INTENSE AG

„Oh man, wer hat DAS denn bitte programmiert?!? Oh, das war ja ich ☹“

Dominik Panzer, INTENSE AG

PanzerDominik

PanzerDominik@sw-development-is.social

Was machen wir heute?

Programmieren ist kein Wettbewerb, wer die cleverste Lösung findet, sondern wer die einfachste und verständlichste findet.

Basic Refactorings

Das Kata „Meterdata Manager“™

```
METHOD manage.
  DATA(mrvalue_as_string) = to_string( meterreadingvalue ).
  CHECK meterreadingvalue_supplied( meterreadingvalue ).
  CHECK vks_supplied( vks ).
  CHECK contract_is_supplied( contract ).
  CHECK meterreadingandvksvalid( mrvalue_as_string = mrvalue_as_string meterreadingvalue = meterreadingvalue vks = vks ).
  DATA(zeabl) = read_All_meterreadings( contract ).
  enhance_all_meterreadings( EXPORTING i_meterreadingvalue = meterreadingvalue
                                       i_date = date
                                       i_contract = contract
                             CHANGING c_zeabl = zeabl ).
  CHECK check_for_overflow( zeabl ).
  DATA(usage) = calculate_usage( zeabl ).
  CHECK no_zero_usage( usage ).
  bill( i_contract = contract i_usage = usage ).
  success = abap_true.
ENDMETHOD.
```

Next Steps:

Namen weiter anpassen

Werte zu Konstanten ändern

Tests aufräumen

Conditionals ggf. prüfen

Value Objects

Ungenutzte Variablen löschen

String-konvertierung extrahieren

Reihenfolge anpassen

Dependecy Injection

Infos und Kontakt

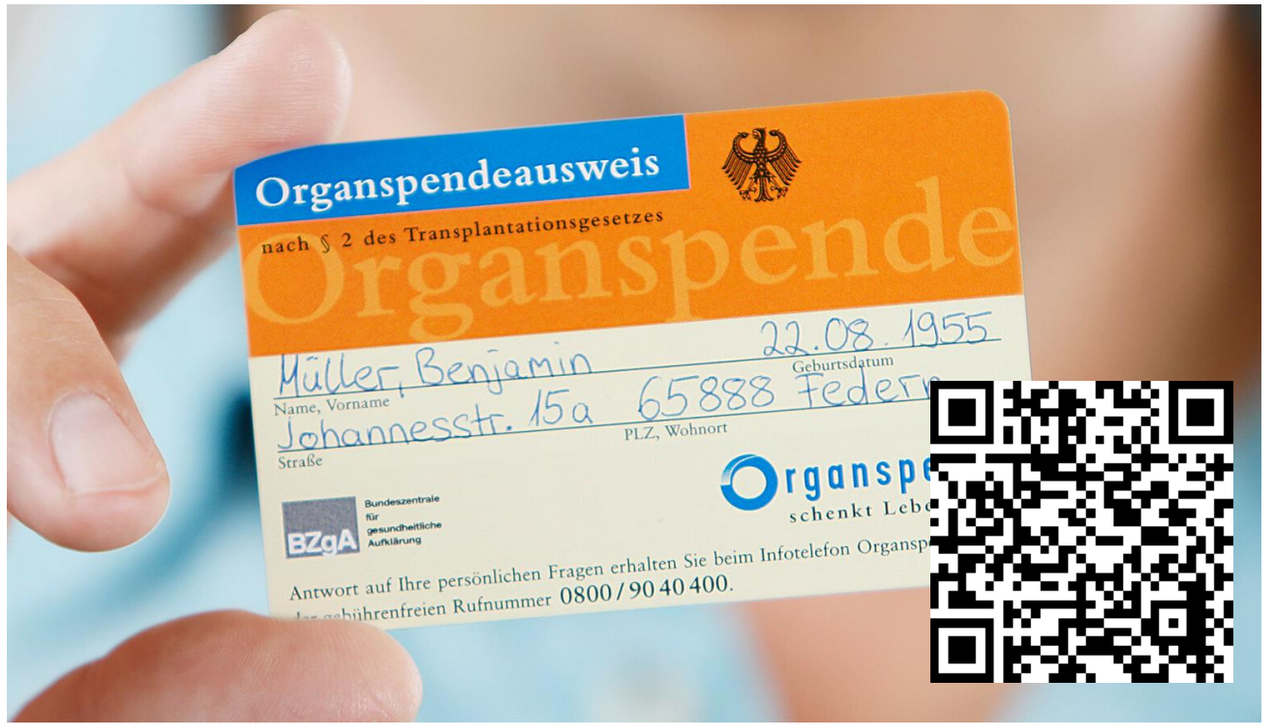
Mehr Infos:

- Clean ABAP
- IOSP
- ABAP quick fix plugin
- Ubiquitous Language
- Legacy of SoCraTes
- Golden Master Testing
- Guard Clause
- Subclass – Extract - Override
- Das Kata auf Github

Dominik.Panzer@intense.de

PanzerDominik

PanzerDominik@sw-development-is.social



```
1  *&---------------------------------------------------------------------*
2  *& Beschreibung:                                                      *
3  *& METERDATA MANAGER                                                  *
4  *&---------------------------------------------------------------------*
5  *& Geändert     Autor      Grund
6  *& Neu          XXXXXXXX   Ersterstellung
7  *& 20120605     XXXXXXXX   nervige Sprachausgabe hinzugefügt
8  *&                          - XXXXXXXX _2012060500
9  *& 20131001     XXXXXXXX   LG zu TL
10 *& 20140320     XXXXXXXX   Versionierung und das Gewirr  hier AUFRÄUMEN!
11 *& 20140804     XXXXXXXX   TICKETNUMMER -> Korrektur
12 *& 20150803     XXXXXXXX   TICKETNUMMER Versand in Hintergrundjob
13 *& 20161012     XXXXXXXX   TICKETNUMMER Integration Befüllungsjob
14 *& 20180821     XXXXXXXX   TICKETNUMMER Sperrkonzept
15 *& 20191219     XXXXXXXX   neue Variante
16 *&                         Blöden 72 Zeichen Feldkatalog rausgeworfen!!
17 *& 20200427     XXXXXXXX   Jobabrüche, Konvertierung TIMESTAMP
18 *&---------------------------------------------------------------------*
19 CLASS zcl_meterdata_manager DEFINITION
20   PUBLIC
21   CREATE PUBLIC .
22
23   PUBLIC SECTION.
24     METHODS manage IMPORTING meterreadingvalue TYPE int4
25                              date               TYPE dats
26                              vks                TYPE int4
27                              contract           TYPE char20
28                    RETURNING VALUE(success)     TYPE abap_bool.
29   PROTECTED SECTION.
30   PRIVATE SECTION.
31 ENDCLASS.
```

```
38  METHOD manage.
39    DATA: customer          TYPE char20,
40          mrvalue_as_number TYPE string.
41    DATA counter TYPE int4.
42    DATA: zeabl TYPE TABLE OF zeabl.
43
44
45    DATA eabl_old TYPE zeabl.
46    DATA allright TYPE abap_Bool.
47
48    mrvalue_as_number = meterreadingvalue.
49    Condense mrvalue_as_number.
50
51  * check if customer is valid
52    IF meterreadingvalue IS NOT INITIAL.
53      IF vks IS NOT INITIAL.
54        IF contract IS INITIAL.
55          success = ''.
56          EXIT.
57        ELSE.
58          IF meterreadingvalue > 0 AND strlen( mrvalue_as_number ) <= vks.
59  * Alle Ablesungen ermitteln
60            SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.
61
62            " hier noch appenden und sotieren
63            SORT zeabl BY adat DESCENDING.
64            zeabl = value #( base zeabl ( ablbelnr = '' adat = date vertrag = contract zaehlerstand = meterreadingvalue ) ).
65  * allright
66            allright = 'X'.
```

```
66            allright = 'X'.
67            LOOP AT zeabl ASSIGNING FIELD-SYMBOL(<eabl_line>).
68              IF eabl_old-adat >= <eabl_line>-adat.
69                allright = ''.
70              ENDIF.
71  Eabl_old = <eabl_line>.
72            ENDLOOP.
73            IF allright = ''.
74              EXIT.
75            ENDIF.
76  * calculate usage
77            DATA(usg) = zeabl[ lines( zeabl ) ]-zaehlerstand - zeabl[ 1 ]-zaehlerstand.
78
79            IF usg > 0.
80
81  *         CALL FUNCTION 'ZBILLIT'
82  *           EXPORTING
83  *             billdate = '99991231'
84  *             usage    = usg_text
85  *             contract = '0593053'.
86  *             success = 'X'.
87              CALL FUNCTION 'ZBILLIT'
88                EXPORTING
89                  billdate = sy-datum
90                  usage    = usg
91                  contract = contract.
92              success = 'X'.
93            ELSE.
94              success = ''.
95            ENDIF.
```

```
96          ELSE.
97            success = ''.
98            IF usg IS INITIAL.
99              success = ''.
100           ENDIF.
101         ENDIF.
102       ENDIF.
103     ELSE.
104       success = ''.
105     ENDIF.
