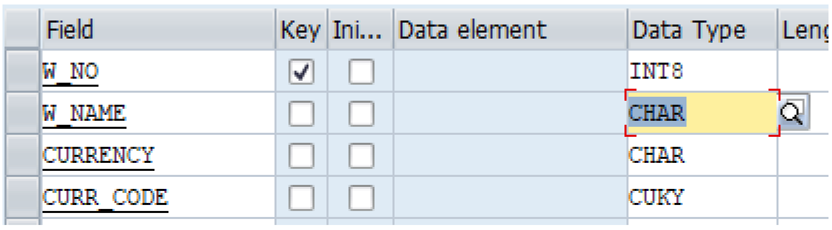
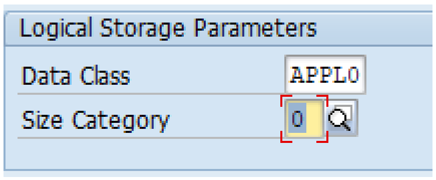
1. A database table with following structure: client, w_no, w_name, currency, curr_code. ✅

SE11
Give name and Create
Set Maintenance to not allowed
Add the fields



Field	Key	Ini...	Data element	Data Type	Leng
W_NO	✓	☐		INT8	
W_NAME	☐	☐		CHAR	
CURRENCY	☐	☐		CHAR	
CURR_CODE	☐	☐		CUKY	

Set parameters in technical settings:



Logical Storage Parameters

Data Class	APPL0
Size Category	0

set enhancement category to annot be enhanced under extras->enhancement category

check and activate.

2. A program containing an internal table and code for populating the internal table. ✅

1. SE80
2. Make program
3. Type code

```
TYPES: BEGIN OF ty_employee,
         employee_id TYPE n,
         first_name  TYPE char10,
         last_name   TYPE char10,
       END OF ty_employee.

DATA: lt_employees TYPE TABLE OF ty_employee,
      ls_employee  TYPE ty_employee.

START-OF-SELECTION.
  ls_employee-employee_id = 1.
  ls_employee-first_name  = 'John'.
  ls_employee-last_name   = 'Doe'.
  APPEND ls_employee TO lt_employees.
  ls_employee-employee_id = 2.
  ls_employee-first_name  = 'Jane'.
  ls_employee-last_name   = 'Smith'.
  APPEND ls_employee TO lt_employees.

WRITE: / 'Employee ID', 'First Name', 'Last Name'.
LOOP AT lt_employees INTO ls_employee.
  WRITE: / ls_employee-employee_id,
          ls_employee-first_name,
          ls_employee-last_name.
ENDLOOP.
```

3. A program containing an internal table and retrieve of data from a database. ✅

1. SE80
2. Make program
3. Type code -> SPFLI is the plane database

```
DATA: BEGIN OF flightPaths OCCURS 0,
         cityFrom TYPE spfli-cityfrom,
         cityTo TYPE spfli-cityto,
       END OF flightPaths.

* Retrieve data from a database table and populate the internal table
SELECT cityFrom, cityTo FROM spfli
  INTO TABLE @flightPaths.

* Display the contents of the internal table
LOOP AT flightPaths.
  WRITE: / flightPaths-cityFrom, flightPaths-cityTo.
ENDLOOP.
```

4. A program containing an internal table and print to screen. ✅

1. SE80
2. Make Program
3. Type Code

```
TYPES: BEGIN OF ty_employee,
         employee_id TYPE n,
         first_name  TYPE char10,
         last_name   TYPE char10,
       END OF ty_employee.

DATA: lt_employees TYPE TABLE OF ty_employee,
      ls_employee  TYPE ty_employee.

START-OF-SELECTION.
  ls_employee-employee_id = 1.
  ls_employee-first_name  = 'John'.
  ls_employee-last_name   = 'Doe'.
  APPEND ls_employee TO lt_employees.

  ls_employee-employee_id = 2.
  ls_employee-first_name  = 'Jane'.
  ls_employee-last_name   = 'Smith'.
  APPEND ls_employee TO lt_employees.

WRITE: / 'Employee ID', 'First Name', 'Last Name'.
LOOP AT lt_employees INTO ls_employee.
  WRITE: / ls_employee-employee_id,
           ls_employee-first_name,
           ls_employee-last_name.
ENDLOOP.
```

5. A function group for geometric figures with a function module for square calculation. ✅

1. SE80
2. make function module.
3. Make function group.

In the function module: (use import and export to define the references.

```
FUNCTION Z_99_SQUAREAREA.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(SIDE) TYPE  I
*"  EXPORTING
*"     REFERENCE(AREA) TYPE  N
*"----------------------------------------------------------------------


area = side * 2.
ENDFUNCTION.
```

Make a program:
In the program:

```
DATA:
      l_area TYPE n.

CALL FUNCTION 'Z_99_SQUAREAREA'
  EXPORTING
    side          = 2
IMPORTING
 AREA          = l_area.

WRITE: l_area.
```

6. A program using function module and code for print to screen of the result. ✅

1. make function module
2. Make function group.

