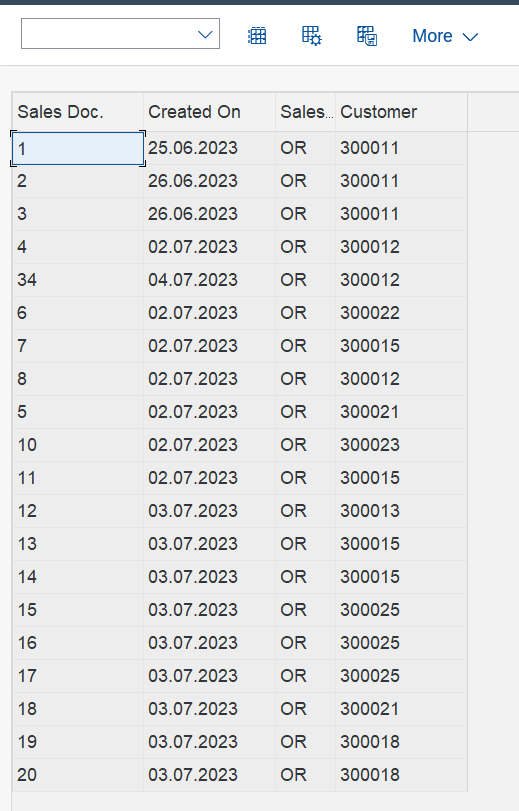
TUTORIALS TASKS:  
1ST.REPORT.  
REPORT ZRK\_REPORT.  
  
CLASS lcl\_report DEFINITION.  
*\**  
 PUBLIC SECTION.  
*\**  
*\* Final output table*  
 TYPES: BEGIN OF ty\_vbak,  
 vbeln TYPE vbak-vbeln,  
 erdat TYPE erdat,  
 auart TYPE auart,  
 kunnr TYPE kunnr,  
 END OF ty\_vbak.  
*\**  
 DATA: t\_vbak TYPE STANDARD TABLE OF ty\_vbak.  
*\**  
*\* ALV reference*  
 DATA: o\_alv TYPE REF TO cl\_salv\_table.  
*\**  
 METHODS:  
*\* data selection*  
 get\_data,  
*\**  
*\* Generating output*  
 generate\_output.  
*\**  
ENDCLASS.  
  
START-OF-SELECTION.  
 DATA: lo\_report TYPE REF TO lcl\_report.  
*\**  
 CREATE OBJECT lo\_report.  
*\**  
 lo\_report->get\_data( ).  
*\**  
 lo\_report->generate\_output( ).  
  
 CLASS lcl\_report IMPLEMENTATION.  
*\**  
 METHOD get\_data.  
*\* data selection*  
 SELECT vbeln erdat auart kunnr  
 INTO TABLE t\_vbak  
 FROM vbak  
 UP TO 20 ROWS.  
*\**  
 ENDMETHOD. *"get\_data*  
*\**  
*\*.......................................................................*  
 METHOD generate\_output.  
*\* New ALV instance*  
*\* We are calling the static Factory method which will give back*  
*\* the ALV object reference.*  
*\**  
*\* exception class*  
 DATA: lx\_msg TYPE REF TO cx\_salv\_msg.  
 TRY.  
 cl\_salv\_table=>factory(  
 IMPORTING  
 r\_salv\_table = o\_alv  
 CHANGING  
 t\_table = t\_vbak ).  
 CATCH cx\_salv\_msg INTO lx\_msg.  
 ENDTRY.  
  
  
 ENDMETHOD. *"generate\_output*  
  
ENDCLASS.

