDCLASS.  
  
CLASS SON IMPLEMENTATION.  
  METHOD CLASS\_CONSTRUCTOR.  
    WRITE: / 'SUB\_CLASS'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA LO\_SON TYPE REF TO SON.  
  CREATE OBJECT LO\_SON. *" IF SYSTEM FIRST ADDRESS A SUBCLASS SYSTEM LOOK FOR NEXT HIGHEST SUPER CLASS*  
  *" WHOSE STATIC CONSTRUCOR WAS NOT BE EXECUTED*  
  DATA: LO\_FATHER TYPE REF TO FATHER.  
  CREATE OBJECT LO\_FATHER.

Output:



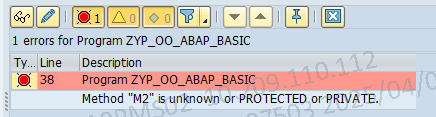
4.11 REPORT ZYP\_OO\_ABAP\_BASIC.   
  
CLASS C1 DEFINITION. *" C1 IS STATIC*  
  PUBLIC SECTION.  
    METHODS M1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    WRITE: / 'IAM C1 M1'.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1. *" C2 IS DYNAMIC*  
  PUBLIC SECTION.  
    METHODS M1 REDEFINITION.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION .  
  METHOD M1.  
    WRITE: / 'IAM C2 M1'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_C1\_1 TYPE REF TO C1,  
        LO\_C1\_2 TYPE REF TO C1,  
        LO\_C1\_3 TYPE REF TO C1,  
        LO\_C2\_1 TYPE REF TO C2.  
  
  CREATE OBJECT: LO\_C1\_1,  
                 LO\_C1\_2 TYPE C2,  
                 LO\_C1\_3,  
                 LO\_C2\_1.  
  LO\_C1\_3 = LO\_C2\_1.  
  
  CALL METHOD: LO\_C1\_1->M1, *" BOTH STSTIC AND DYNAMIC TYPE OF LO\_C1\_1 REFER C1.*  
               LO\_C1\_2->M1, *" STATIC TYPE OF LO\_C1\_2 REFER C1 DYNAMIC TYPER REFER C2.*  
               LO\_C1\_3->M1, *" STATIC TYPE OF LO\_C1\_3 REFER C1 DYNAMIC TYPER REFER C2.*  
               LO\_C2\_1->M1. *" BOTH STSTIC & DYNAMIC TYPE LO\_C2\_1 REFER TO C2*

Output:



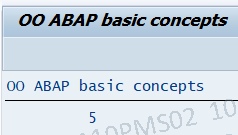
4.12. REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION. *"Reference variable will identify all the common component of super & sub class.*  
  PUBLIC SECTION.  
    METHODS M1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    WRITE: 'IAM C1 M1'. *"Static type of reference variable can refer super class*  
  ENDMETHOD.   
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
  PUBLIC SECTION. *" It will not able to identify any new component of sub class which are not present in super class*  
    METHODS M1 REDEFINITION.  
    METHODS M2.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.   
  METHOD M1.  
    WRITE: 'IAM C2 M1'.  
  ENDMETHOD.  
  
  METHOD M2.  
    WRITE: 'IAM M2'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_C1 TYPE REF TO C1.  
  CREATE OBJECT LO\_C1 TYPE C2.  
  CALL METHOD LO\_C1->M2.

Output:



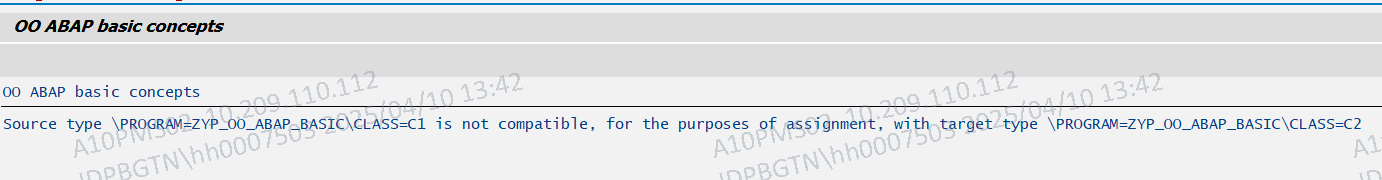
4.13 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS M1.  
  PRIVATE SECTION.  
    DATA NUM TYPE I VALUE 5.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    WRITE NUM.  *" USING PRIVATE ATTRIBUTE INHERIT FROM SUPER CLASS NOT SUB CLASS*  
     *" EVEN IF SUB CLASS HAVE PRIVATE ATTRIBUTE SAME NAME*  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS C2 DEFINITION INHERITING FROM C1.  
  PRIVATE SECTION.  
    DATA NUM TYPE I VALUE 8.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_NUM TYPE REF TO C2.  
  CREATE OBJECT LO\_NUM.  
  CALL METHOD LO\_NUM->M1.

Output:



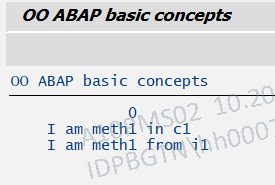
4.14 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    DATA : NUM TYPE I VALUE 5.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
ENDCLASS.  
CLASS C2 DEFINITION INHERITING FROM C1.  
  PUBLIC SECTION.  
ENDCLASS.  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
START-OF-SELECTION .  
  DATA : OBJ1 TYPE REF TO C1,  
         OBJ2 TYPE REF TO C2.  
  DATA: LO\_ROOT TYPE REF TO CX\_ROOT,  
        LT\_TEXT TYPE STRING.  
  CREATE OBJECT : OBJ1 ,  
  OBJ2 .  
  TRY.  
*" WIDING CASTE OPERATOR CORRECT METHOD*  
      OBJ1 ?= OBJ2.  
      BREAK-POINT.  
*" WIDING CASTE OPERATOR*  
*\*      OBJ2 ?= OBJ1.*  
*"WAY - 1*  
*\*    CATCH CX\_SY\_MOVE\_CAST\_ERROR.*  
*\*      WRITE:/5 'CX\_SY\_MOVE\_CAST\_ERROR TRAPPED'.*  
    *" OR WANT TO GET A ABAP DUMP TEXT.*  
*"WAY - 2*  
    CATCH CX\_SY\_MOVE\_CAST\_ERROR INTO LO\_ROOT.  
      LT\_TEXT = LO\_ROOT->GET\_TEXT( ).  
      WRITE LT\_TEXT.  
  ENDTRY.

Output:



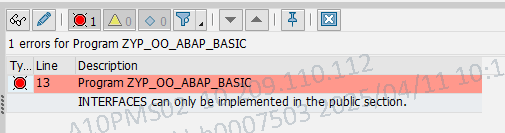
5.1 INTERFACE I1.  
  DATA: NUM TYPE I.  
  METHODS : METH1.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS: METH1.  
*\*    “ CLASS C1’S OWN METHOD*  
INTERFACES : I1.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD: METH1.  
    WRITE:/5 'I AM METH1 IN C1'.  
  ENDMETHOD.  
  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.

Output:



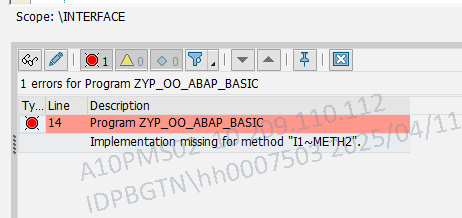
5.2 INTERFACE I1.  
  METHODS : METH1.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PROTECTED SECTION.  
    INTERFACES : I1.  
ENDCLASS.

Output:



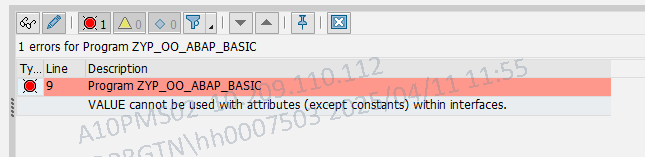
5.3 INTERFACE I1.  
  METHODS : METH1 ,  
    METH2 .  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES : I1.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD I1~METH1.  
    WRITE:/5 'I AM METH1 FROM I1'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA : OREF TYPE REF TO C1.  
  CREATE OBJECT OREF.  
  CALL METHOD OREF->I1~METH1.