106   ELSE.
107     success = ''.
108   ENDIF.
109 ENDMETHOD.
110 ENDCLASS.
```

```
    mrvalue_as_number = meterreadingvalue.
    Condense mrvalue_as_number.

* check if customer is valid
    IF meterreadingvalue IS NOT INITIAL.
      IF vks IS NOT INITIAL.
        IF contract IS INITIAL.
          success = ''.
          EXIT.
        ELSE.
          OR meterreadingvalue > 0 AND strlen( mrvalue_as_number ) <= vks.
* Alle Ablesungen ermitteln
          SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.

          " hier noch anpassen und sotieren
          SORT zeabl BY adat DESCENDING.
          zeabl = value #( base zeabl ( ablbelnr = '' adat = date vertrag = contract zaehlerstand = meterreadingvalue ) ).
* allright
          allright = 'X'.
          LOOP AT zeabl ASSIGNING FIELD-SYMBOL(<eabl_line>).
            IF eabl_old-adat >= <eabl_line>-adat.
              allright = ''.
            ENDIF.
eabl_old = <eabl_line>.
          ENDLOOP.
          IF allright = ''.
            EXIT.
          ENDIF.
* calculate usage
          DATA(usg) = zeabl[ lines( zeabl ) ]-zaehlerstand - zeabl[ 1 ]-zaehlerstand.

          IF usg > 0.

*         CALL FUNCTION 'ZBILLIT'
*          EXPORTING
*           billdate = '99991231'
*           usage    = usg_text
*           contract = '0593053'.
*           success = 'X'.
            CALL FUNCTION 'ZBILLIT'
             EXPORTING
               billdate = sy-datum
               usage    = usg
               contract = contract.
            success = 'X'.
          ELSE.
            success = ''.
          ENDIF.
        ELSE.
          success = ''.
          IF usg IS INITIAL.
            success = ''.
          ENDIF.
        ENDIF.
      ENDIF.
    ELSE.
      success = ''.
    ENDIF.
  ELSE.
```

```
20   METHOD manage.
21     DATA: mrvalue_as_number TYPE string.
22     DATA: zeabl TYPE TABLE OF zeabl.
23     DATA eabl_old TYPE zeabl.
24     DATA allright TYPE abap_Bool.
25
26     mrvalue_as_number = meterreadingvalue.
27     CONDENSE mrvalue_as_number.
28
```

```
20   METHOD manage.
21     DATA: mrvalue_as_string TYPE string.
22     DATA: zeabl TYPE TABLE OF zeabl.
23     DATA eabl_old TYPE zeabl.
24     DATA allright TYPE abap_Bool.
25
26     mrvalue_as_string = meterreadingvalue.
27     CONDENSE mrvalue_as_string.
28
```

```
55          DATA(usg) = zeabl[ lines( zeabl ) ]-zaehlerstand - zeabl[ 1 ]-zaehlerstand.
56
57          IF usg > 0.
58            CALL FUNCTION 'ZBILLIT'
59              EXPORTING
60                billdate = sy-datum
61                usage    = usg
62                contract = contract.
63            success = 'X'.
64          ELSE.
65            success = ''.
66          ENDIF.
```

```
62            DATA(usage) = calculate_usage( zeabl ).
63
64            IF usage > 0.
65              CALL FUNCTION 'ZBILLIT'
66                EXPORTING
67                  billdate = sy-datum
68                  usage    = usage
69                  contract = contract.
70              success = 'X'.
71            ELSE.
72              success = ''.
73            ENDIF.
```

```
29  * check if customer is valid
30      IF meterreadingvalue IS NOT INITIAL.
31        IF vks IS NOT INITIAL.
32          IF contract IS INITIAL.
33            success = ''.
34            EXIT.
35          ELSE.
```

```
72            ENDIF.
73          ENDIF.
74        ELSE.
75          success = ''.
76        ENDIF.
77      ELSE.
78        success = ''.
79      ENDIF.
80    ENDMETHOD.
81  ENDCLASS.
```

```
CLASS meterdata_manager_Test  DEFINITION FINAL FOR TESTING
  DURATION SHORT
  RISK LEVEL HARMLESS.

  PRIVATE SECTION.
    METHODS:
      meteterreadingvaluenotsupplied FOR TESTING RAISING cx_static_check.
ENDCLASS.


CLASS meterdata_manager_Test IMPLEMENTATION.

  METHOD meteterreadingvaluenotsupplied.
    DATA: meterreadingvalue TYPE int4,
          date              TYPE dats,
          vks               TYPE int4,
          contract          TYPE char20.
* arrange
    DATA(meterdata_manager) = NEW zcl_meterdata_manager( ).

* act
    DATA(result) = meterdata_manager->manage(
