Beispiel-Quellcodes aus dem Buch „Praxishandbuch BOPF - Business Object Processing Framework. Das neue SAP S/4HANA-Programmiermode“ (Christoph Lordieck, Espresso Tutorials 2019)

Übersicht

Listing 3.1 – Erzeugung eines Transaktionsmanagers

```
DATA lo_transaction_manager     TYPE REF TO /bobf/if_tra_transaction_mgr.

lo_transaction_manager     =
/bobf/cl_tra_trans_mgr_factory=>get_transaction_manager( ).
```

Listing 3.2 – Erzeugung eines Servicemanagers zu Ihrem Geschäftsobjekt

```
DATA lo_service_manager   TYPE REF TO /bobf/if_tra_service_manager.

lo_service_manager =
      /bobf/cl_tra_serv_mgr_factory=>get_service_manager(
        iv_bo_key = zif_kr_kundenrechnung_c=>sc_bo_key ).
```

Listing 3.3 – Daten über eine Assoziation abfragen

```
DATA lo_service_manager_kr  TYPE REF TO /bobf/if_tra_service_manager.

lo_service_manager_kr->retrieve_by_association(
  EXPORTING
    iv_node_key       = zif_kr_kundenrechnung_c=>sc_node-kopf
    it_key            = lt_kopf_keys
    iv_association    = zif_kr_kundenrechnung_c=>sc_association-kopf-
position
    iv_fill_data      = abap_true
    iv_edit_mode      = /bobf/if_conf_c=>sc_edit_read_only
  IMPORTING
    et_data           = lt_position_data
    et_target_key     = lt_target_key
).
```

Listing 4.1 – Implementierte Aktion „Rechnung bezahlen“

```
METHOD /bobf/if_frw_action~execute.
  DATA lt_failed_key        TYPE /bobf/t_frw_key.
  DATA lt_kopf_data         TYPE zkr_t_kopf. "Kombi-Tabellentyp
  DATA lo_message           TYPE REF TO /bobf/if_frw_message.
  DATA lt_changed_fields    TYPE /bobf/t_frw_name.
  DATA ls_symsg             TYPE symsg.
  DATA lv_message           TYPE string.

  "Vorbereitung der Rückgabeparameter
  eo_message = /bobf/cl_frw_factory=>get_message( ).
  FREE et_failed_key[].
  FREE et_data[].


  "Daten zu Instanzen in IT_KEY lesen
  io_read->retrieve(
    EXPORTING
      "Aktueller Knoten des GO aus dem Kontext
      iv_node                 = is_ctx-node_key
      it_key                  = it_key
```

```
      iv_fill_data            = abap_true
*      it_requested_attributes = "Obsolet, nicht benutzen!
    IMPORTING
      eo_message              = lo_message
      et_data                 = lt_kopf_data
      et_failed_key           = lt_failed_key
  ).


  "Fehlerhandling
  APPEND LINES OF lt_failed_key
    TO et_failed_key.
  FREE lt_failed_key.

  "ACHTUNG: Nicht alle Standard-BOPF-Implementierungen
  "legen das Objekt eo_message an!
  IF lo_message IS BOUND.
    eo_message->add( lo_message ).
    FREE lo_message.
  ENDIF.


  "Zu ändernde Felder spezifizieren => Das Framework übernimmt dann nur
  "diese Änderungen
  APPEND zif_kr_kundenrechnung_c=>sc_node_attribute-kopf-status
    TO lt_changed_fields.


  "Pro KOPF-Datensatz den Status prüfen und den neuen setzen
  LOOP AT lt_kopf_data
    REFERENCE INTO DATA(ld_kopf_data).

    IF ld_kopf_data->status = '1'.    "= Offen;

      ld_kopf_data->status = '2'.     "Neuen Status setzen

      io_modify->update(
        EXPORTING
          iv_node           = is_ctx-node_key
          iv_key            = ld_kopf_data->key
          "IS_DATA ist ein Referenzparameter!
          is_data           = ld_kopf_data
          it_changed_fields = lt_changed_fields
      ).