Output:



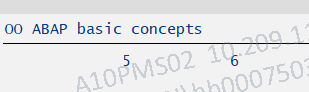
5.4.1   
INTERFACE I1 .  
  DATA : NUM1 TYPE I VALUE 5,  
         NUM2 TYPE I VALUE 6.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES : I1 .  
    METHODS M1.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    WRITE:/5 I1~NUM1,  
     I1~NUM2.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA : OREF TYPE REF TO C1.  
  CREATE OBJECT OREF.  
  CALL METHOD OREF->M1.

Output:



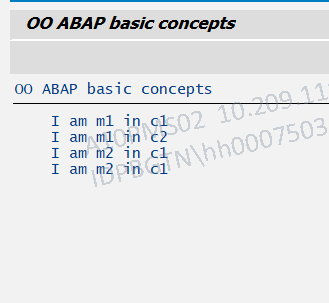
5.4.2   
INTERFACE I1 .  
  DATA : NUM1 TYPE I,  
         NUM2 TYPE I.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES : I1 DATA VALUES NUM1 = 5 NUM2 = 6 .  
    METHODS M1.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    WRITE:/5 I1~NUM1,  
    I1~NUM2.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA : OREF TYPE REF TO C1.  
  CREATE OBJECT OREF.  
  CALL METHOD OREF->M1.

Output:



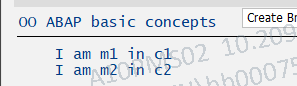
5.5 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
INTERFACE I1.  
  METHODS : M1,  
    M2 .  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES: I1 FINAL METHODS M2.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD I1~M1.  
    WRITE:/5 'I AM M1 IN C1'.  
  ENDMETHOD.  
  METHOD I1~M2.  
    WRITE:/5 'I AM M2 IN C1'.  
  ENDMETHOD.  
ENDCLASS.  
CLASS C2 DEFINITION INHERITING FROM C1.  
  PUBLIC SECTION.  
    METHODS : I1~M1 REDEFINITION .  
ENDCLASS.  
CLASS C2 IMPLEMENTATION.  
  METHOD : I1~M1.  
    WRITE:/5 'I AM M1 IN C2'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  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 .               *"P OUTPUT : I AM M2 IN C1*

Output:



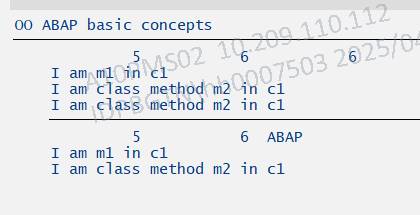
5.6 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
INTERFACE I1.  
  METHODS: M1 ,M2 .  
ENDINTERFACE.  
CLASS C1 DEFINITION ABSTRACT.  
  PUBLIC SECTION.  
    INTERFACES: I1 ABSTRACT METHODS M2.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD I1~M1.  
    WRITE:/5 'I AM M1 IN C1'.  
  ENDMETHOD.  
ENDCLASS.  
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.

Output:



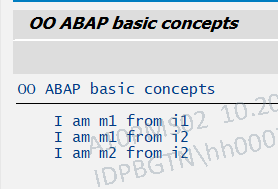
5.7 INTERFACE I1.  
  CONSTANTS: C\_NAME(4) TYPE C VALUE 'ABAP'.  
  DATA: INUM TYPE I.  
  CLASS-DATA: CNUM TYPE I.  
  METHODS: M1.  
  CLASS-METHODS: M2.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES: I1 DATA VALUES INUM = 5 CNUM = 6.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD I1~M1.  
    WRITE:/5 'I AM M1 IN C1'.  
  ENDMETHOD.  
  METHOD I1~M2.  
    WRITE:/5 'I AM CLASS METHOD M2 IN C1'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: LO\_I1 TYPE REF TO I1,  
        LO\_C1 TYPE REF TO C1.  
  CREATE OBJECT: LO\_C1.  
  WRITE:/5 LO\_C1->I1~INUM,  
  LO\_C1->I1~CNUM,  
  C1=>I1~CNUM.  
  CALL METHOD: LO\_C1->I1~M1,  
  LO\_C1->I1~M2,  
  C1=>I1~M2 .  
  WRITE:/5 SY-ULINE.  
  LO\_I1 = LO\_C1.  
  WRITE:/5 LO\_I1->INUM,  
  LO\_I1->CNUM,  
  I1=>C\_NAME.  
  CALL METHOD: LO\_I1->M1,  
  LO\_I1->M2.