                     meterreadingvalue = meterreadingvalue
                     date              =  date
                     vks               = vks
                     contract          = contract
                   ).

* assert
cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).

  ENDMETHOD
```

```
26  METHOD manage.
27    DATA: mrvalue_as_string TYPE string.
28    DATA: zeabl TYPE TABLE OF zeabl.
29    DATA eabl_old TYPE zeabl.
30    DATA allright TYPE abap_Bool.
31
32    mrvalue_as_string = meterreadingvalue.
33    CONDENSE mrvalue_as_string.
34
35  * check if customer is valid
36    IF meterreadingvalue IS NOT INITIAL.
37      IF vks IS NOT INITIAL.
38        IF contract IS INITIAL.
39          success = ''.
40          EXIT.
41        ELSE.
42          IF meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks.
43  * Alle Ablesungen ermitteln
```

```
85          success = ''.
86      ENDIF.
87    ENDMETHOD.
```

```
35    IF meterreadingvalue IS INITIAL.
36      success = ''.
37      EXIT.
38    ENDIF.
```

```
19    METHODS meterreadingvalue_supplied
20      IMPORTING
21        meterreadingvalue TYPE int4
22      RETURNING
23        value(success)    TYPE abap_bool.
```

```
90  METHOD meterreadingvalue_supplied.
91    IF meterreadingvalue IS NOT INITIAL.
92      success = abap_True.
93    ENDIF.
94  ENDMETHOD.
```

```
31  METHOD manage.
32    DATA: mrvalue_as_string TYPE string.
33    DATA: zeabl TYPE TABLE OF zeabl.
34    DATA eabl_old TYPE zeabl.
35    DATA allright TYPE abap_Bool.
36
37    mrvalue_as_string = meterreadingvalue.
38    CONDENSE mrvalue_as_string.
39
40    CHECK meterreadingvalue_supplied( meterreadingvalue )
41    IF vks IS NOT INITIAL.
42      IF contract IS INITIAL.
43        success = ''.
44        EXIT.
45      ELSE.
```

```
41    IF vks IS NOT INITIAL.
42      IF contract IS INITIAL.
43        success = ''.
44        EXIT.
45      ELSE.
46        IF meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks.
47* Alle Ablesungen ermitteln
48          SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl
49
```

```
82          ENDIF.
83        ENDIF.
84      ENDIF.
85    ELSE.
86      success = ''.
87    ENDIF.
88  ENDMETHOD.
```

```
35   METHOD vksnotsupplied.
36     DATA: meterreadingvalue TYPE int4,
37           date              TYPE dats,
38           vks               TYPE int4,
39           contract          TYPE char20.
40 * arrange
41     DATA(meterdata_manager) = NEW zcl_meterdata_manager( ).
42     meterreadingvalue = 10.
43
44 * act
45     DATA(result) = meterdata_manager->manage(
46                      meterreadingvalue = meterreadingvalue
47                      date              =  date
48                      vks               = vks
49                      contract          = contract
50                    ).
51
52 * assert
53 cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).
54
55   ENDMETHOD.
```

```
1  CLASS meterdata_manager_Test  DEFINITION FINAL FOR TESTING
2    DURATION SHORT
3    RISK LEVEL HARMLESS.
4
5    PRIVATE SECTION.
6      DATA: meterreadingvalue TYPE int4,
7            date              TYPE dats,
8            vks               TYPE int4,
9            contract          TYPE char20,
10           meterdata_manager TYPE REF TO zcl_meterdata_manager.
11     METHODS:
12       meteterreadingvaluenotsupplied FOR TESTING RAISING cx_static_check,
13       vksnotsupplied FOR TESTING,
14       setup.
15 ENDCLASS.
16
17
18 CLASS meterdata_manager_Test IMPLEMENTATION.
19
20   METHOD setup.
21     meterdata_manager = NEW zcl_meterdata_manager( ).
22   ENDMETHOD.
```

```
31  METHOD manage.
32    DATA: mrvalue_as_string TYPE string.
33    DATA: zeabl TYPE TABLE OF zeabl.
34    DATA eabl_old TYPE zeabl.
35    DATA allright TYPE abap_Bool.
36
37    mrvalue_as_string = meterreadingvalue.
38    CONDENSE mrvalue_as_string.
39
40    CHECK meterreadingvalue_supplied( meterreadingvalue ).
41    IF vks IS NOT INITIAL.
42      IF contract IS INITIAL.
43        success = ''.
44        EXIT.
45      ELSE.
```

```
