```
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.


    TYPES: ty_t_vbak TYPE STANDARD TABLE 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.
    METHODS:
      set_columns
        CHANGING
          co_alv TYPE REF TO cl_salv_table.


    METHODS:
      set_hotspot_vbeln
        CHANGING
          co_alv    TYPE REF TO cl_salv_table
          co_report TYPE REF TO lcl_report.
*
*   Event Handler for HOTSPOT event
    METHODS:
      on_link_click
        FOR EVENT link_click OF cl_salv_events_table
        IMPORTING
          row
          column  .



    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.

  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.
DATA: lc_sort  TYPE REF TO cl_salv_sorts.
TRY.
    lc_sort = O_alv->get_sorts( ).

    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.

CALL METHOD me->set_columns
  CHANGING
    co_alv = o_alv.

CALL METHOD set_hotspot_vbeln
  CHANGING
    co_alv    = o_alv
    co_report = lo_report.

CALL METHOD set_colors
  CHANGING
    co_alv  = o_alv
    ct_vbak = t_vbak.
    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.
    co_alv->set_screen_status(
      report        = 'ZBABU_HANDLE'
      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' ).
*

  ENDMETHOD.
  METHOD set_columns.
*
*...Get all the Columns
    DATA: lo_cols TYPE REF TO cl_salv_columns.
    lo_cols = o_alv->get_columns( ).
*
*   set the Column optimization
    lo_cols->set_optimize( 'X' ).
*
*...Process individual columns
    DATA: lo_column TYPE REF TO cl_salv_column.
*
*   Change the properties of the Columns KUNNR
    TRY.
        lo_column = lo_cols->get_column( 'KUNNR' ).
        lo_column->set_long_text( 'Sold-To Party' ).
        lo_column->set_medium_text( 'Sold-To Party' ).
        lo_column->set_short_text( 'Sold-To' ).
        lo_column->set_output_length( 10 ).
      CATCH cx_salv_not_found.                          "#EC NO_HANDLER
    ENDTRY.

  ENDMETHOD.

  METHOD set_hotspot_vbeln.
*
*...HotSpot
    DATA: lo_cols_tab TYPE REF TO cl_salv_columns_table,
          lo_col_tab  TYPE REF TO cl_salv_column_table.
*
*   get Columns object
    lo_cols_tab = co_alv->get_columns( ).
*
*   Get VBELN column
    TRY.
        lo_col_tab ?= lo_cols_tab->get_column( 'VBELN' ).
      CATCH cx_salv_not_found.
    ENDTRY.
*
*   Set the HotSpot for VBELN Column
    TRY.
        CALL METHOD lo_col_tab->set_cell_type
          EXPORTING
            value = if_salv_c_cell_type=>hotspot.
        .
      CATCH cx_salv_data_error .
    ENDTRY.
*
*...Events
    DATA: lo_events TYPE REF TO cl_salv_events_table.
*
*   all events
    lo_events = o_alv->get_event( ).
*
*   event handler
    SET HANDLER co_report->on_link_click FOR lo_events.
*
  ENDMETHOD.                    "set_hotspot_vbeln
*
* Handles the UI on the VBELN (HotSpot)
  METHOD on_link_click.
*
    DATA: la_vbak TYPE ty_vbak.
*
*   Get the Sales Order number from the table
    READ TABLE lo_report->t_vbak INTO la_vbak INDEX row.
    IF la_vbak-vbeln IS NOT INITIAL.
      MESSAGE i398(00) WITH 'You have selected' la_vbak-vbeln.
    ENDIF.

  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( 'ERDAT' ).
        ls_color-col = col_total.
        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.
      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.
*

    o_alv->display( ).

  ENDMETHOD.




ENDCLASS.
```