Output



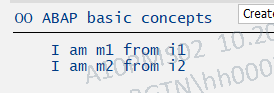
5.8 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
INTERFACE I1.  
  METHODS M1.  
ENDINTERFACE.  
INTERFACE I2.  
  METHODS: M1, M2.  
  INTERFACES I1.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES: I2.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD: I1~M1.  
    WRITE:/5 'I AM M1 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->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*

Output



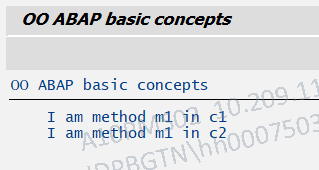
5.9 INTERFACE I1.  
  METHODS M1.  
ENDINTERFACE.  
INTERFACE I2.  
  METHODS: M1, M2 .  
  INTERFACES I1.  
  ALIASES METH1 FOR I1~M1.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES: I2.  
    ALIASES METH2 FOR I2~M2.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD I1~M1.  
    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.

Output



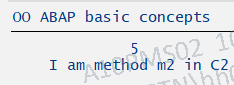
5.10 INTERFACE I1.  
  METHODS : M1.  
ENDINTERFACE.  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES : I1.  
ENDCLASS.  
CLASS C1 IMPLEMENTATION.  
  METHOD I1~M1.  
    WRITE:/5 'I AM METHOD M1 IN C1'.  
  ENDMETHOD.  
ENDCLASS.  
CLASS C2 DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES : 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.

Output



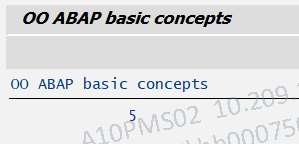
6.1 CLASS C1 DEFINITION DEFERRED.                  *" C2 FRIENDS PRIVATE ALL ACCESS IN C1*  
CLASS C2 DEFINITION CREATE PRIVATE FRIENDS C1. *" THIS MAENS CLASS ITSELF AND FRIENDS CAN INSTANTIATE THIS CLASS*  
  PROTECTED SECTION.  
    DATA : NUM TYPE I VALUE 5.  
    METHODS : M2.  
ENDCLASS.  
CLASS 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.

Output



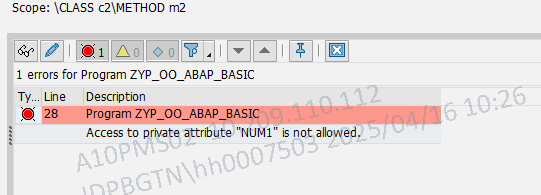
6.2 CLASS C1 DEFINITION DEFERRED.  
  
CLASS C2 DEFINITION FRIENDS C1.  
  PROTECTED SECTION.  
    DATA NUM TYPE I VALUE 5.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
ENDCLASS.  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS M1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
METHOD M1.  
  DATA OREF2 TYPE REF TO C2.  
  CREATE OBJECT OREF2.  
  WRITE: /5 OREF2->NUM.  
ENDMETHOD.  
ENDCLASS.  
  
CLASS C11 DEFINITION INHERITING FROM C1. *" SUBCLASS ALSO FRIENDS WITH SUPER CLASS FRIEND*  
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.

Output



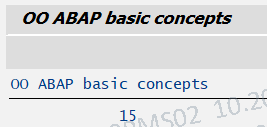
6.3.1 CLASS C1 DEFINITION DEFERRED.  
  