86        success = ''.
87      ENDIF.
88    ENDMETHOD.
```

```
41
42      mrvalue_as_string = meterreadingvalue.
43      CONDENSE mrvalue_as_string.
44
45      CHECK meterreadingvalue_supplied( meterreadingvalue ).
46      CHECK vks_supplied( vks ).
47      IF contract IS INITIAL.
48        success = ''.
49        EXIT.
50      ELSE.
```

```
92    METHOD vks_supplied.
93      IF vks IS NOT INITIAL.
94        success = abap_true.
95      ENDIF.
```

```
45      CHECK meterreadingvalue_supplied( meterreadingvalue ).
46      CHECK vks_supplied( vks ).
47      IF contract IS INITIAL.
48        success = ''.
49        EXIT.
50      ELSE.
51        IF meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks.
52  * Alle Ablesungen ermitteln
53          SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.
54
```

```
59   method contractnotsupplied.
60 * arrange
61     meterreadingvalue = 10.
62     vks = 1.
63
64 * act
65     DATA(result) = meterdata_manager->manage(
66                      meterreadingvalue = meterreadingvalue
67                      date              = date
68                      vks               = vks
69                      contract          = contract
70                    ).
71
72 * assert
73     cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).
74
75   ENDMETHOD.
```

```
42      mrvalue_as_string = meterreadingvalue.
43      CONDENSE mrvalue_as_string.
44
45      CHECK meterreadingvalue_supplied( meterreadingvalue ).
46      CHECK vks_supplied( vks ).
47      IF contract IS INITIAL.
48        success = ''.
49        EXIT.
50      ELSE.
51        IF meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks.
52  * Alle Ablesungen ermitteln
53          SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl
```

```
93   METHOD contract_is_supplied.
94
95     IF contract IS not INITIAL.
96       success = abap_true.
97     ENDIF.
98
99   ENDMETHOD.
100
```

```
47     mrvalue_as_string = meterreadingvalue.
48     CONDENSE mrvalue_as_string.
49
50     CHECK meterreadingvalue_supplied( meterreadingvalue ).
51     CHECK vks_supplied( vks ).
52     CHECK contract_is_supplied( contract ).
53     IF meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks.
54   * Alle Ablesungen ermitteln
55       SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.
56
```

```
78     method meterreadingandvksinvalid.
79       CONSTANTS random_contract_number TYPE char20 VALUE '121245252'.
80 * arrange
81     meterreadingvalue = 10.
82     vks = 1.
83     contract = random_contract_number.
84
85 * act
86     DATA(result) = meterdata_manager->manage(
87                      meterreadingvalue = meterreadingvalue
88                      date              = date
89                      vks               = vks
90                      contract          = contract
91                    ).
92
93 * assert
94     cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).
95
96   ENDMETHOD.
```

```
47      mrvalue_as_string = meterreadingvalue.
48      CONDENSE mrvalue_as_string.
49
50      CHECK meterreadingvalue_supplied( meterreadingvalue ).
51      CHECK vks_supplied( vks ).
52      CHECK contract_is_supplied( contract ).
53      IF meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks.
54  * Alle Ablesungen ermitteln
55        SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.
```

```
85      ELSE.
86        success = ''.
87        IF usage IS INITIAL.
88          success = ''.
89        ENDIF.
90      ENDIF.
91    ENDMETHOD.
```

```
54      result = xsdbool( meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks ).
55      IF result = abap_true.
56  * Alle Ablesungen ermitteln
57        SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.
58
```

```
99    METHOD meterreadingandvksvalid.
100
101     r_result = xsdbool( meterreadingvalue > 0 AND strlen( mrvalue_as_string ) <= vks ).
102
103   ENDMETHOD.
```

```
55      mrvalue_as_string = meterreadingvalue.
56      CONDENSE mrvalue_as_string.
57
58      CHECK meterreadingvalue_supplied( meterreadingvalue ).
59      CHECK vks_supplied( vks ).