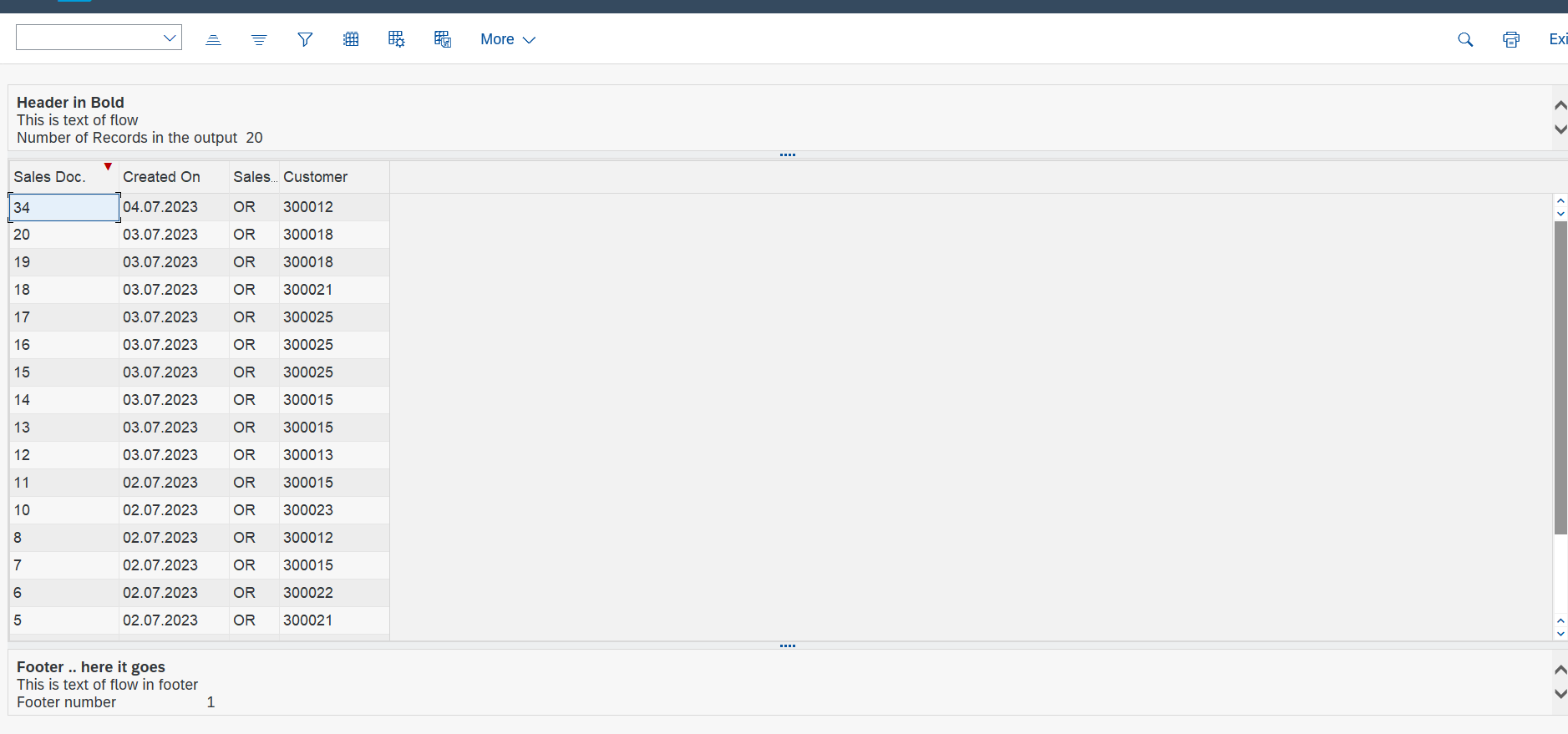


…………………………………………………………………………………\*\*\*…………………………………………………………..  
**LAYOUTS**  
REPORT zrk\_layouts.  
CLASS lcl\_report DEFINITION.  
  
 PUBLIC SECTION.  
  
  
 TYPES: BEGIN OF ty\_vbak,  
 vbeln TYPE vbak-vbeln,  
 erdat TYPE erdat,  
 auart TYPE auart,  
 kunnr TYPE kunnr,  
 END OF ty\_vbak.  
  
 DATA: t\_vbak TYPE STANDARD TABLE OF ty\_vbak.  
  
  
 DATA: o\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
  
 get\_data,  
  
  
 generate\_output.  
  
 PRIVATE SECTION.  
  
 METHODS:  
 set\_pf\_status  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
*\**  
 METHODS:  
 set\_layout  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
  
  
  
  
  
ENDCLASS.  
  
START-OF-SELECTION.  
 DATA: lo\_report TYPE REF TO lcl\_report.  
  
 CREATE OBJECT lo\_report.  
  
 lo\_report->get\_data( ).  
  
 lo\_report->generate\_output( ).  
  
CLASS lcl\_report IMPLEMENTATION.  
*\**  
 METHOD get\_data.  
  
 SELECT vbeln erdat auart kunnr  
 INTO TABLE t\_vbak  
 FROM vbak  
 UP TO 20 ROWS.  
  
 ENDMETHOD.  
  
 METHOD generate\_output.  
  