CLASS C2 DEFINITION FRIENDS C1.  
  PROTECTED SECTION.  
    DATA NUM2 TYPE I VALUE 15.  
    METHODS M2.  
ENDCLASS.  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS METHOD\_PUBLIC.  
  PRIVATE SECTION.  
    DATA NUM1 TYPE I VALUE 10.  
    METHODS 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 METHOD\_PUBLIC.  
    CALL METHOD M1.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA OREF TYPE REF TO C1.  
  CREATE OBJECT OREF.  
  CALL METHOD OREF->METHOD\_PUBLIC.

Output



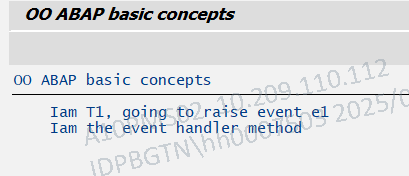
6.3.2 CLASS C1 DEFINITION DEFERRED.  
  
CLASS C2 DEFINITION FRIENDS C1.  
  PROTECTED SECTION.  
    DATA NUM2 TYPE I VALUE 15.  
    METHODS M2.  
ENDCLASS.  
  
CLASS C1 DEFINITION FRIENDS C2.  
  PUBLIC SECTION.  
    METHODS METHOD\_PUBLIC.  
  PRIVATE SECTION.  
    DATA NUM1 TYPE I VALUE 10.  
    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 METHOD\_PUBLIC.  
    CALL METHOD M1.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA OREF TYPE REF TO C1.  
  CREATE OBJECT OREF.  
  CALL METHOD OREF->METHOD\_PUBLIC.

Output



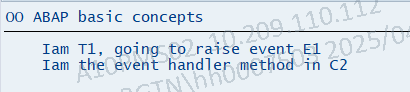
7.1 CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    EVENTS: E1.  *" CREATING EVENT E1.*  
    *" CREATING AN EVENT HANDLING METHOD. THIS METHOD CAN BELONG TO SAME OR DIFFERENT CLASS.*  
    METHODS M1 FOR EVENT E1 OF C1. *" METHOD TO RAISE THE EVENT*  
    METHODS T1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  *" METHOD M1 WILL BE CALLED WHEN THE EVENT IS RAISED*  
  METHOD M1.  
    WRITE: /5 'IAM THE EVENT HANDLER METHOD'.  
  ENDMETHOD.  
  
  *" METHOD T1 WILL RAISE THE EVENT*  
  METHOD T1.  
    WRITE: /5 'IAM 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.

Output



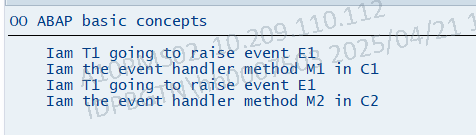
7.2 CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    *"CREATING EVENT E1.*  
    EVENTS E1.  
    *"TRIGGERING METHOD T1.*  
    METHODS T1.  
ENDCLASS.  
  
CLASS C2 DEFINITION.  
  PUBLIC SECTION.  
    *"CREATE AN EVENT HANDLING METHOD*  
    METHODS: M1 FOR EVENT E1 OF C1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  *"METHOD T1 WILL RAISE THE EVENT*  
  METHOD T1.  
    WRITE: /5 'IAM T1, GOING TO RAISE EVENT E1'.  
    RAISE EVENT E1.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS C2 IMPLEMENTATION.  
  *"METHOD M1 WILL BE CALLED WHEN EVENT IS RAISED.*  
  METHOD M1.  
    WRITE:/5 'IAM THE EVENT HANDLER METHOD IN C2'.  
  ENDMETHOD.  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: O\_REF1 TYPE REF TO C1,  
        O\_REF2 TYPE REF TO C2.  
  
  CREATE OBJECT: O\_REF1, O\_REF2.  
  
  *"REGISTERING THE EVENT HANDLER METHOD*  
  SET HANDLER O\_REF2->M1 FOR O\_REF1.  
  