    ELSE.

      "Fehlerhandling -> Status nicht "Offen". Aktion wird für diese
Instanz
      "nicht durchgeführt, daher wird der KEY zu ET_FAILED_KEY hinzugefügt.
      READ TABLE it_key
        WITH TABLE KEY key = ld_kopf_data->key
        REFERENCE INTO DATA(ld_key).
      IF sy-subrc = 0.
        INSERT ld_key->*
          INTO TABLE et_failed_key.
        FREE ld_key.
      ENDIF.

      "Nachricht mitgeben und auf die richtige Instanz und
      "das Attribut verweisen
```

```
      "Rechnungsstatus von { ld_kopf_data->rechnungsnr } ist nicht
"offen"|.
      MESSAGE ID 'ZKR_MESSAGES'
         TYPE /bobf/cm_frw=>co_severity_error
       NUMBER 001
         WITH ld_kopf_data->rechnungsnr
         INTO lv_message.
      MOVE-CORRESPONDING sy TO ls_symsg.

      eo_message->add_message(
        EXPORTING
          is_msg       = ls_symsg
          iv_node      = is_ctx-node_key
          iv_key       = ld_kopf_data->key
          iv_attribute = zif_kr_kundenrechnung_c=>sc_node_attribute-kopf-
status
      ).

      FREE ls_symsg.
      FREE lv_message.
    ENDIF.
  ENDLOOP.



  "Änderungen anstoßen und Meldungen mit als Ergebnis rausgeben
  io_modify->end_modify(
    IMPORTING
      eo_message       = lo_message
  ).

  IF lo_message IS BOUND.
    eo_message->add( lo_message ).
  ENDIF.

ENDMETHOD.
```

Listing 4.2 – Ermittlung zur Berechnung der Positionssumme

```
METHOD /bobf/if_frw_determination~execute.
  DATA lt_pos_data       TYPE zkr_t_position.
  DATA lt_changed_fields TYPE /bobf/t_frw_name.

  io_read->retrieve(
    EXPORTING
      iv_node                 = is_ctx-node_key
      it_key                  = it_key
      iv_fill_data            = abap_true
    IMPORTING
      et_data                 = lt_pos_data
  ).


  LOOP AT lt_pos_data
    REFERENCE INTO DATA(ld_pos_data).

    ld_pos_data->gesamtbetrag =
        ld_pos_data->menge * ld_pos_data->preis_einzel.

    io_modify->update(
      EXPORTING
```

```
        iv_node           = is_ctx-node_key
        iv_key            = ld_pos_data->key
        iv_root_key       = ld_pos_data->root_key
        is_data           = ld_pos_data
    ).

  ENDLOOP.

ENDMETHOD.
```

Listing 4.3 – Konsistenzvalidierung auf Positionsmenge und -preis

```
METHOD /bobf/if_frw_validation~execute.
  DATA lt_pos_data    TYPE zkr_t_position.
  DATA ls_location    TYPE /bobf/s_frw_location.
  DATA lo_cm_kr       TYPE REF TO zcm_kundenrechnung.

  eo_message = /bobf/cl_frw_factory=>get_message( ).
  FREE et_failed_key[].

  io_read->retrieve(
    EXPORTING
      iv_node                 = is_ctx-node_key
      it_key                  = it_key
      iv_fill_data            = abap_true
    IMPORTING
      et_data                 = lt_pos_data
  ).

  ls_location-bo_key    = is_ctx-bo_key.
  ls_location-node_key  = is_ctx-node_key.
  APPEND
    zif_kr_kundenrechnung_c=>sc_node_attribute-position-menge
    TO ls_location-attributes.
  APPEND
    zif_kr_kundenrechnung_c=>sc_node_attribute-position-preis_einzel
    TO ls_location-attributes.


  LOOP AT lt_pos_data
    REFERENCE INTO DATA(ld_pos_data).