60      CHECK contract_is_supplied( contract ).
61      CHECK meterreadingandvksvalid( mrvalue_as_string = mrvalue_as_string meterreadingvalue = meterreadingvalue vks = vk
62  * Alle Ablesungen ermitteln
63        SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.
64
```

```
55      mrvalue_as_string = meterreadingvalue.
56      CONDENSE mrvalue_as_string.
57
58      CHECK meterreadingvalue_supplied( meterreadingvalue ).
59      CHECK vks_supplied( vks ).
60      CHECK contract_is_supplied( contract ).
61      CHECK meterreadingandvksvalid( mrvalue_as_string = mrvalue_as_string meterreadingvalue = meterreadingvalue vks = vks ).
62  * Alle Ablesungen ermitteln
63        SELECT * FROM zeabl WHERE vertrag = @contract INTO CORRESPONDING FIELDS OF TABLE @zeabl.
64
65        " hier noch appenden und sotieren
66        SORT zeabl BY adat DESCENDING.
67        zeabl = VALUE #( BASE zeabl ( ablbelnr = '' adat = date vertrag = contract zaehlerstand = meterreadingvalue ) ).
```

```
61      mrvalue_as_string = meterreadingvalue.
62      CONDENSE mrvalue_as_string.
63
64      CHECK meterreadingvalue_supplied( meterreadingvalue ).
65      CHECK vks_supplied( vks ).
66      CHECK contract_is_supplied( contract ).
67      CHECK meterreadingandvksvalid( mrvalue_as_string = mrvalue_as_string meterreadingvalue = meterreadingvalue vks = vks ).
68      zeabl = read_All_meterreadings( contract ).
69
70      " hier noch appenden und sotieren
71      SORT zeabl BY adat DESCENDING.
72      zeabl = VALUE #( BASE zeabl ( ablbelnr = '' adat = date vertrag = contract zaehlerstand = meterreadingvalue ) ).
73  * allright
74      allright = 'X'.
75      LOOP AT zeabl ASSIGNING FIELD-SYMBOL(<eabl_line>).
76        IF eabl_old-zaehlerstand > <eabl_line>-zaehlerstand or lines( zeabl ) <> 12.
77          allright = ''.
78        ENDIF.
79        Eabl_old = <eabl_line>.
80      ENDLOOP.
81      IF allright = ''.
82        EXIT.
83      ENDIF.
84
```

```
121  METHOD enhance_all_meterreadings.
122
123    SORT c_zeabl BY adat DESCENDING.
124    c_zeabl = VALUE #( BASE c_zeabl ( ablbelnr = '' adat = i_date vertrag = i_contract zaehlerstand = i_meterreadingvalue ) ).
125
126  ENDMETHOD.
```

```
103  METHOD check_for_overflow.
104    CONSTANTS _12_monate TYPE i VALUE 12.
105
106    DATA eabl_old TYPE zeabl.
107    DATA allright TYPE abap_bool.
108
109    result = abap_true.
110    LOOP AT c_zeabl ASSIGNING FIELD-SYMBOL(<eabl_line>).
111      IF eabl_old-zaehlerstand > <eabl_line>-zaehlerstand OR lines( c_zeabl ) <> _12_monate.
112        result = ''.
113      ENDIF.
114      Eabl_old = <eabl_line>.
115    ENDLOOP.
116
117  ENDMETHOD.
```

```
99      method meterreadingoverflow.
100       CONSTANTS random_contract_number TYPE char20 VALUE '121245252'.
101   * arrange
102     meterreadingvalue = 10.
103     vks = 2.
104     contract = random_contract_number.
105
106   * act
107     DATA(result) = meterdata_manager->manage(
108                      meterreadingvalue = meterreadingvalue
109                      date              = date
110                      vks               = vks
111                      contract          = contract
112                    ).
113
114   * assert
115     cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).
116
```

```
75      mrvalue_as_string = meterreadingvalue.
76      CONDENSE mrvalue_as_string.
77
78      CHECK meterreadingvalue_supplied( meterreadingvalue ).
79      CHECK vks_supplied( vks ).
80      CHECK contract_is_supplied( contract ).
81      CHECK meterreadingandvksvalid( mrvalue_as_string = mrvalue_as_string meterreadingvalue = meterreadingvalue vks = vks ).
82      zeabl = read_All_meterreadings( contract ).
83
84      enhance_all_meterreadings( EXPORTING i_meterreadingvalue = meterreadingvalue
85                                           i_date = date
86                                           i_contract = contract
87                                 CHANGING c_zeabl = zeabl ).
88      CHECK check_for_overflow( zeabl ).
89      DATA(usage) = calculate_usage( zeabl ).
```

find pattern 11 rows retrieved - 20 ms

MANDT	ABLBELNR	ADAT	ZAEHLERSTAND	VERTRAG
100	1.001	2021-01-01	100	1004254252
100	1.002	2021-02-01	110	1004254252
100	1.003	2021-03-01	115	1004254252
100	1.004	2021-04-01	127	1004254252
100	1.005	2021-05-01	200	1004254252
100	1.006	2021-06-01	201	1004254252
100	1.007	2021-07-01	234	1004254252
100	1.008	2021-08-01	234	1004254252
100	1.009	2021-09-01	300	1004254252
100	1.010	2021-10-01	310	1004254252
100	1.011	2021-11-01	330	1004254252

```
134 * arrange
135     meterreadingvalue = 310.
136     vks = 3.
137     contract = '1004254252'.
138     date = 20211201.
139
140 * act
141     DATA(result) = meterdata_manager->manage(
142                      meterreadingvalue = meterreadingvalue
143                      date              = date
144                      vks               = vks
145                      contract          = contract
146                    ).
147
148 * assert
149     cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).
150
151 endmethod.
```

```
1  CLASS meterdata_manager_Testable DEFINITION INHERITING FROM zcl_meterdata_manager.
2    PUBLIC SECTION.
3      METHODS constructor IMPORTING all_meterreadings TYPE ztt_zeabl.
4
5    PROTECTED SECTION.
6      METHODS: read_all_meterreadings REDEFINITION.
7      DATA: all_meterreadings TYPE ztt_zeabl.
8
9  ENDCLASS.
10
11 CLASS meterdata_manager_testable IMPLEMENTATION.
12
13   METHOD constructor.
14     super->constructor( ).
15     me->all_meterreadings = all_meterreadings.
16   ENDMETHOD.
17   METHOD read_all_meterreadings.
18     r_zeabl =  all_meterreadings.
19   ENDMETHOD.
20 ENDCLASS.
```

```
183   METHOD get_no_usage_meterreadings.
184
185     ro_usage_meterreadings  =  VALUE ztt_zeabl( ( mandt ='100' ablbelnr ='1001' adat ='20210101' zaehlerst
186 ( mandt ='100' ablbelnr ='1002' adat ='20210201' zaehlerstand ='100' vertrag ='1004254252'  )
187 ( mandt ='100' ablbelnr ='1003' adat ='20210301' zaehlerstand ='100' vertrag ='1004254252'  )
188 ( mandt ='100' ablbelnr ='1004' adat ='20210401' zaehlerstand ='100' vertrag ='1004254252'  )
189 ( mandt ='100' ablbelnr ='1005' adat ='20210501' zaehlerstand ='100' vertrag ='1004254252'  )
190 ( mandt ='100' ablbelnr ='1006' adat ='20210601' zaehlerstand ='100' vertrag ='1004254252'  )
191 ( mandt ='100' ablbelnr ='1007' adat ='20210701' zaehlerstand ='100' vertrag ='1004254252'  )
192 ( mandt ='100' ablbelnr ='1008' adat ='20210801' zaehlerstand ='100' vertrag ='1004254252'  )
193 ( mandt ='100' ablbelnr ='1009' adat ='20210901' zaehlerstand ='100' vertrag ='1004254252'  )
194 ( mandt ='100' ablbelnr ='1010' adat ='20211001' zaehlerstand ='100' vertrag ='1004254252'  )
195 ( mandt ='100' ablbelnr ='1011' adat ='20211101' zaehlerstand ='100' vertrag ='1004254252'  )
196 ).
197
198   ENDMETHOD.
```

```
157   METHOD zerousageerror.
158
159 * arrange
160     DATA(no_usage_meterreadings) = get_no_usage_meterreadings( ).
161     DATA(meterdata_manager) = NEW meterdata_manager_testable( no_usage_meterreadings ).
162
163     meterreadingvalue = 100.
164     vks = 3.
165     contract = '1004254252'.
166     date = 20211201.
167
168 * act
169     DATA(result) = meterdata_manager->manage(
170                      meterreadingvalue = meterreadingvalue
171                      date              = date
172                      vks               = vks
173                      contract          = contract
174                    ).
175
176 * assert
177     cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).
178
179   ENDMETHOD.
```

```
85      enhance_all_meterreadings( EXPORTING i_meterreadingvalue = meterreadingvalue
86                                           i_date = date
87                                           i_contract = contract
88                                 CHANGING c_zeabl = zeabl ).
89      CHECK check_for_overflow( zeabl ).
90      DATA(usage) = calculate_usage( zeabl ).
91
92      IF usage > 0.
93        CALL FUNCTION 'ZBILLIT'
94          EXPORTING
95            billdate = sy-datum
96            usage    = usage
97            contract = contract.
98        success = 'X'.
99      ELSE.
100       success = ''.
101     ENDIF.
102   ENDMETHOD.
```

```
90      enhance_all_meterreadings( EXPORTING i_meterreadingvalue = meterreadingvalue
91                                           i_date = date
92                                           i_contract = contract
93                                 CHANGING c_zeabl = zeabl ).
94      CHECK check_for_overflow( zeabl ).
95      DATA(usage) = calculate_usage( zeabl ).
96      CHECK no_zero_usage( usage ).
97      CALL FUNCTION 'ZBILLIT'
98        EXPORTING
99          billdate = sy-datum
100         usage    = usage
101         contract = contract.
102     success = 'X'.
103   ENDMETHOD.
```

```
185     METHOD successfull_billing.
186
187   * arrange
188     DATA(valid_meterreadings) = get_valid_meterreadings( ).
189     DATA(meterdata_manager) = NEW meterdata_manager_testable( valid_meterreadings ).
190
191     meterreadingvalue = 200.
192     vks = 3.
193     contract = '1004254252'.
194     date = 20211201.
195
196   * act
197     DATA(result) = meterdata_manager->manage(
198                      meterreadingvalue = meterreadingvalue
199                      date              = date
200                      vks               = vks
201                      contract          = contract
202                    ).
203
204   * assert
205     cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = '' act = result ).
206
207   ENDMETHOD.
```

```
 98      CHECK check_for_overflow( zeabl ).
 99      DATA(usage) = calculate_usage( zeabl ).
100      CHECK no_zero_usage( usage ).
101      bill( i_contract = contract i_usage = usage ).
102      success = abap_true.
103    ENDMETHOD.
104
105    METHOD bill.
106
107      CALL FUNCTION 'ZBILLIT'
108        EXPORTING
109          billdate = sy-datum
110          usage    = i_usage
111          contract = i_contract.
112
113    ENDMETHOD.
114
```

```
1  CLASS meterdata_manager_Testable DEFINITION INHERITING FROM zcl_meterdata_manager.
2    PUBLIC SECTION.
3      METHODS constructor IMPORTING all_meterreadings TYPE ztt_zeabl.
4
5    PROTECTED SECTION.
6      METHODS: read_all_meterreadings REDEFINITION.
7      methods: bill REDEFINITION.
8      DATA: all_meterreadings TYPE ztt_zeabl.
9
10 ENDCLASS.
11
12 CLASS meterdata_manager_testable IMPLEMENTATION.
13
14   METHOD constructor.
15     super->constructor( ).
16     me->all_meterreadings = all_meterreadings.
17   ENDMETHOD.
18   METHOD read_all_meterreadings.
19     r_zeabl =  all_meterreadings.
20   ENDMETHOD.
21   METHOD bill.
22
23   ENDMETHOD.
24
25 ENDCLASS.
```

```
185    METHOD successfull_billing.
186
187  * arrange
188      DATA(valid_meterreadings) = get_valid_meterreadings( ).
189      DATA(meterdata_manager) = NEW meterdata_manager_testable( valid_meterreadings ).
190
191      meterreadingvalue = 200.
192      vks = 3.
193      contract = '1004254252'.
194      date = '20211201'.
195
196  * act
197      DATA(result) = meterdata_manager->manage(
198                       meterreadingvalue = meterreadingvalue
199                       date              = date
200                       vks               = vks
201                       contract          = contract
202                     ).
203
204  * assert
205      cl_abap_unit_assert=>assert_equals( msg = 'Fehler' exp = 'X' act = result ).
206
207    ENDMETHOD.
```

```
89      CHECK meterreadingvalue_supplied( meterreadingvalue ).
90      CHECK vks_supplied( vks ).
91      CHECK contract_is_supplied( contract ).
92      CHECK meterreadingandvksvalid( mrvalue_as_string = mrvalue_as_string meterreadingvalue = meterreadingvalue vks = vks ).
93      zeabl = read_All_meterreadings( contract ).
94
95      enhance_all_meterreadings( EXPORTING i_meterreadingvalue = meterreadingvalue
96                                           i_date = date
97                                           i_contract = contract
98                                 CHANGING c_zeabl = zeabl ).
99      CHECK check_for_overflow( zeabl ).
100     DATA(usage) = calculate_usage( zeabl ).
101     CHECK no_zero_usage( usage ).
102     bill( i_contract = contract i_usage = usage ).
103     success = abap_true.
104   ENDMETHOD.
```