  *"CALLING THE EVENT WHICH WILL RAISE THE EVENT*  
  CALL METHOD O\_REF1->T1.

Output:



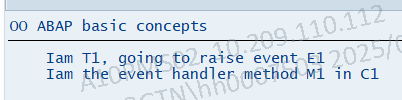
7.3 CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    *"CREATING EVENT E1*  
    EVENTS E1.  
    *"CREATING AN EVENT HANDLING METHOD*  
    METHODS M1 FOR EVENT E1 OF C1.  
    *" METHOD TO RAISE THE EVENT*  
    METHODS T1.  
ENDCLASS.  
  
CLASS C2 DEFINITION.  
  PUBLIC SECTION.  
    *" CREATING AN EVENT HANDLING METHOD*  
    METHODS: M2 FOR EVENT E1 OF C1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  *"METHOD T1 WILL RAISE THE EVENT*  
  METHOD T1.  
    WRITE: /5 'IAM T1 GOING TO RAISE EVENT E1'.  
    RAISE EVENT E1.  
  ENDMETHOD.  
  *"METHOD M1 WILL BE CALLED WHEN THE EVENT IS RAISED*  
  METHOD M1.  
    WRITE: /5 'IAM 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 'IAM THE EVENT HANDLER METHOD M2 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 OREF1->M1 FOR OREF1.  
  
  *"CALLING THE EVENT WHICH WILL RAISE THE EVENT*  
  CALL METHOD OREF1->T1.  
  
  *"DE-REGEISTERING 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.

Output



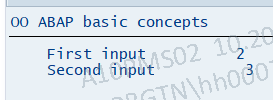
7.4 CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    *" CREATING EVENT E1*  
    CLASS-EVENTS: E1.  
    *" CREATING AN EVENT HANDLING METHOD*  
    METHODS M1 FOR EVENT E1 OF C1.  
    *" METHOD TO RAISE THE EVENT*  
    CLASS-METHODS: T1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
  *" METHOD: T1 WILL RAISE THE EVENT*  
  METHOD: T1.  
    WRITE:/5 'IAM T1, GOING TO RAISE EVENT E1'.  
    RAISE EVENT E1.  
  ENDMETHOD.  
  
  *" METHOD: M1 WILL BE CALLED WHEN EVENT IS RAISED*  
  METHOD M1.  
    WRITE: /5 'IAM THE EVENT HANDLER METHOD M1 IN C1'.  
  ENDMETHOD.  
  
ENDCLASS.  
  
START-OF-SELECTION.  
  DATA: O\_REF1 TYPE REF TO C1.  
  CREATE OBJECT O\_REF1.  
  *" REGISTERING THE EVENT HANDLER METHOD.*  
  SET HANDLER O\_REF1->M1.  
  *" CALLING THE EVENT WHICH WILL RAISE THE EVENT*  
  CALL METHOD O\_REF1->T1.

Output:



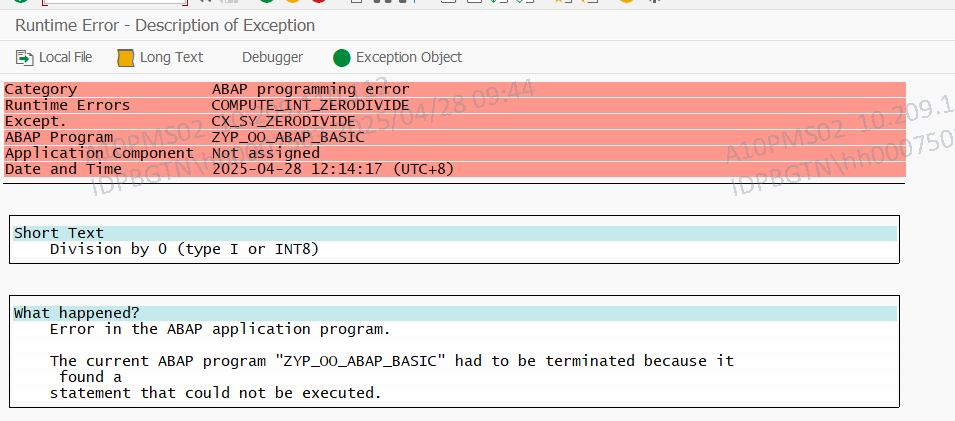
7.5 CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    EVENTS: E1 EXPORTING VALUE(NUM1) TYPE I  
                         VALUE(NUM2) TYPE I.  
  