 DATA: lx\_msg TYPE REF TO cx\_salv\_msg.  
 TRY.  
 cl\_salv\_table=>factory(  
 IMPORTING  
 r\_salv\_table = o\_alv  
 CHANGING  
 t\_table = t\_vbak ).  
 CATCH cx\_salv\_msg INTO lx\_msg.  
 ENDTRY.  
  
  
 CALL METHOD set\_pf\_status  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD set\_layout  
 CHANGING  
 co\_alv = o\_alv.  
  
 DATA: lo\_functions TYPE REF TO cl\_salv\_functions\_list.  
  
 lo\_functions = o\_alv->get\_functions( ). *"--------added*  
 lo\_functions->set\_default( abap\_true ).  
 o\_alv->display( ).  
  
 ENDMETHOD.  
 METHOD set\_pf\_status.  
 co\_alv->set\_screen\_status(  
 report = 'ZBABU\_LAYOUT'  
 pfstatus = 'STANDARD'  
  
 ).  
 ENDMETHOD.  
 METHOD set\_layout.  
  
 DATA: lo\_layout TYPE REF TO cl\_salv\_layout,  
 lf\_variant TYPE slis\_vari,  
 ls\_key TYPE salv\_s\_layout\_key.  
  
 lo\_layout = co\_alv->get\_layout( ).  
  
 ls\_key-report = sy-repid.  
 lo\_layout->set\_key( ls\_key ).  
  
 lo\_layout->set\_save\_restriction( if\_salv\_c\_layout=>restrict\_none ).  
  
 lf\_variant = 'DEFAULT'.  
 lo\_layout->set\_initial\_layout( lf\_variant ).  
  
  
  
 o\_alv->display( ).  
  
  
 ENDMETHOD.  
ENDCLASS.



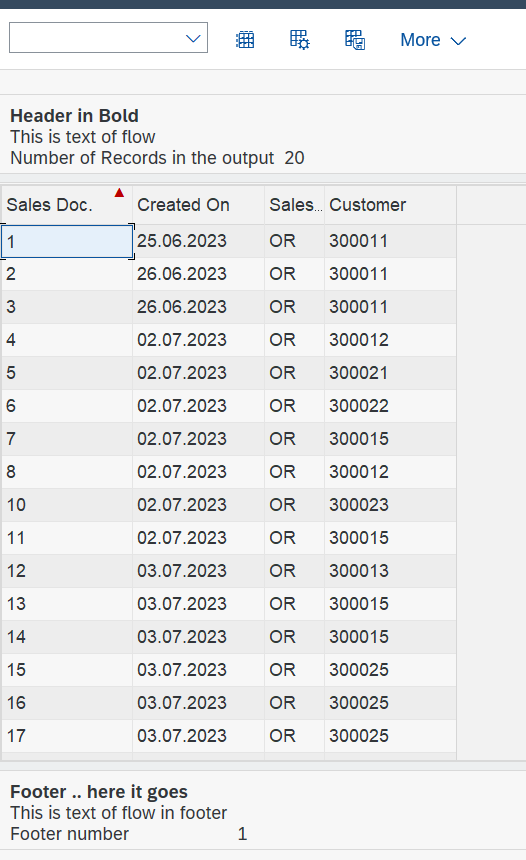
\*…………………………………………………………………………\*\*\*………………………………………………………………………\*  
**SORT DECSENDING**

REPORT ZROHIT\_DISPLAY.  
CLASS lcl\_report DEFINITION.  
  
 PUBLIC SECTION.  
  
  
 TYPES: BEGIN OF ty\_vbak,  
 vbeln TYPE vbak-vbeln,  
 erdat TYPE erdat,  
 auart TYPE auart,  
 kunnr TYPE kunnr,  
 END OF ty\_vbak.  
  
 DATA: t\_vbak TYPE STANDARD TABLE OF ty\_vbak.  
  
  
 DATA: o\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
  
 get\_data,  
  
  
 generate\_output.  
  
 PRIVATE SECTION.  
  
 METHODS:  
 set\_pf\_status  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
*\**  
 METHODS:  
 set\_layout  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
 set\_top\_of\_page  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
 METHODS:  
 set\_end\_of\_page  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
 set\_display\_setting  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
  
  
  
  
  
ENDCLASS.  
  
START-OF-SELECTION.  
 DATA: lo\_report TYPE REF TO lcl\_report.  
  
 CREATE OBJECT lo\_report.  
  
 lo\_report->get\_data( ).  
  
 lo\_report->generate\_output( ).  
  