In the function module: (use import and export to define the references.

```
FUNCTION Z_99_SQUAREAREA.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(SIDE) TYPE  I
*"  EXPORTING
*"     REFERENCE(AREA) TYPE  N
*"----------------------------------------------------------------------


area = side * 2.
ENDFUNCTION.
```

3. Make a program:

In the program:

```
DATA:
      l_area TYPE n.

CALL FUNCTION 'Z_99_SQUAREAREA'
  EXPORTING
    side          = 2
IMPORTING
 AREA          = l_area.

WRITE: l_area.
```

7. A program using parameters and print to screen. ✅

1. SE80
2. Make Program
3. Type code

```
TYPES: BEGIN OF ty_employee,
         employee_id TYPE n,
         first_name  TYPE char10,
         last_name   TYPE char10,
       END OF ty_employee.

DATA: lt_employees TYPE TABLE OF ty_employee,
      ls_employee  TYPE ty_employee.

PARAMETERS: p_emp_id TYPE n,
            p_fname  TYPE char10,
            p_lname   TYPE char10.

START-OF-SELECTION.
  ls_employee-employee_id = p_emp_id.
  ls_employee-first_name  = p_fname.
  ls_employee-last_name   = p_lname.
  APPEND ls_employee TO lt_employees.

  LOOP AT lt_employees INTO ls_employee.
    WRITE: / ls_employee-employee_id,
            ls_employee-first_name,
            ls_employee-last_name.
  ENDLOOP.
```

8. A program using modularization ”Include” for calculation of the area of a circle and print the result to screen. ✅

1. SE80
2. Make Program
3. Type Code

Make program to be included:

```
REPORT ZY_99_TOBEINCLUDED.

DATA: area TYPE DECFLOAT16,
      pi TYPE DECFLOAT16.

pi = '3.14'.

PARAMETERS: radius TYPE n.

area = pi * radius * radius.

WRITE: area.
```

4. Make a new program and include the one from before:

```
INCLUDE zy_99_tobeincluded.
```

9. A program using modularization ”Subroutine” for calculation of the area of a circle and print the result to screen. ✅

1. Make a program
2. Right click on the program and add subroutine.
3. Make the code look like this

```
DATA: num1 TYPE i,
      num2 TYPE i,
      sum  TYPE i.

num1 = 2. num2 = 4.
PERFORM addit USING num1 num2 CHANGING sum.

num1 = 7. num2 = 11.
PERFORM addit USING num1 num2 CHANGING sum.

FORM addit
       USING add_num1    TYPE any
             add_num2    TYPE any
       CHANGING add_sum TYPE any.

  add_sum = add_num1 + add_num2.
  WRITE: / 'Sum of', add_num1, 'and', add_num2, 'is', add_sum.

ENDFORM.
```

10. A program using a dynpro screen.

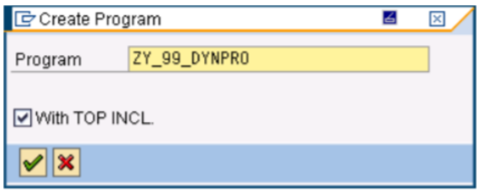
Menu path

Tools • ABAP Workbench • Overview • Object Navigator

You may also use the transaction code **SE80** for direct access.

TOP INCL

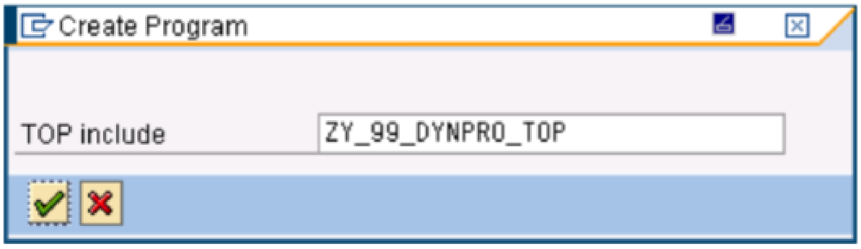
Create a new program called '**ZY_##_DYNPRO**'. Please use '**TOP INCLUDE**'.


Create Program

Program ZY_99_DYNPRO

☑ With TOP INCL.

Modify the name of your top include to '**ZY_##_DYNPRO_TOP**'.

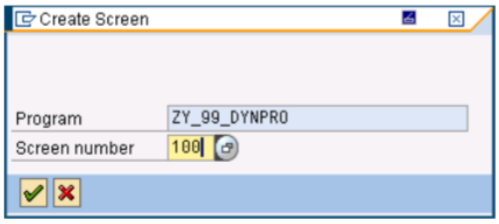

Create Program

TOP include ZY_99_DYNPRO_TOP

Select the status '**T Test Program**'

Create dynpro

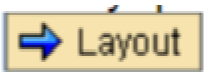
The next step concerns the creation of the first Dynpro. Right click on your program name in the navigation tree and create a new Dynpro (screen) with the screen number 100.


Create Screen

Program ZY_99_DYNPRO

Screen number 100

pressing the button



(MAKE THE SCREEN YOU WANT NOW)

11. A program using an ALV-list. ✅

Make a program -> and write this code.

```
DATA: it_spfli TYPE TABLE OF spfli.

SELECT * FROM spfli INTO TABLE it_spfli.

CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
    i_structure_name = 'SPFLI'
  TABLES
    t_outtab         = it_spfli.
```

12. A program using a global class ✅

zcl_customers already exist so we just use it.

```
DATA: customers_obj TYPE REF TO zcl_customers.

START-OF-SELECTION.

  CREATE OBJECT customers_obj TYPE zcl_customers.

  " Create a new customer
  customers_obj->create_cust(
    p_id = '12345'
    p_name = 'John Smith'
  ).

  " Delete a customer
  customers_obj->delete_cust(
    p_id = '12345'
  ).
```

13. A program using a local class. ✅

Copy paste the definition and the implementation from the global class and add them as a local class. (Dropdown->classes->ZCL_CUSTOMERS->code->copy this.

Change it a bit so everything is public.

```
DATA: customers_obj TYPE REF TO zcl_customers.

START-OF-SELECTION.
```

```
  CREATE OBJECT customers_obj TYPE zcl_customers.

  " Create a new customer
  customers_obj->create_cust(
    p_id = '12345'
    p_name = 'John Smith'
  ).

  " Delete a customer
  customers_obj->delete_cust(
    p_id = '12345'
  ).

*&---------------------------------------------------------------------*
*& Class LCD_ADD
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
CLASS lcd_add DEFINITION.
  public section.
    class-methods CREATE_CUST
      importing
        !P_ID   type ZCUSTOMERS-ID
        !P_NAME type ZCUSTOMERS-F_NAME.
    class-methods DELETE_CUST
      importing !P_ID type ZCUSTOMERS-ID.
    class-data ZCUSTOMERS_WA type ZCUSTOMERS.
ENDCLASS.
*&---------------------------------------------------------------------*
*& Class (Implementation) LCI_add
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
CLASS lcd_add IMPLEMENTATION.

*
<SIGNATURE>-----------------------------------------------------------------------------
-----------------------+
* | Static Public Method ZCL_CUSTOMERS=>CREATE_CUST
*
+-------------------------------------------------------------------------------------------
-----------------------+
* | [--->] P_ID                          TYPE         ZCUSTOMERS-ID
* | [--->] P_NAME                        TYPE         ZCUSTOMERS-F_NAME
*
+-------------------------------------------------------------------------------------------
-----------</SIGNATURE>
```

```
  method CREATE_CUST.

    ZCUSTOMERS_WA-ID = P_ID.
    ZCUSTOMERS_WA-F_NAME = P_NAME.

    insert  ZCUSTOMERS from ZCUSTOMERS_WA .

  endmethod.


*
<SIGNATURE>---------------------------------------------------------------------------------------+
* | Static Public Method ZCL_CUSTOMERS=>DELETE_CUST
*
+-------------------------------------------------------------------------------------------------+
* | [--->] P_ID                           TYPE        ZCUSTOMERS-ID
*
+--------------------------------------------------------------------------------------</SIGNATURE>
  method delete_cust.
    delete from zcustomers where id = p_id.

* you can also create a fuction to delete and then call in here!


  endmethod.
ENDCLASS.
```