    METHODS: M1 FOR EVENT E1 OF C1 IMPORTING NUM1 NUM2.  
    METHODS: T1.  
ENDCLASS.  
  
CLASS C1 IMPLEMENTATION.  
METHOD: M1.  
WRITE: /5 'FIRST INPUT', NUM1,  
       /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.

Output:



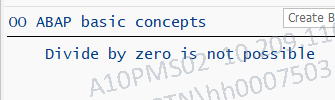
8.1 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
*" ABAP DUMP GET BY DIVIDE ZERO ERROR*  
DATA: LV\_I TYPE I VALUE 1.  
START-OF-SELECTION.  
LV\_I = LV\_I / 0.

Output:

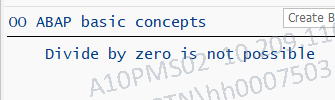


*" HOW TO AVAOD ABAP DUMP*  
DATA: LV\_I TYPE I VALUE 1.  
  
START-OF-SELECTION.  
  CATCH SYSTEM-EXCEPTIONS COMPUTE\_INT\_ZERODIVIDE = 2.  
    LV\_I = LV\_I / 0.  
  ENDCATCH.  
  IF SY-SUBRC = 2.  
    WRITE: /5 'DIVIDE BY ZERO IS NOT POSSIBLE'.  
  ENDIF.

Output:

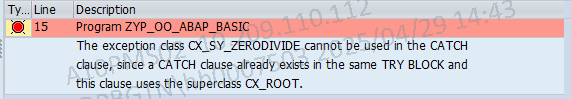
  
  
  *" OR*  
  
  DATA: LV\_I TYPE I VALUE 1.  
  
START-OF-SELECTION.  
  TRY .  
      LV\_I = LV\_I / 0.  
    CATCH CX\_SY\_ZERODIVIDE.  
      WRITE: /5 'DIVIDE BY ZERO IS NOT POSSIBLE'.  
  ENDTRY.

Output:



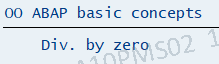
8.2 DATA LV\_DIV TYPE I VALUE 1.  
  
START-OF-SELECTION.  
  TRY .  
      LV\_DIV = LV\_DIV / 0.  
    CATCH CX\_ROOT.  
      WRITE: /5 'ERROR TRAPPED'.  
    CATCH CX\_SY\_ZERODIVIDE.  
      WRITE: /5 'DIV. BY ZERO'.  
  ENDTRY.

Output



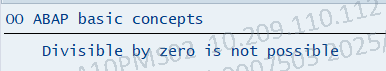
DATA LV\_DIV TYPE I VALUE 1.  
  
START-OF-SELECTION.  
  TRY .  
      LV\_DIV = LV\_DIV / 0.  
    CATCH CX\_SY\_ZERODIVIDE.  
      WRITE: /5 'DIV. BY ZERO'.  
    CATCH CX\_ROOT.  
      WRITE: /5 'ERROR TRAPPED'.  
  ENDTRY.

Output



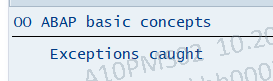
8.3 TRY .  
    PERFORM SUB\_CHECK USING 5.  
  CATCH CX\_SY\_ZERODIVIDE.  
    WRITE: /5 'DIVISIBLE BY ZERO IS NOT POSSIBLE'.  
ENDTRY.  
  
FORM SUB\_CHECK USING A RAISING CX\_SY\_ZERODIVIDE.  
  A = A / 0.  
ENDFORM.