CLASS lcl\_report IMPLEMENTATION.  
*\**  
 METHOD get\_data.  
  
 SELECT vbeln erdat auart kunnr  
 INTO TABLE t\_vbak  
 FROM vbak  
 UP TO 20 ROWS.  
  
 ENDMETHOD.  
  
 METHOD generate\_output.  
  
 DATA: lx\_msg TYPE REF TO cx\_salv\_msg.  
 TRY.  
 cl\_salv\_table=>factory(  
 IMPORTING  
 r\_salv\_table = o\_alv  
 CHANGING  
 t\_table = t\_vbak ).  
 CATCH cx\_salv\_msg INTO lx\_msg.  
 ENDTRY.  
  
 data: lc\_sort type ref to cl\_salv\_sorts.  
 lc\_sort = O\_alv->get\_sorts( ).  
 try.  
 lc\_sort->add\_sort( columnname = 'VBELN'  
 subtotal = abap\_true  
 sequence = if\_salv\_c\_sort=>sort\_DOWN ).  
 catch: cx\_salv\_not\_found cx\_salv\_data\_error cx\_salv\_existing.  
 endtry.  
 CALL METHOD set\_pf\_status  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD set\_layout  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD me->set\_top\_of\_page  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD me->set\_end\_of\_page  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD set\_display\_setting  
 CHANGING  
 co\_alv = o\_alv.  
  
 DATA: lo\_functions TYPE REF TO cl\_salv\_functions\_list.  
  
 lo\_functions = o\_alv->get\_functions( ).  
 lo\_functions->set\_default( abap\_true ).  
  
  
  
*\*write : /o\_alv*  
  
 o\_alv->display( ).  
  
*\* data: lc\_sort type ref to cl\_salv\_sorts.*  
  
*\*lc\_sort = O\_alv->get\_sorts( ).*  
*\**  
*\* try.*  
*\* lc\_sort->add\_sort( columnname = 'VBELN'*  
*\* subtotal = abap\_true*  
*\* sequence = if\_salv\_c\_sort=>sort\_up ).*  
*\* catch: cx\_salv\_not\_found cx\_salv\_data\_error cx\_salv\_existing.*  
*\* endtry.*  
  
 ENDMETHOD.  
 METHOD set\_pf\_status.  
 co\_alv->set\_screen\_status(  
 report = 'ZBABU\_DISPLAY'  
 pfstatus = 'STANDARD'  
  
 ).  
 ENDMETHOD.  
 METHOD set\_layout.  
  
 DATA: lo\_layout TYPE REF TO cl\_salv\_layout,  
 lf\_variant TYPE slis\_vari,  
 ls\_key TYPE salv\_s\_layout\_key.  
  
 lo\_layout = co\_alv->get\_layout( ).  
  
 ls\_key-report = sy-repid.  
 lo\_layout->set\_key( ls\_key ).  
  
 lo\_layout->set\_save\_restriction( if\_salv\_c\_layout=>restrict\_none ).  
  
 lf\_variant = 'DEFAULT'.  
 lo\_layout->set\_initial\_layout( lf\_variant ).  
  
  
 ENDMETHOD.  
 METHOD set\_top\_of\_page.  
*\**  
 DATA: lo\_header TYPE REF TO cl\_salv\_form\_layout\_grid,  
 lo\_h\_label TYPE REF TO cl\_salv\_form\_label,  
 lo\_h\_flow TYPE REF TO cl\_salv\_form\_layout\_flow.  
*\**  
*\* header object*  
 CREATE OBJECT lo\_header.  
*\**  
*\* To create a Lable or Flow we have to specify the target*  
*\* row and column number where we need to set up the output*  
*\* text.*  
*\**  
*\* information in Bold*  
 lo\_h\_label = lo\_header->create\_label( row = 1 column = 1 ).  
 lo\_h\_label->set\_text( 'Header in Bold' ).  
*\**  
*\* information in tabular format*  
 lo\_h\_flow = lo\_header->create\_flow( row = 2 column = 1 ).  
 lo\_h\_flow->create\_text( text = 'This is text of flow' ).  
*\**  
 lo\_h\_flow = lo\_header->create\_flow( row = 3 column = 1 ).  
 lo\_h\_flow->create\_text( text = 'Number of Records in the output' ).  
*\**  
 lo\_h\_flow = lo\_header->create\_flow( row = 3 column = 2 ).  
 lo\_h\_flow->create\_text( text = 20 ).  
*\**  
*\* set the top of list using the header for Online.*  
 co\_alv->set\_top\_of\_list( lo\_header ).  
*\**  
*\* set the top of list using the header for Print.*  
 co\_alv->set\_top\_of\_list\_print( lo\_header ).  
*\**  
 ENDMETHOD. *"set\_top\_of\_page*  
*\**  
 METHOD set\_end\_of\_page.  
*\**  
 DATA: lo\_footer TYPE REF TO cl\_salv\_form\_layout\_grid,  
 lo\_f\_label TYPE REF TO cl\_salv\_form\_label,  
 lo\_f\_flow TYPE REF TO cl\_salv\_form\_layout\_flow.  
*\**  
*\* footer object*  
 CREATE OBJECT lo\_footer.  
*\**  
*\* information in bold*  
 lo\_f\_label = lo\_footer->create\_label( row = 1 column = 1 ).  
 lo\_f\_label->set\_text( 'Footer .. here it goes' ).  
*\**  
*\* tabular information*  
 lo\_f\_flow = lo\_footer->create\_flow( row = 2 column = 1 ).  
 lo\_f\_flow->create\_text( text = 'This is text of flow in footer' ).  
*\**  
 lo\_f\_flow = lo\_footer->create\_flow( row = 3 column = 1 ).  
 lo\_f\_flow->create\_text( text = 'Footer number' ).  
*\**  
 lo\_f\_flow = lo\_footer->create\_flow( row = 3 column = 2 ).  
 lo\_f\_flow->create\_text( text = 1 ).  
*\**  
*\* Online footer*  
 co\_alv->set\_end\_of\_list( lo\_footer ).  
*\**  
*\* Footer in print*  
 co\_alv->set\_end\_of\_list\_print( lo\_footer ).  
*\**  
  
 ENDMETHOD.  
 METHOD set\_display\_setting.  
*\**  
 DATA: lo\_display TYPE REF TO cl\_salv\_display\_settings.  
*\**  
*\* get display object*  
 lo\_display = co\_alv->get\_display\_settings( ).  
*\**  
*\* set ZEBRA pattern*  
 lo\_display->set\_striped\_pattern( 'X' ).  
*\**  
*\* Title to ALV*  
 lo\_display->set\_list\_header( 'ALV Test for Display Settings' ).  
*\**  
  
  
 o\_alv->display( ).  
  
 ENDMETHOD.  
  
  
  
ENDCLASS.



\*……………………………………………………..\*\*\*………………………………………………………………………………………\*\*

**SORT ACSENDING**  
REPORT ZROHIT\_DISPLAY.  
CLASS lcl\_report DEFINITION.  
  
 PUBLIC SECTION.  
  
  
 TYPES: BEGIN OF ty\_vbak,  
 vbeln TYPE vbak-vbeln,  
 erdat TYPE erdat,  
 auart TYPE auart,  
 kunnr TYPE kunnr,  
 END OF ty\_vbak.  
  
 DATA: t\_vbak TYPE STANDARD TABLE OF ty\_vbak.  
  
  
 DATA: o\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
  
 get\_data,  
  
  
 generate\_output.  
  
 PRIVATE SECTION.  
  
 METHODS:  
 set\_pf\_status  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
*\**  
 METHODS:  
 set\_layout  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
 set\_top\_of\_page  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
 METHODS:  
 set\_end\_of\_page  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
 set\_display\_setting  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
  
  
  
  
  
ENDCLASS.  
  
START-OF-SELECTION.  
 DATA: lo\_report TYPE REF TO lcl\_report.  
  
 CREATE OBJECT lo\_report.  
  
 lo\_report->get\_data( ).  
  
 lo\_report->generate\_output( ).  
  
CLASS lcl\_report IMPLEMENTATION.  
*\**  
 METHOD get\_data.  
  
 SELECT vbeln erdat auart kunnr  
 INTO TABLE t\_vbak  
 FROM vbak  
 UP TO 20 ROWS.  
  
 ENDMETHOD.  
  
 METHOD generate\_output.  
  
 DATA: lx\_msg TYPE REF TO cx\_salv\_msg.  
 TRY.  
 cl\_salv\_table=>factory(  
 IMPORTING  
 r\_salv\_table = o\_alv  
 CHANGING  
 t\_table = t\_vbak ).  
 CATCH cx\_salv\_msg INTO lx\_msg.  
 ENDTRY.  
  
 data: lc\_sort type ref to cl\_salv\_sorts.  
 lc\_sort = O\_alv->get\_sorts( ).  
 try.  
 lc\_sort->add\_sort( columnname = 'VBELN'  
 subtotal = abap\_true  
 sequence = if\_salv\_c\_sort=>sort\_UP ).  
 catch: cx\_salv\_not\_found cx\_salv\_data\_error cx\_salv\_existing.  
 endtry.  
 CALL METHOD set\_pf\_status  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD set\_layout  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD me->set\_top\_of\_page  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD me->set\_end\_of\_page  
 CHANGING  
 co\_alv = o\_alv.  
  
 CALL METHOD set\_display\_setting  
 CHANGING  
 co\_alv = o\_alv.  
  
 DATA: lo\_functions TYPE REF TO cl\_salv\_functions\_list.  
  
 lo\_functions = o\_alv->get\_functions( ).  
 lo\_functions->set\_default( abap\_true ).  
  
  
  
*\*write : /o\_alv*  
  
 o\_alv->display( ).  
  
*\* data: lc\_sort type ref to cl\_salv\_sorts.*  
  
*\*lc\_sort = O\_alv->get\_sorts( ).*  
*\**  
*\* try.*  
*\* lc\_sort->add\_sort( columnname = 'VBELN'*  
*\* subtotal = abap\_true*  
*\* sequence = if\_salv\_c\_sort=>sort\_up ).*  
*\* catch: cx\_salv\_not\_found cx\_salv\_data\_error cx\_salv\_existing.*  
*\* endtry.*  
  
 ENDMETHOD.  
 METHOD set\_pf\_status.  
 co\_alv->set\_screen\_status(  
 report = 'ZBABU\_DISPLAY'  
 pfstatus = 'STANDARD'  
  
 ).  
 ENDMETHOD.  
 METHOD set\_layout.  
  
 DATA: lo\_layout TYPE REF TO cl\_salv\_layout,  
 lf\_variant TYPE slis\_vari,  
 ls\_key TYPE salv\_s\_layout\_key.  
  
 lo\_layout = co\_alv->get\_layout( ).  
  
 ls\_key-report = sy-repid.  
 lo\_layout->set\_key( ls\_key ).  
  
 lo\_layout->set\_save\_restriction( if\_salv\_c\_layout=>restrict\_none ).  
  
 lf\_variant = 'DEFAULT'.  
 lo\_layout->set\_initial\_layout( lf\_variant ).  
  
  
 ENDMETHOD.  
 METHOD set\_top\_of\_page.  
*\**  
 DATA: lo\_header TYPE REF TO cl\_salv\_form\_layout\_grid,  
 lo\_h\_label TYPE REF TO cl\_salv\_form\_label,  
 lo\_h\_flow TYPE REF TO cl\_salv\_form\_layout\_flow.  
*\**  
*\* header object*  
 CREATE OBJECT lo\_header.  
*\**  
*\* To create a Lable or Flow we have to specify the target*  
*\* row and column number where we need to set up the output*  
*\* text.*  
*\**  
*\* information in Bold*  
 lo\_h\_label = lo\_header->create\_label( row = 1 column = 1 ).  
 lo\_h\_label->set\_text( 'Header in Bold' ).  
*\**  
*\* information in tabular format*  
 lo\_h\_flow = lo\_header->create\_flow( row = 2 column = 1 ).  
 lo\_h\_flow->create\_text( text = 'This is text of flow' ).  
*\**  
 lo\_h\_flow = lo\_header->create\_flow( row = 3 column = 1 ).  
 lo\_h\_flow->create\_text( text = 'Number of Records in the output' ).  
*\**  
 lo\_h\_flow = lo\_header->create\_flow( row = 3 column = 2 ).  
 lo\_h\_flow->create\_text( text = 20 ).  
*\**  
*\* set the top of list using the header for Online.*  
 co\_alv->set\_top\_of\_list( lo\_header ).  
*\**  
*\* set the top of list using the header for Print.*  
 co\_alv->set\_top\_of\_list\_print( lo\_header ).  
*\**  
 ENDMETHOD. *"set\_top\_of\_page*  
*\**  
 METHOD set\_end\_of\_page.  
*\**  
 DATA: lo\_footer TYPE REF TO cl\_salv\_form\_layout\_grid,  
 lo\_f\_label TYPE REF TO cl\_salv\_form\_label,  
 lo\_f\_flow TYPE REF TO cl\_salv\_form\_layout\_flow.  
*\**  
*\* footer object*  
 CREATE OBJECT lo\_footer.  
*\**  
*\* information in bold*  
 lo\_f\_label = lo\_footer->create\_label( row = 1 column = 1 ).  
 lo\_f\_label->set\_text( 'Footer .. here it goes' ).  
*\**  
*\* tabular information*  
 lo\_f\_flow = lo\_footer->create\_flow( row = 2 column = 1 ).  
 lo\_f\_flow->create\_text( text = 'This is text of flow in footer' ).  
*\**  
 lo\_f\_flow = lo\_footer->create\_flow( row = 3 column = 1 ).  
 lo\_f\_flow->create\_text( text = 'Footer number' ).  
*\**  
 lo\_f\_flow = lo\_footer->create\_flow( row = 3 column = 2 ).  
 lo\_f\_flow->create\_text( text = 1 ).  
*\**  
*\* Online footer*  
 co\_alv->set\_end\_of\_list( lo\_footer ).  
*\**  
*\* Footer in print*  
 co\_alv->set\_end\_of\_list\_print( lo\_footer ).  
*\**  
  
 ENDMETHOD.  
 METHOD set\_display\_setting.  
*\**  
 DATA: lo\_display TYPE REF TO cl\_salv\_display\_settings.  
*\**  
*\* get display object*  
 lo\_display = co\_alv->get\_display\_settings( ).  
*\**  
*\* set ZEBRA pattern*  
 lo\_display->set\_striped\_pattern( 'X' ).  
*\**  
*\* Title to ALV*  
 lo\_display->set\_list\_header( 'ALV Test for Display Settings' ).  
*\**  
  
  
 o\_alv->display( ).  
  
 ENDMETHOD.  
  
  
  
ENDCLASS.



\*…………………………………………………………………\*\*…………………………………………………………………………………\*

**COLOR**  
REPORT ZRK\_ALV\_COLOR.  
  
CLASS lcl\_report DEFINITION.  
  
 PUBLIC SECTION.  
  
 TYPES: BEGIN OF ty\_vbak,  
 vbeln TYPE vbak-vbeln,  
 erdat TYPE erdat,  
 auart TYPE auart,  
 kunnr TYPE kunnr,  
 t\_color TYPE lvc\_t\_scol,  
 END OF ty\_vbak.  
  
 DATA: t\_vbak TYPE STANDARD TABLE OF ty\_vbak.  
 TYPES: ty\_t\_vbak TYPE STANDARD TABLE OF ty\_vbak.  
  
 DATA: o\_alv TYPE REF TO cl\_salv\_table.  
  
 METHODS:  
 get\_data,  
 generate\_output.  
  
  
*\*private SECTION.*  
PRIVATE SECTION.  
METHODS:  
 set\_pf\_status  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table.  
METHODS:  
 set\_colors  
 CHANGING  
 co\_alv TYPE REF TO cl\_salv\_table  
 ct\_vbak TYPE ty\_t\_vbak.  
  
ENDCLASS.  
  
  
START-OF-SELECTION.  
 DATA: lo\_report TYPE REF TO lcl\_report.  
 *"DATA: o\_alv TYPE REF TO cl\_salv\_table.*  
  
 CREATE OBJECT lo\_report.  
  
 lo\_report->get\_data( ).  
  
 lo\_report->generate\_output( ).  
  
  
  
  
 CLASS lcl\_report IMPLEMENTATION.  
  
  
  
  
 METHOD get\_data.  
  
 SELECT vbeln erdat auart kunnr  
 INTO CORRESPONDING FIELDS OF TABLE t\_vbak  
 FROM vbak  
 UP TO 20 ROWS.  
  
 ENDMETHOD.  
  
 METHOD generate\_output.  
  
 DATA: lx\_msg TYPE REF TO cx\_salv\_msg.  
 TRY.  
 cl\_salv\_table=>factory(  
 IMPORTING  
 r\_salv\_table = o\_alv  
 CHANGING  
 t\_table = t\_vbak ).  
 CATCH cx\_salv\_msg INTO lx\_msg.  
 ENDTRY.  
  
CALL METHOD set\_pf\_status  
 CHANGING  
 co\_alv = o\_alv.  
*\* Set the colors to ALV display*  
 CALL METHOD set\_colors  
 CHANGING  
 co\_alv = o\_alv  
 ct\_vbak = t\_vbak.  
  
*\* Setting Default PFSTATUS USING SALV functions*  
DATA: lo\_functions TYPE REF TO cl\_salv\_functions\_list.  
  
 lo\_functions = o\_alv->get\_functions( ).  
 lo\_functions->set\_default( abap\_true ).  
 o\_alv->display( ).  
  
 ENDMETHOD.  
  
 METHOD set\_pf\_status.  
*\*\**  
  
*\* Calling method to set the PF-Status*  
 co\_alv->set\_screen\_status(  
 pfstatus = 'ZVJ2\_ALV'  
 report = 'ZVJ\_ALV\_REPORT'  
 set\_functions = co\_alv->c\_functions\_all  
 ).  
*\*\**  
 ENDMETHOD.  
 METHOD set\_colors.  
*\**  
*\*.....Color for COLUMN.....*  
 DATA: lo\_cols\_tab TYPE REF TO cl\_salv\_columns\_table,  
 lo\_col\_tab TYPE REF TO cl\_salv\_column\_table.  
 DATA: ls\_color TYPE lvc\_s\_colo. *" Colors strucutre*  
*\**  
*\* get Columns object*  
 lo\_cols\_tab = co\_alv->get\_columns( ).  
*\**  
 INCLUDE <color>.  
*\**  
*\* Get ERDAT column & set the yellow Color fot it*  
 TRY.  
 lo\_col\_tab ?= lo\_cols\_tab->get\_column( 'AUART' ).  
 ls\_color-col = col\_total.  
 lo\_col\_tab->set\_color( ls\_color ).  
 CATCH cx\_salv\_not\_found.  
 ENDTRY.  
*\* Get ERDAT column & set the yellow Color fot it*  
 TRY.  
 lo\_col\_tab ?= lo\_cols\_tab->get\_column( 'ERDAT' ).  
 ls\_color-col = col\_group.  
 lo\_col\_tab->set\_color( ls\_color ).  
 CATCH cx\_salv\_not\_found.  
 ENDTRY.  
*\**  
*\*.......Color for Specific Cell & Rows.................*  
*\* Applying color on the 3rd Row and Column AUART*  
*\* Applying color on the Entire 5th Row*  
*\**  
 DATA: lt\_s\_color TYPE lvc\_t\_scol,  
 ls\_s\_color TYPE lvc\_s\_scol,  
 la\_vbak LIKE LINE OF ct\_vbak,  
 l\_count TYPE i.  
*\**  
 LOOP AT ct\_vbak INTO la\_vbak.  
 l\_count = l\_count + 1.  
 CASE l\_count.  
*\* Apply RED color to the AUART Cell of the 3rd Row*  
 WHEN 3.  
 ls\_s\_color-fname = 'AUART'.  
 ls\_s\_color-color-col = col\_negative.  
 ls\_s\_color-color-int = 0.  
 ls\_s\_color-color-inv = 0.  
 APPEND ls\_s\_color TO lt\_s\_color.  
 CLEAR ls\_s\_color.  
*\**  
*\* Apply GREEN color to the entire row # 5*  
*\* For entire row, we don't pass the Fieldname*  
 WHEN 5.  
 ls\_s\_color-color-col = col\_positive.  
 ls\_s\_color-color-int = 0.  
 ls\_s\_color-color-inv = 0.  
 APPEND ls\_s\_color TO lt\_s\_color.  
 CLEAR ls\_s\_color.  
  
 WHEN 10.  
 ls\_s\_color-color-col = col\_group.  
 ls\_s\_color-color-int = 0.  
 ls\_s\_color-color-inv = 0.  
 APPEND ls\_s\_color TO lt\_s\_color.  
 CLEAR ls\_s\_color.  
 ENDCASE.  
*\*\* Modify that data back to the output table*  
 la\_vbak-t\_color = lt\_s\_color.  
 MODIFY ct\_vbak FROM la\_vbak.  
 CLEAR la\_vbak.  
 CLEAR lt\_s\_color.  
 ENDLOOP.  
*\**  
*\* We will set this COLOR table field name of the internal table to*  
*\* COLUMNS tab reference for the specific colors*  
 TRY.  
 lo\_cols\_tab->set\_color\_column( 'T\_COLOR' ).  
 CATCH cx\_salv\_data\_error. *"#EC NO\_HANDLER*  
 ENDTRY.  
*\**  
 ENDMETHOD. *"set\_colors*  
 ENDCLASS.