    IF ld_pos_data->menge <= 0
    OR ld_pos_data->preis_einzel <= 0.

      READ TABLE it_key
        WITH TABLE KEY key = ld_pos_data->key
        REFERENCE INTO DATA(ld_key).
      CHECK ld_key IS BOUND.

      INSERT ld_key->*
        INTO TABLE et_failed_key.
      FREE ld_key.

      ls_location-key = ld_pos_data->key.
      CREATE OBJECT lo_cm_kr
        EXPORTING
          ms_origin_location = ls_location
          textid             =
zcm_kundenrechnung=>menge_oder_preis_ist_null
```

```
          severity           = zcm_kundenrechnung=>co_severity_error
          mv_attr1           = |{ ld_pos_data->positionsnr }|.

      eo_message->add_cm( lo_cm_kr ).
      FREE lo_cm_kr.

    ENDIF.
  ENDLOOP.

ENDMETHOD.
```

Listing 4.4 – Abfrage zur Suche von Kundenrechnungen zu bestimmten Artikelnummern

```
METHOD /bobf/if_frw_query~query.
  DATA lt_r_artnr   TYPE RANGE OF artnr.
  DATA lt_kopf_key  TYPE /bobf/t_frw_key.

  eo_message = /bobf/cl_frw_factory=>get_message( ).
  FREE et_key.
  FREE et_data.

  "Range-Werte in echte Range-Tabelle übernehmen
  LOOP AT it_selection_parameters
    REFERENCE INTO DATA(ld_sel_param).

    APPEND INITIAL LINE TO lt_r_artnr
      ASSIGNING FIELD-SYMBOL(<ls_r_artnr>).
    <ls_r_artnr>-sign   = ld_sel_param->sign.
    <ls_r_artnr>-option = ld_sel_param->option.
    <ls_r_artnr>-low    = ld_sel_param->low.
    <ls_r_artnr>-high   = ld_sel_param->high.
    UNASSIGN <ls_r_artnr>.
  ENDLOOP.

  "Relevante KOPF-Keys suchen
  SELECT k~db_key
    FROM zkr_d_kopf AS k
    INNER JOIN zkr_d_position AS p
      ON p~parent_key = k~db_key
    INTO TABLE @lt_kopf_key
   WHERE p~artikelnr IN @lt_r_artnr.
  IF sy-subrc = 0.

    "Query mit Filter auf gefundene Keys ausführen
    io_query->query(
      EXPORTING
        is_ctx                 = is_ctx
        it_filter_key          = lt_kopf_key
        io_query_authorities   = io_query_authorities
        is_query_options       = is_query_options
        io_query               = io_query
        io_read                = io_read
        io_modify              = io_modify
        iv_fill_data           = iv_fill_data
        it_requested_attributes = it_requested_attributes
      IMPORTING
        eo_message             = eo_message
        et_key                 = et_key
        es_query_info          = es_query_info
        et_data                = et_data
```

```
    ).
  ENDIF.

ENDMETHOD.
```

Listing 4.5 – Assoziation von KOPF zu Positionen mit einer Menge >10

```
METHOD /bobf/if_frw_association~resolve.
    DATA lt_pos_data            TYPE zkr_t_position.
    DATA lv_mindestmenge        TYPE menge_d.

    FIELD-SYMBOLS <ls_params>   TYPE zkr_s_c_positionsmenge_gt_x.

    "Filterstruktur lesbar machen
    IF is_parameters IS BOUND.
      ASSIGN is_parameters->* TO <ls_params>.
      lv_mindestmenge = <ls_params>-menge.
    ENDIF.

    "Positionsdaten über reg. Assoziation lesen
    io_read->retrieve_by_association(
      EXPORTING
        iv_node                 = is_ctx-node_key
        it_key                  = it_key
        iv_association          = zif_kr_kundenrechnung_c=>sc_association-
kopf-position
        iv_fill_data            = abap_true
      IMPORTING
        et_data                 = lt_pos_data
        et_key_link             = DATA(lt_key_link)
        et_failed_key           = DATA(lt_failed_key)
    ).

    INSERT LINES OF lt_failed_key INTO TABLE et_failed_key.

    "Positionen mit zu kleiner Menge aussortieren
    LOOP AT lt_pos_data
      REFERENCE INTO DATA(ld_position)
      WHERE menge < lv_mindestmenge.

      READ TABLE lt_key_link
        WITH TABLE KEY target_key
        COMPONENTS target_key = ld_position->key
        INTO DATA(ls_key_link).
      IF sy-subrc = 0.
        DELETE TABLE lt_key_link FROM ls_key_link.

        "Gibt es noch eine Position zum Kopf?
        READ TABLE lt_key_link
          WITH KEY source_key = ls_key_link-source_key
          TRANSPORTING NO FIELDS.
        IF sy-subrc <> 0.
          "Keine gefunden = Failed-Key
          INSERT VALUE #( key = ls_key_link-source_key )
            INTO TABLE et_failed_key.
        ENDIF.

        FREE ls_key_link.
      ENDIF.
    ENDLOOP.
```

```
    et_key_link = lt_key_link.

  ENDMETHOD.
```

Listing 5.1 – Generische Ermittlung des Kombi-Tabellentypen eines Knotens über die Konfiguration

```
DATA:
  ls_node       TYPE /bobf/s_confro_node,
  lt_data       TYPE REF TO data,
  lo_conf       TYPE REF TO /bobf/if_frw_configuration.

FIELD-SYMBOLS:
  <lt_data>     TYPE INDEX TABLE.


lo_conf = /bobf/cl_frw_factory=>get_configuration( is_ctx-bo_key ).

lo_conf->get_node(
  EXPORTING
    iv_node_key = is_ctx-node_key
  IMPORTING
    es_node     = ls_node ).


CREATE DATA lt_data TYPE (ls_node-data_table_type).
ASSIGN lt_data->* TO <lt_data>.
```

Listing 6.1 – Zentrale Funktion als Auslöser für BOPF-Ausnahmen

```
METHOD set_application_error.

  DATA:
    lx_previous          TYPE REF TO cx_root,
    lv_previous_text     TYPE        string,              "#EC NEEDED
    lx_pre_previous      TYPE REF TO cx_root,             "#EC NEEDED
    lv_pre_previous_text TYPE        string,              "#EC NEEDED
    lv_prev_progname     TYPE        syrepid,             "#EC NEEDED
    lv_prev_inclname     TYPE        syrepid,             "#EC NEEDED
    lv_prev_line         TYPE        i,                   "#EC NEEDED
    lv_pre_prev_progname TYPE        syrepid,             "#EC NEEDED
    lv_pre_prev_inclname TYPE        syrepid,             "#EC NEEDED
    lv_pre_prev_line     TYPE        i.                   "#EC NEEDED

  " remember the fatal exception in order to refuse any further transaction
manager calls
  gv_application_error = abap_true.

  " enrich to ease dump analysing
  IF io_exception           IS BOUND AND
     io_exception->previous IS BOUND.

    lx_previous = io_exception->previous.
    lv_previous_text = lx_previous->get_text( ).

    lx_previous->get_source_position(
```

```
      IMPORTING
        program_name = lv_prev_progname
        include_name = lv_prev_inclname
        source_line  = lv_prev_line ).

    IF lx_previous->previous IS BOUND.
       lx_pre_previous = lx_previous->previous.
       lv_pre_previous_text = lx_pre_previous->get_text( ).
       lx_previous->get_source_position(
        IMPORTING
          program_name = lv_pre_prev_progname
          include_name = lv_pre_prev_inclname
          source_line  = lv_pre_prev_line ).
    ENDIF.

  ENDIF.

  RAISE EXCEPTION TYPE /bobf/cx_frw_fatal
    EXPORTING
      textid   = /bobf/cx_frw_fatal=>sc_application_error
      previous = io_exception.


ENDMETHOD.                    "set_application_error
```

Listing 7.1 – Rechnung anlegen und Aktion RECHNUNG_BEZAHLEN aufrufen

```
REPORT ZKR_CREATE_AND_PAY.

PARAMETERS pa_rechn   TYPE zkr_s_kopf-rechnungsnr.
PARAMETERS pa_kunnr   TYPE zkr_s_kopf-kundennr.
PARAMETERS pa_artnr   TYPE zkr_s_position-artikelnr.
PARAMETERS pa_menge   TYPE zkr_s_position-menge.

*---------------------------------------------------------------------*
START-OF-SELECTION.

  "Transaktions- und Servicemanager
  DATA lo_transaction_manager     TYPE REF TO /bobf/if_tra_transaction_mgr.
  DATA lo_service_manager_kr      TYPE REF TO /bobf/if_tra_service_manager.

  "Tabellen: Alternativschlüssel, BOPF-Keys, Kopf-Daten,
  "Position-Daten, Modification-Tabelle
  DATA lt_altkey_rechnungsnr      TYPE zkr_t_altkey_rechnum.
  DATA lt_bopf_keys               TYPE /bobf/t_frw_key.
  DATA lt_kopf_data               TYPE zkr_t_kopf.
  DATA lt_position_data           TYPE zkr_t_position.
  DATA lt_modification            TYPE /bobf/t_frw_modification.

  "Strukturen/Referenzen: Alternativschlüssel, Kopf-Daten,
  "Position-Daten, Modification-Struktur
  DATA ls_altkey_rechnungsnr      TYPE zkr_s_altkey_rechnum.
  DATA ls_root_data               TYPE zkr_s_kopf.
  DATA ls_position_data           TYPE zkr_s_position.
  DATA ls_modification_kopf       TYPE /bobf/s_frw_modification.
  DATA ld_modification_pos        TYPE REF TO /bobf/s_frw_modification.

* Schritte:
*  1. Transaktionsmanager holen
*  2. Service-Manager zu unserem BO holen -> Konstanten-Interface!
```

```
*   3. Daten für Kopf zusammenstellen
*   4. Modification-Tabelle füllen
*   5. Kopf-Satz anlegen
*   6. Kopf-Schlüssel über Alternativ-Schlüssel auslesen
*   7. Position-Satz anlegen
*   8. <was auch immer noch getan werden soll>
*   9. Save über Transaktionsmanager

  lo_transaction_manager    =
/bobf/cl_tra_trans_mgr_factory=>get_transaction_manager( ).
  lo_service_manager_kr     =
      /bobf/cl_tra_serv_mgr_factory=>get_service_manager(
        iv_bo_key = zif_kr_kundenrechnung_c=>sc_bo_key ).


  ls_root_data-rechnungsnr    = pa_rechn.
  ls_root_data-kundennr       = pa_kunnr.
  ls_root_data-status         = '1'.      "= Offen

  ls_modification_kopf-node         = zif_kr_kundenrechnung_c=>sc_node-
kopf.
  ls_modification_kopf-change_mode  = /bobf/if_frw_c=>sc_modify_create.

  CREATE DATA ls_modification_kopf-data TYPE zkr_s_kopf.
  GET REFERENCE OF ls_root_data INTO ls_modification_kopf-data.

  INSERT ls_modification_kopf
    INTO TABLE lt_modification.

  "Über Service-Manager die Modifikation durchführen lassen
  DATA(lo_message) = /bobf/cl_frw_factory=>get_message( ).
  lo_service_manager_kr->modify(
    EXPORTING
      it_modification = lt_modification
    IMPORTING
      eo_message      = lo_message
  ).

  "Check( ) ist bei Error-Meldungen = 'X'
  CHECK lo_message->check( ) = abap_false.


  "Schlüssel zum neuen Satz lesen und Daten holen
  ls_altkey_rechnungsnr-rechnungsnr   = pa_rechn.
  APPEND ls_altkey_rechnungsnr TO lt_altkey_rechnungsnr.

  lo_service_manager_kr->convert_altern_key(
    EXPORTING
      iv_node_key      = zif_kr_kundenrechnung_c=>sc_node-kopf
      iv_altkey_key    = zif_kr_kundenrechnung_c=>sc_alternative_key-kopf-
rechnungsnr
      it_key           = lt_altkey_rechnungsnr
    IMPORTING
      et_key           = lt_bopf_keys
  ).


  lo_service_manager_kr->retrieve(
    EXPORTING
      iv_node_key             = zif_kr_kundenrechnung_c=>sc_node-kopf
      it_key                  = lt_bopf_keys
      iv_fill_data            = abap_true
    IMPORTING
```

```
      et_data                 = lt_kopf_data
  ).


  READ TABLE lt_kopf_data
    INDEX 1
    INTO DATA(ls_kopf_data).

  CHECK sy-subrc = 0.

  "Jetzt das Item anlegen
  ls_position_data-artikelnr    = pa_artnr.
  ls_position_data-positionsnr  = '1'.
  ls_position_data-menge        = pa_menge.

  "Jetzt kommt der Teil mit Association! Item anlegen
  FREE lt_modification[].
  APPEND INITIAL LINE TO lt_modification
    REFERENCE INTO ld_modification_pos.

  ld_modification_pos->node         = zif_kr_kundenrechnung_c=>sc_node-
position.
  ld_modification_pos->change_mode  = /bobf/if_frw_c=>sc_modify_create.
  ld_modification_pos->association  =
zif_kr_kundenrechnung_c=>sc_association-position-to_parent.
  ld_modification_pos->root_key     = ls_kopf_data-key.
  ld_modification_pos->source_key   = ls_kopf_data-key.
  ld_modification_pos->source_node  = zif_kr_kundenrechnung_c=>sc_node-
kopf.

  CREATE DATA ld_modification_pos->data TYPE zkr_s_position.
  GET REFERENCE OF ls_position_data INTO ld_modification_pos->data.

  DATA(lo_message_position) = /bobf/cl_frw_factory=>get_message( ).
  lo_service_manager_kr->modify(
    EXPORTING
      it_modification = lt_modification
    IMPORTING
      eo_message      = lo_message_position
  ).

  "Item Daten von Root aus lesen
  lo_service_manager_kr->retrieve_by_association(
    EXPORTING
      iv_node_key             = zif_kr_kundenrechnung_c=>sc_node-kopf
      it_key                  = lt_bopf_keys
      iv_association          = zif_kr_kundenrechnung_c=>sc_association-
kopf-position
      iv_fill_data            = abap_true
    IMPORTING
      et_data                 = lt_position_data
  ).


  "Angelegte Rechnungen über Aktion bezahlen
  lo_service_manager_kr->do_action(
    EXPORTING
      iv_act_key           = zif_kr_kundenrechnung_c=>sc_action-kopf-
a_rechnung_bezahlen
      it_key               = lt_bopf_keys
  ).

  "Speichern der Daten
```

```
  lo_transaction_manager->save( ).
```

Listing 7.2 – Erzeugen und bezahlen einer neuen Rechnung mittels Modell-Klasse

```
REPORT zkr_create_and_pay.

PARAMETERS pa_rechn   TYPE zkr_s_kopf-rechnungsnr.
PARAMETERS pa_kunnr   TYPE zkr_s_kopf-kundennr.
PARAMETERS pa_artnr   TYPE zkr_s_position-artikelnr.
PARAMETERS pa_menge   TYPE zkr_s_position-menge.

*********************************************
CLASS lcl_kundenrechnung_model DEFINITION
  CREATE PUBLIC.

  PUBLIC SECTION.
    CLASS-METHODS class_constructor.

    CLASS-METHODS create_new_kr
      IMPORTING
                is_kopf                  TYPE zkr_s_kopf
                it_position              TYPE zkr_t_position
      RETURNING VALUE(ro_kundenrechnung) TYPE REF TO
lcl_kundenrechnung_model.

    METHODS constructor
      IMPORTING iv_rechnungsnr TYPE zkr_e_rechnungsnr.

    METHODS pay.

  PRIVATE SECTION.
    CLASS-DATA mo_service_manager   TYPE REF TO
/bobf/if_tra_service_manager.

    DATA  ms_kopf   TYPE zkr_s_kopf.
ENDCLASS.

CLASS lcl_kundenrechnung_model IMPLEMENTATION.
  METHOD class_constructor.
    mo_service_manager =
      /bobf/cl_tra_serv_mgr_factory=>get_service_manager(
        iv_bo_key = zif_kr_kundenrechnung_c=>sc_bo_key ).
  ENDMETHOD.

  METHOD create_new_kr.
    DATA lt_modification       TYPE /bobf/t_frw_modification.
    DATA ls_modification_kopf  TYPE /bobf/s_frw_modification.
    DATA ld_modification_pos   TYPE REF TO /bobf/s_frw_modification.

    "Schlüssel holen, um KOPF und POSITION gleichzeitig anlegen zu können
    DATA(lv_kopf_key)    = /bobf/cl_frw_factory=>get_new_key( ).

    "Kopf-Modifikationseintrag anlegen
    ls_modification_kopf-node         = zif_kr_kundenrechnung_c=>sc_node-
kopf.
    ls_modification_kopf-change_mode  = /bobf/if_frw_c=>sc_modify_create.
    ls_modification_kopf-key          = lv_kopf_key.

    CREATE DATA ls_modification_kopf-data TYPE zkr_s_kopf.
    ASSIGN ls_modification_kopf-data->* TO FIELD-SYMBOL(<ls_mod>).
```

```
    <ls_mod> = is_kopf.

    INSERT ls_modification_kopf
      INTO TABLE lt_modification.


    "Position-Modifikationseintrag anlegen
    LOOP AT it_position
      REFERENCE INTO DATA(ld_position).

      APPEND INITIAL LINE TO lt_modification
      REFERENCE INTO ld_modification_pos.

      ld_modification_pos->node         = zif_kr_kundenrechnung_c=>sc_node-
position.
      ld_modification_pos->change_mode  = /bobf/if_frw_c=>sc_modify_create.
      ld_modification_pos->association  =
zif_kr_kundenrechnung_c=>sc_association-position-to_parent.
      ld_modification_pos->root_key     = lv_kopf_key.
      ld_modification_pos->source_key   = lv_kopf_key.
      ld_modification_pos->source_node  = zif_kr_kundenrechnung_c=>sc_node-
kopf.

      CREATE DATA ld_modification_pos->data TYPE zkr_s_position.
      ASSIGN ld_modification_pos->data->* TO <ls_mod>.
      <ls_mod> = ld_position->*.

      FREE ld_modification_pos.
    ENDLOOP.

    "Über Service-Manager die Modifikation durchführen lassen
    DATA(lo_message) = /bobf/cl_frw_factory=>get_message( ).
    mo_service_manager->modify(
      EXPORTING
        it_modification = lt_modification
      IMPORTING
        eo_message      = lo_message
    ).


    "Objekt erzeugen mit der Rechnungsnummer als Schlüssel
    CREATE OBJECT ro_kundenrechnung
      EXPORTING
        iv_rechnungsnr = is_kopf-rechnungsnr.
  ENDMETHOD.

  METHOD constructor.
    DATA lt_altkey_rechnungsnr  TYPE zkr_t_altkey_rechnum.
    DATA ls_altkey_rechnungsnr  TYPE zkr_s_altkey_rechnum.
    DATA lt_kopf_data           TYPE zkr_t_kopf.
    DATA lt_kopf_key            TYPE /BObf/t_frw_key.


    ls_altkey_rechnungsnr-rechnungsnr = iv_rechnungsnr.

    INSERT ls_altkey_rechnungsnr
      INTO TABLE lt_altkey_rechnungsnr.

    mo_service_manager->convert_altern_key(
      EXPORTING
        iv_node_key          = zif_kr_kundenrechnung_c=>sc_node-position
```

```
        iv_altkey_key        = zif_kr_kundenrechnung_c=>sc_alternative_key-
kopf-rechnungsnr
        it_key               = lt_altkey_rechnungsnr
      IMPORTING
        et_key               = lt_kopf_key
    ).

    mo_service_manager->retrieve(
      EXPORTING
        iv_node_key             = zif_kr_kundenrechnung_c=>sc_node-kopf
        it_key                  = lt_kopf_key
        iv_fill_data            = abap_true
      IMPORTING
        et_data                 = lt_kopf_data
    ).

    ms_kopf = lt_kopf_data[ 1 ].
  ENDMETHOD.


  METHOD pay.
    DATA lt_kopf_key  TYPE /Bobf/t_frw_key.

    lt_kopf_key = VALUE #( ( key = ms_kopf-key ) ).

    mo_service_manager->do_action(
      EXPORTING
        iv_act_key           = zif_kr_kundenrechnung_c=>sc_action-kopf-
a_rechnung_bezahlen
        it_key               = lt_kopf_key
    ).

  ENDMETHOD.
ENDCLASS.

*****************************************************
START-OF-SELECTION.

  "Transaktions- und Servicemanager
  DATA lo_transaction_manager     TYPE REF TO /bobf/if_tra_transaction_mgr.
  DATA lo_kundenrechnung          TYPE REF TO lcl_kundenrechnung_model.
  DATA ls_kopf_data               TYPE zkr_s_kopf.
  DATA lt_position_data           TYPE zkr_t_position.
  DATA ls_position                TYPE zkr_s_position.

* Schritte:
*  1. Transaktionsmanager holen
*  2. Eingegebene Daten in passende Struktur bringen
*  3. Rechnung über Klasse anlegen und Objekt bekommen
*  4. Rechnung auf "Bezahlt" setzen
*  5. Speichern über Transaktionsmanager

  lo_transaction_manager    =
/bobf/cl_tra_trans_mgr_factory=>get_transaction_manager( ).

  ls_kopf_data-rechnungsnr    = pa_rechn.
  ls_kopf_data-kundennr       = pa_kunnr.
  ls_kopf_data-status         = '1'.      "= Offen

  ls_position-artikelnr       = pa_artnr.
  ls_position-menge           = pa_menge.
  ls_position-positionsnr     = '1'.
```

```
  INSERT ls_position
    INTO TABLE lt_position_data.

  "Jetzt das Rechnungsobjekt erzeugen
  lo_kundenrechnung = lcl_kundenrechnung_model=>create_new_kr(
                        is_kopf     = ls_kopf_data
                        it_position = lt_position_data
                      ).

  "Rechnung bezahlen
  lo_kundenrechnung->pay( ).

  "Speichern der Daten
  lo_transaction_manager->save( ).
```

Listing 8.1 – CDS-View-Definition mit ObjectModel-Annotationen

```
@AbapCatalog.sqlViewName: 'zkr_head'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Kopf-Daten Kundenrechnung'

@ObjectModel.compositionRoot: true
@ObjectModel.modelCategory: #BUSINESS_OBJECT
@ObjectModel.writeActivePersistence: 'zkr_d_kopf'
@ObjectModel.transactionalProcessingEnabled: true
@ObjectModel.representativeKey: 'DB_KEY'
@ObjectModel.semanticKey: 'RECHNUNGSNR'
@ObjectModel: { createEnabled: true,
                updateEnabled: true,
                deleteEnabled: true }

@OData.publish: true
define view zkr_kundenrechnung as select from zkr_d_kopf AS k {
  key k.mandt,
  key k.db_key,
  k.rechnungsnr,
  k.kundennr
}
```