Output



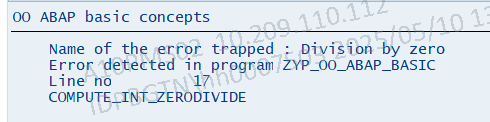
8.4 DATA LV\_INUM TYPE I.  
  
TRY .  
    RAISE EXCEPTION TYPE CX\_SY\_ZERODIVIDE.  
  CATCH CX\_SY\_ZERODIVIDE.  
    WRITE: /5 'EXCEPTIONS CAUGHT'.  
ENDTRY.

Output:



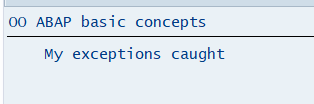
8.5 DATA: LV\_INUM TYPE I VALUE 5,  
      LV\_DES  TYPE STRING,  
      LV\_PNAM TYPE SY-REPID,  
      LV\_LNO  TYPE I,  
      LO\_OBJ  TYPE REF TO CX\_SY\_ZERODIVIDE.  
  
START-OF-SELECTION.  
  
  TRY .  
      LV\_INUM = LV\_INUM / 0.  
    CATCH CX\_SY\_ZERODIVIDE INTO LO\_OBJ.  
      *" UTILIZING METHODS / ATTRIBUTES USING OBJECT OF THE EXCEPTION CLASS*  
      CALL METHOD LO\_OBJ->GET\_TEXT  
        RECEIVING  
          RESULT = LV\_DES.  
      WRITE: /5 'NAME OF THE ERROR TRAPPED :', LV\_DES.  
  
      CALL METHOD LO\_OBJ->GET\_SOURCE\_POSITION  
        IMPORTING  
          PROGRAM\_NAME = LV\_PNAM  
          SOURCE\_LINE  = LV\_LNO.  
  
      WRITE: /5 'ERROR DETECTED IN PROGRAM', LV\_PNAM,  
             /5 'LINE NO', LV\_LNO.  
  
      WRITE: /5 LO\_OBJ->KERNEL\_ERRID.  
  ENDTRY.

Output:



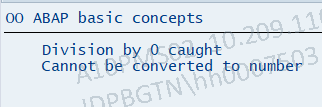
8.6 CLASS CX\_MY\_EXCEPTION DEFINITION INHERITING FROM CX\_STATIC\_CHECK.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) CX\_MY\_EXCEPTION*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS CX\_MY\_EXCEPTION IMPLEMENTATION.  
ENDCLASS.  
  
CLASS C1 DEFINITION.  
  PUBLIC SECTION.  
    METHODS M1 RAISING CX\_MY\_EXCEPTION.  
ENDCLASS.  
*\*&---------------------------------------------------------------------\**  
*\*& CLASS (IMPLEMENTATION) C1*  
*\*&---------------------------------------------------------------------\**  
*\*&*  
*\*&---------------------------------------------------------------------\**  
CLASS C1 IMPLEMENTATION.  
  METHOD M1.  
    RAISE EXCEPTION TYPE CX\_MY\_EXCEPTION.  
  ENDMETHOD.  
ENDCLASS.  
  
DATA: EX    TYPE REF TO CX\_MY\_EXCEPTION,  
      O\_REF TYPE REF TO C1.  
  
START-OF-SELECTION.  
  TRY .  
     CREATE OBJECT O\_REF.  
      O\_REF->M1( ).  
    CATCH CX\_MY\_EXCEPTION INTO EX.  
      WRITE: /5 'MY EXCEPTIONS CAUGHT'.  
  ENDTRY.

Output:



8.7 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.

Output:



8.8 REPORT ZYP\_OO\_ABAP\_BASIC.  
  
START-OF-SELECTION.  
  
  DATA: NUM TYPE I VALUE 5.  
  
  TRY .  
      TRY .  
          NUM = 'ABCHDE'.  
        CLEANUP.  
          WRITE: /5 'IN CLEANUP'.  
        ENDTRY.  
      CATCH CX\_SY\_CONVERSION\_NO\_NUMBER.  
        WRITE: /5 'CANNOT BE CONVERTED TO NUMBER'.  
    ENDTRY.

Output:

