www.abapconf.org | #abapconf

December 5th 2024

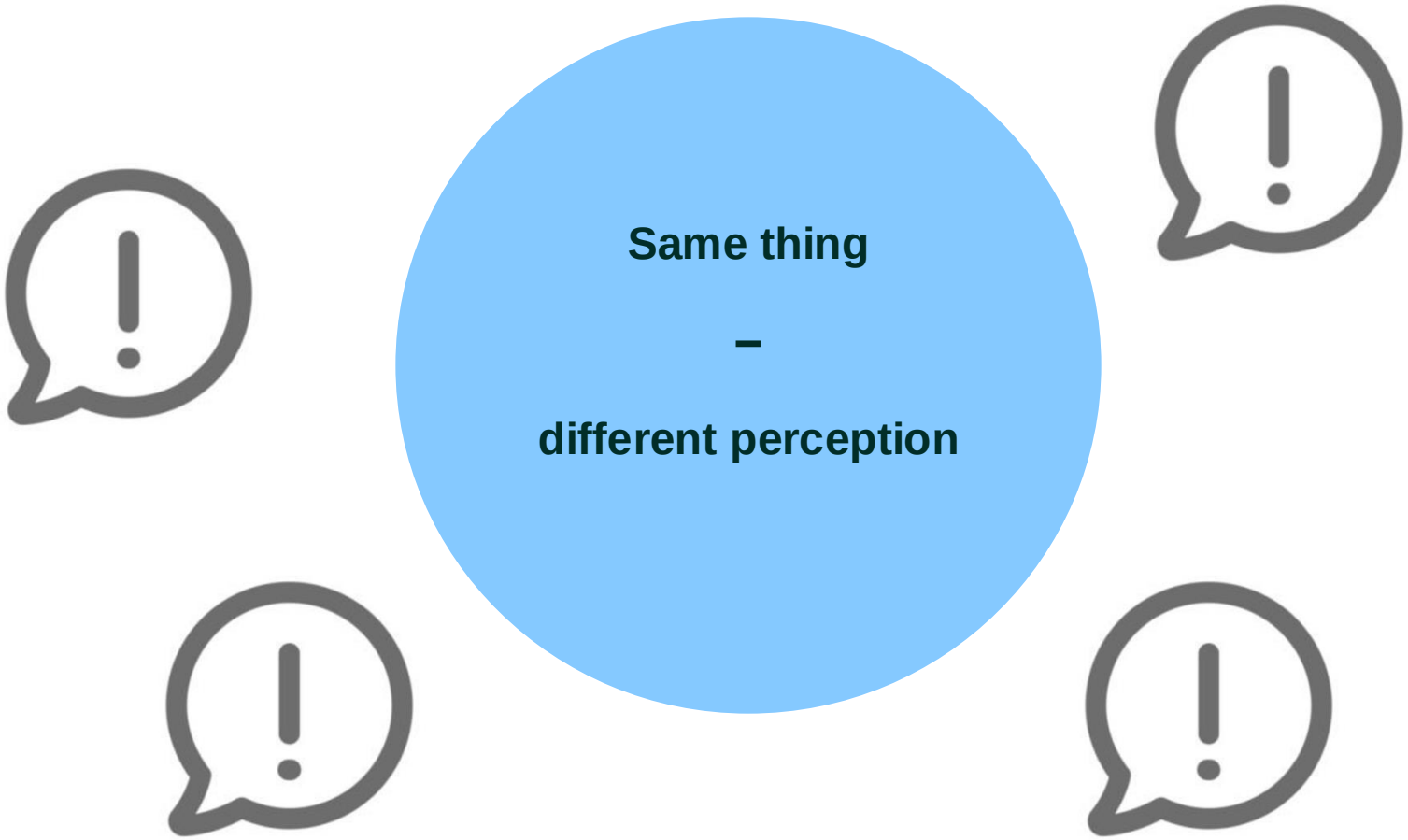
abap conf2024

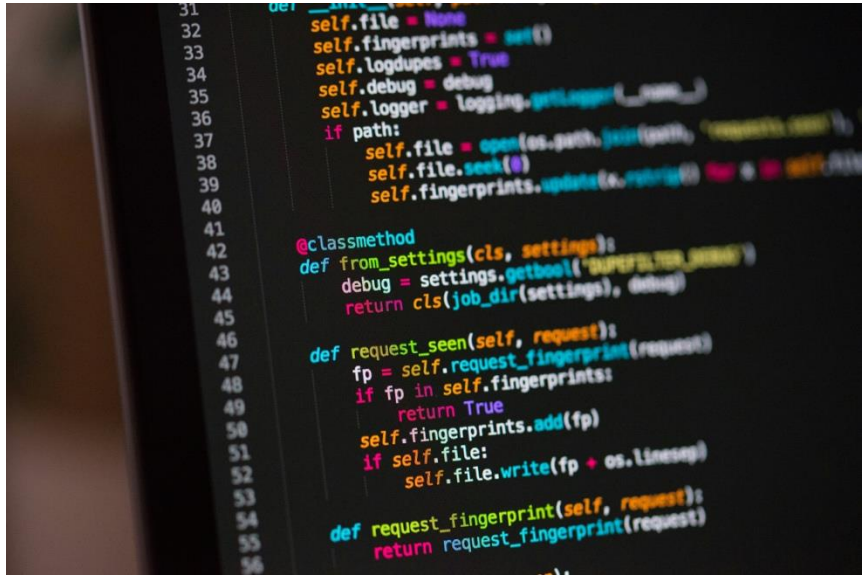
Unit Test your Business Objects in RAP

Martin Jonen

13:30 – 14:20 CET
Channel 1
English

What is your first association with this blue dot?





Three opinions
(Pair programming)

«Power of three»

Three different
solution designs

Martin Jonen

SAP Consultant & Developer | SAP Clean Code Coach

My journey so far …

- Freelanced **SAP Consultant & Developer** (+14 years)
- Certified **SAP ABAP Clean Code Coach**
  (Collaboration with Damir Majer & Prof. Dr. Christian Drumm)
- **Author** (TDD course with Espresso Tutorials)
- Founder of the “**ABAP Conscious Clean Coder Hub**”
- **Hobbies**: Community, Running, Yoga

ASE Coaching

www.abapconf.org | #abapconf

December 5th 2024

abap conf2024

Unit Test your Business Objects in RAP

Martin Jonen

13:30 – 14:20 CET
Channel 1
English

At the end of this presentation

... You understood …

- The **UNIT Test framework** in ABAP OO
- How to write **UNIT tests** for “Business Behaviors” in **Test After Development** style
- How to **mock ENTITY (READ) access** by example
- How to use **clean code principles** to structure your code

How do we get there?

... Without studying tons of theoretical stuff up front!

- By using an example of ***SAP Steampunk*** by Christian Drumm
- By creating test cases on base of existing coding (**Test After Development**)
- By using SAP standard classes to **mock ENTITY access** to decouple dependencies on DB data

What is this not about!

... sorry for that!

- No TDD, but we part of the **TDD cycle** (hell, yeah!!!)
- No mocking of Business Object itself
- No ENTITY (UPDATE) mock examples
- No RAP or OO beginners workshop

Upcoming (Olé!)

Now let’s dive into it …

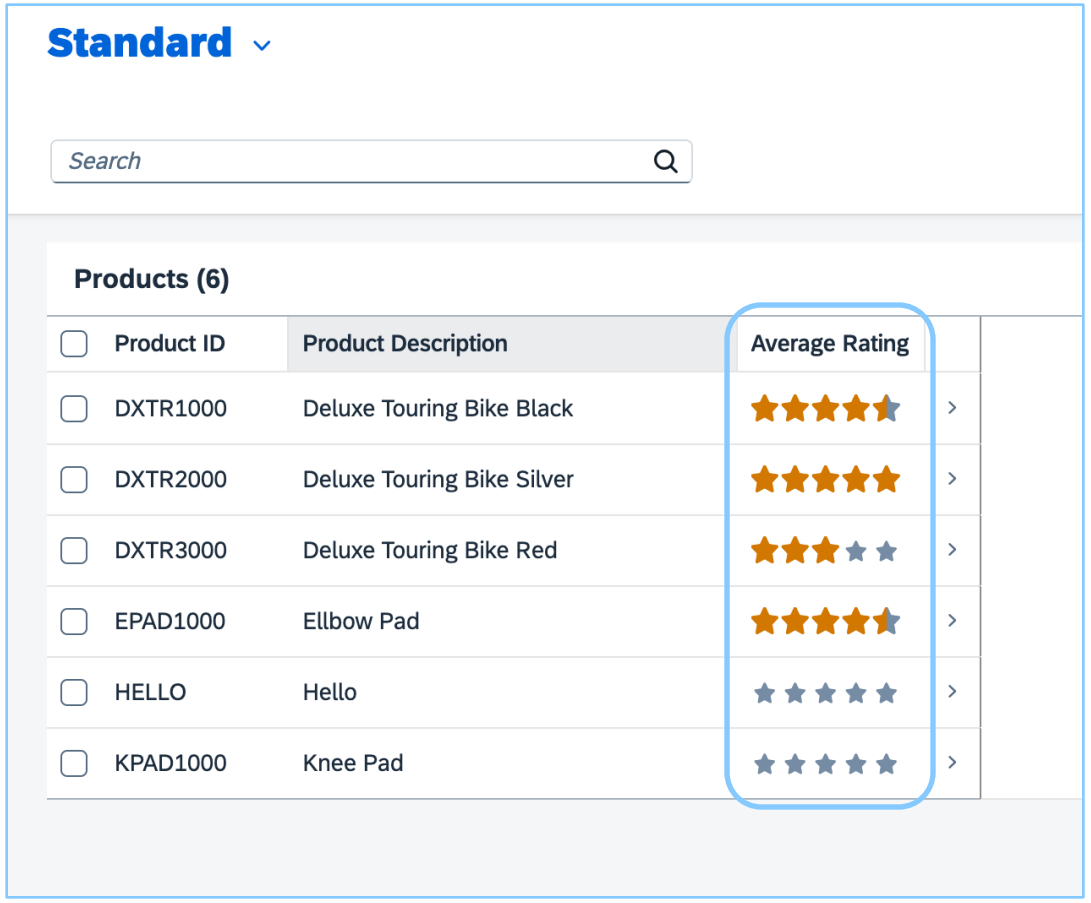
Our setup

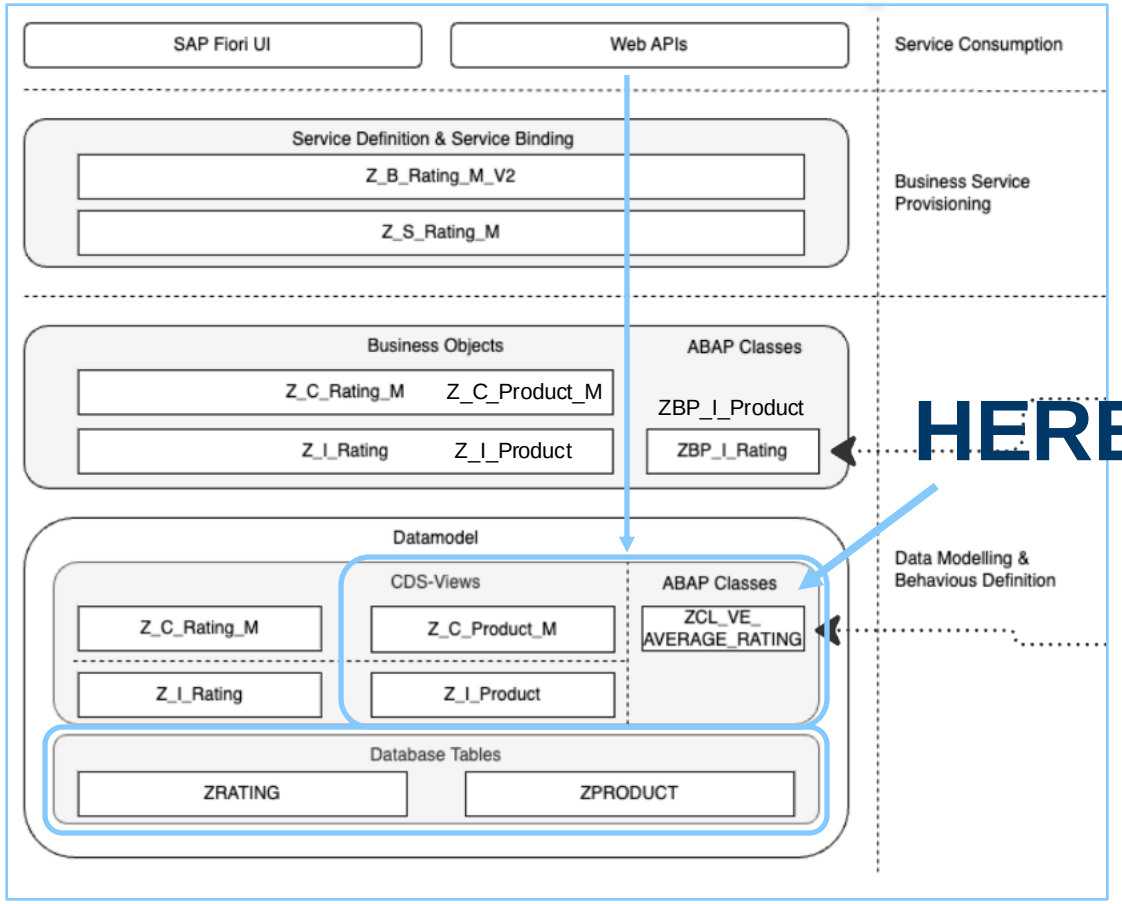
… Thank goodness we don't have to build everything ourselves!

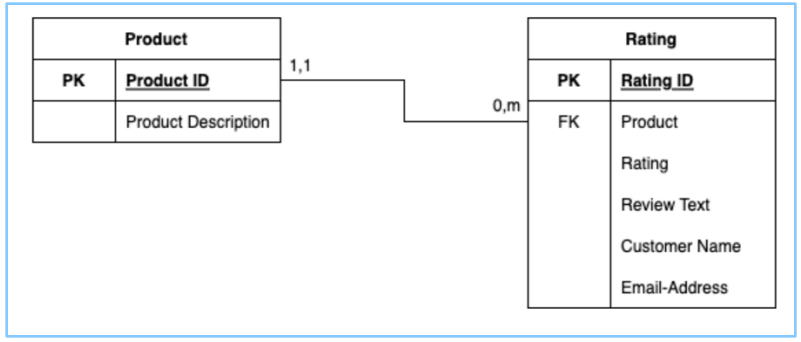
- Espresso Tutorials – **ABAP Steampunk** (https://t.ly/jJljD)
- GitHub Repo "modern-abap-curriculum" (https://github.com/ceedee666/modern-abap-curriculum.git)
  - Clone the Repo (Cloud based – use ABAP trial environment)
  - Get started!

Showcase – Calculate average rating

… I like those ellbow pads!







- **Business logic class**

ZCL_VE_AVERAGE_RATING_MJ

```
1  CLASS zcl_ve_average_rating_mj DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7      INTERFACES if_sadl_exit_calc_element_read .
8
```

ZCL_VE_AVERAGE_RATING_MJ ▸ IF_SADL_EXIT_CALC_ELEMENT_READ~CALCULATE

```
17
18   METHOD if_sadl_exit_calc_element_read~calculate.
19     DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
20     products = CORRESPONDING #( it_original_data ).
21
22     " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
23     READ ENTITIES OF Z_C_Product_M_MJ
24       ENTITY Product BY \_Rating
25       FIELDS ( Rating )
26       WITH CORRESPONDING #( products )
27       RESULT FINAL(ratings).
28
29     " Calculate the average rating for each product by DB select
30     SELECT Product,
31            AVG( rating AS DEC( 2, 1 ) ) AS average_rating
32       FROM @ratings AS r
33       GROUP BY Product
34       INTO TABLE @DATA(average_product_ratings).
35
36     " Map average ratings to the products (output)
37     LOOP AT products ASSIGNING FIELD-SYMBOL(<product>).
38       READ TABLE average_product_ratings
39         WITH KEY Product = <product>-ProductId
40         INTO DATA(average_product_rating).
41       IF sy-subrc = 0.
42         <product>-AverageRating = average_product_rating-average_rating.
43       ENDIF.
44     ENDLOOP.
45
46     ct_calculated_data = CORRESPONDING #( products ).
47   ENDMETHOD.
48
```

- **Business logic class**

ZCL_VE_AVERAGE_RATING_MJ

```
1  CLASS zcl_ve_average_rating_mj DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7      INTERFACES if_sadl_exit_calc_element_read .
8
```

ZCL_VE_AVERAGE_RATING_MJ ▸ IF_SADL_EXIT_CALC_ELEMENT_READ~CALCULATE

```
17
18   METHOD if_sadl_exit_calc_element_read~calculate.
19     DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
20     products = CORRESPONDING #( it_original_data ).
21
22     " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
23     READ ENTITIES OF Z_C_Product_M_MJ
24       ENTITY Product BY \_Rating
25       FIELDS ( Rating )
26       WITH CORRESPONDING #( products )
27       RESULT FINAL(ratings).
28
```

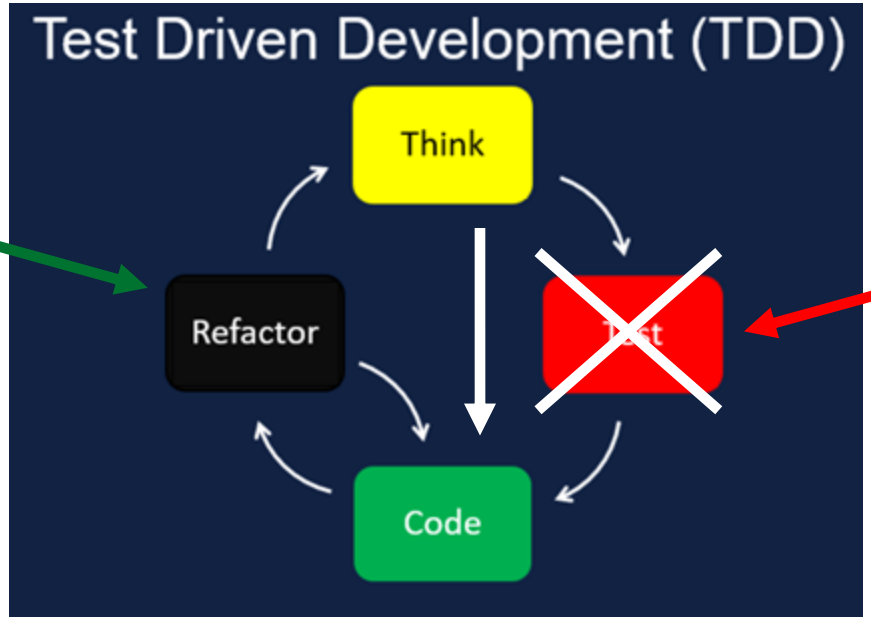
```
11 @Metadata.allowExtensions: true
12 define root view entity Z_C_Product_M_MJ
13   as projection on Z_I_Product_MJ
14 {
15   key ProductId,
16       ProductDescription,
17
18       @ObjectModel.virtualElementCalculatedBy: 'ABAP:ZCL_VE_AVERAGE_RATING_MJ'
19       @EndUserText.label: 'Average Customer Rating'
20       virtual AverageRating: abap.dec( 2, 1 ),
21
22       /* Associations */
23       _Rating : redirected to composition child Z_C_RATING_M_MJ
24 }
25
```

How to test legacy code?

... everyone likes to code clean!

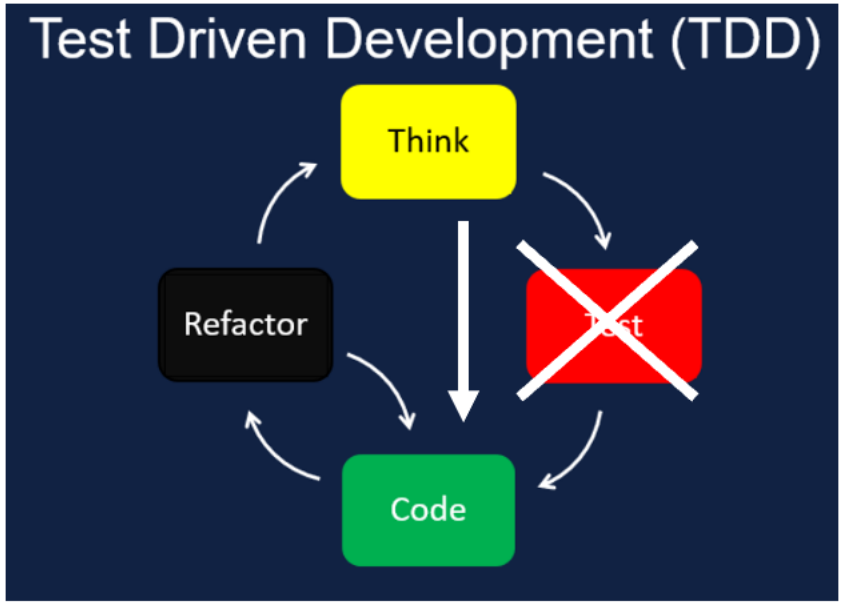
- **T**est **A**fter **D**evelopment

Here, the magic happens!



Whole logic covered?

- **Think**
  - Where to start?
  - Create a GREEN test for the existing business logic **including DB access**
  - Test average rating for DXTR1000 (= 4.5)

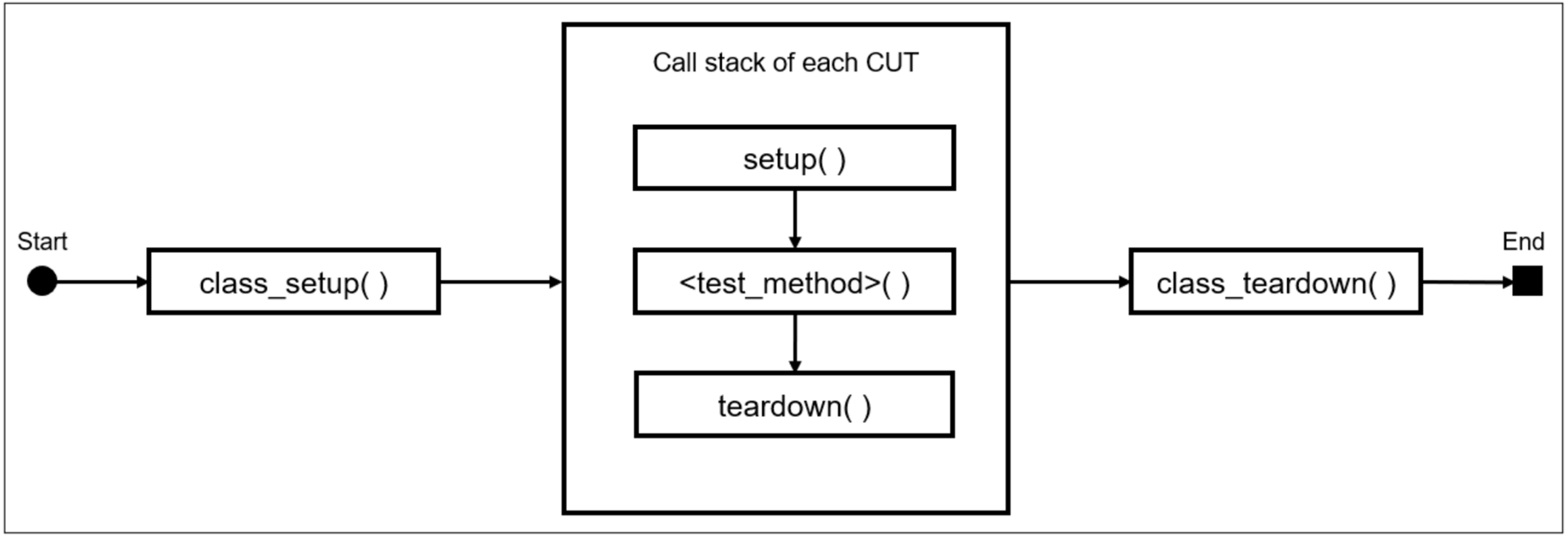


Data Preview

find pattern — 6 rows retrieved - 20 ms

CLIENT	RATING_UUID	PRODUCT	NAME	EMAIL	RATING	REVIEW
100	E21FD1B74D261E	DXTR1000	Anna Baue	anna@bau	4	The color o
100	E21FD1B74D261E	DXTR2000	Chris Drur	cd@test.d	5	The best bil
100	E21FD1B74D261E	DXTR3000	Eve Smith	es@email.	3	
100	E21FD1B74D261E	EPAD1000	Frank Jon	f@jones.m	5	Size varies
100	E21FD1B74D261E	EPAD1000	Gill Miller	g.miller@c	4	
100	E21FD1B74D261E	DXTR1000	G. Miller	g.miller@c	5	

- # ABAP Unit Test Framework



Source: https://developers.sap.com/tutorials/abap-environment-rap100-unit-testing..html

- **GREEN**
  - Debugger -> In-/Output Data
  - DB access -> CDSs are fine

```
ZCL_VE_AVERAGE_RATING_MJ  LTCL_AVERAGE_RATING
1  CLASS ltcl_average_rating DEFINITION FINAL FOR TESTING
2    DURATION SHORT
3    RISK LEVEL HARMLESS.
4
5    PRIVATE SECTION.
6      DATA cut TYPE REF TO if_sadl_exit_calc_element_read.
7
8      METHODS:
9        setup.
10     METHODS:
11       " GIVEN: Input data for Product DXTR1000 WHEN: Calculate average rating THEN: ...
12       should_calculate_correctly FOR TESTING RAISING cx_static_check.
13
14 ENDCLASS.
15
```

```
18
19   METHOD setup.
20     cut = NEW zcl_ve_average_rating_mj( ).
21   ENDMETHOD.
22
23   METHOD should_calculate_correctly.
24     DATA products        TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
25     DATA calculated_data TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
26     DATA expected_values TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
27
28     " Set Input Data
29     products        = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black'   AverageRating = 0 ) ).
30
31     calculated_data = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black'   AverageRating = 4 ) ).
32
33     " Set Expected Result
34     expected_values = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black'   AverageRating = '4.5' ) ).
35
36     " Execute calculation
37     cut->calculate( EXPORTING it_original_data           = products
38                               it_requested_calc_elements = VALUE if_sadl_exit_calc_element_read=>tt_elements( )
39                     CHANGING  ct_calculated_data         = calculated_data ).
40
41     " Compare expected and calculated data
42     cl_abap_unit_assert=>assert_equals( exp = expected_values
43                                         act = calculated_data ).
44
45
46   ENDMETHOD.
47
```

```
Probl  Prope  Templ  Trans  abap  ABAP  Searc  ATC P  ABAP  Quick  Git Re  Cons  Histor  ABAP  abap  ABAP
[TRL] ZCL_VE_AVERAGE_RATING_MJ (20:20:13)   Methods: 1  Duration: 1.830 s   0  0  0  1  0
```

- **Refactor (Magic time)**
  - **Extract “functional” method**
  - **“Information hiding”**
  - **“Best of three”**

```
27  METHOD if_sadl_exit_calc_element_read~calculate.
28    DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
29    DATA average_product_ratings TYPE tt_average_product_ratings.
30
31    products = CORRESPONDING #( it_original_data ).
32
33    " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
34    READ ENTITIES OF Z_C_Product_M_MJ
35      ENTITY Product BY \_Rating
36      FIELDS ( Rating )
37      WITH CORRESPONDING #( products )
38      RESULT FINAL(ratings).
39
40    " Calculate the average rating for each product by DB select
41    SELECT Product,
42           AVG( rating AS DEC( 2, 1 ) ) AS average_rating
43      FROM @ratings AS r
44      GROUP BY Product
45      INTO TABLE @average_product_ratings.
46
47    " Map average ratings to the products (output)
48    products = map_average_ratings2products( average_product_ratings = average_product_ratings
49                                             products                = products ).
50
51    ct_calculated_data = CORRESPONDING #( products ).
52  ENDMETHOD.
53
```

```
54  METHOD map_average_ratings2products.
55
56    result = products.
57
58    LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
59      <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.
60    ENDLOOP.
61
62  ENDMETHOD.
63
```

- **Refactor (Magic time)**
  - **Extract “functional” method**
  - **“Information hiding”**
  - **“Best of three”**

```
27  METHOD if_sadl_exit_calc_element_read~calculate.
28    DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
29    DATA average_product_ratings TYPE tt_average_product_ratings.
30
31    products = CORRESPONDING #( it_original_data ).
32
33    " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
34    READ ENTITIES OF Z_C_Product_M_MJ
35      ENTITY Product BY \_Rating
36      FI
37      WI
38      RE
39
40    " Ca
41    SELE
42
43      FR
44      GR
45      IN
46
47    " Ma
48    proc
49
50
51    ct_calculated_data = CORRESPONDING #( products ).
52  ENDMETHOD.
53
```

```
PRIVATE SECTION.
  TYPES:
    ty_products TYPE STANDARD TABLE OF z_c_product_m_mj WITH DEFAULT KEY,

    BEGIN OF ty_average_product_rating,
      Product        TYPE ze_product_id,
      average_rating TYPE z_c_product_m_mj-AverageRating,
    END OF ty_average_product_rating,

    tt_average_product_ratings TYPE STANDARD TABLE OF ty_average_product_rating WITH DEFAULT KEY.

  METHODS map_average_ratings2products IMPORTING average_product_ratings TYPE tt_average_product_ratings
                                                 products                TYPE ty_products
                                       RETURNING VALUE(result)           TYPE ty_products.
```

```
54  METHOD map_average_ratings2products.
55
56    result = products.
57
58    LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
59      <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.
60    ENDLOOP.
61
62  ENDMETHOD.
63
```

- **Refactor (Magic time)**
  - **Extract “functional” method**
  - **“Information hiding”**
  - **“Best of three”**

```
METHOD if_sadl_exit_calc_element_read~calculate.
  DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
  DATA average_product_ratings TYPE tt_average_product_ratings.

  products = CORRESPONDING #( it_original_data ).

  " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
  READ ENTITIES OF Z_C_Product_M_MJ
    ENTITY Product BY \_Rating
    FI
    WI
    RE

  " Ca
  SELE

    FR
    GR
    IN

  " Ma
  proc


  ct_calculated_data = CORRESPONDING #( products ).
ENDMETHOD.
```

```
PRIVATE SECTION.
  TYPES:
    ty_products TYPE STANDARD TABLE OF z_c_product_m_mj WITH DEFAULT KEY,

    BEGIN OF ty_average_product_rating,
      Product        TYPE ze_product_id,
      average_rating TYPE z_c_product_m_mj-AverageRating,
    END OF ty_average_product_rating,

    tt_average_product_ratings TYPE STANDARD TABLE OF ty_average_product_rating WITH DEFAULT KEY.

  METHODS map_average_ratings2products IMPORTING average_product_ratings TYPE tt_average_product_ratings
                                                 products                TYPE ty_products
                                       RETURNING VALUE(result)           TYPE ty_products.
```

```
METHOD map_average_ratings2products.

  result = products.

  LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
    <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.
  ENDLOOP.

ENDMETHOD.
```

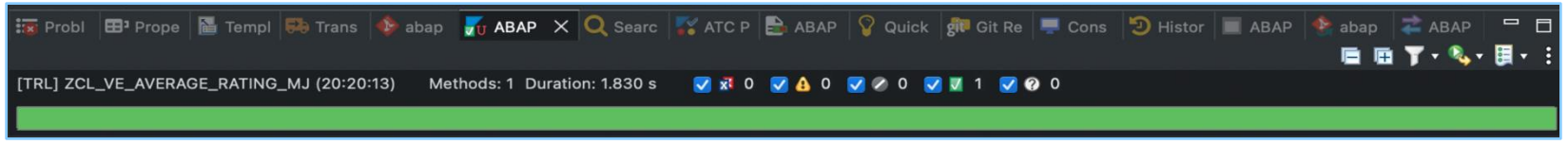
Probl | Prope | Templ | Trans | abap | ABAP | Searc | ATC P | ABAP | Quick | Git Re | Cons | Histor | ABAP | abap | ABAP

[TRL] ZCL_VE_AVERAGE_RATING_MJ (20:20:13) Methods: 1 Duration: 1.830 s 0 0 0 1 0

- **Refactor (Magic time)**
  - **Extract “functional” method**

```
 9   PRIVATE SECTION.
10     TYPES tt_products TYPE STANDARD TABLE OF z_c_product_m_mj WITH DEFAULT KEY.
11     TYPES tt_ratings  TYPE TABLE FOR READ RESULT z_c_product_m_mj\\product\_rating.
12
13     TYPES: BEGIN OF ty_average_product_rating,
14              Product        TYPE z_c_product_m_mj-ProductId,
15              average_rating TYPE z_c_product_m_mj-AverageRating,
16            END OF ty_average_product_rating.
17
18     TYPES tt_average_product_ratings TYPE STANDARD TABLE OF ty_average_product_rating WITH DEFAULT KEY.
19
20     METHODS map_average_ratings2products IMPORTING average_product_ratings TYPE tt_average_product_ratings
21                                                    products                TYPE tt_products
22                                          RETURNING VALUE(result)           TYPE tt_products.
23
24     METHODS read_average_rating4products IMPORTING products      TYPE tt_products
25                                          RETURNING VALUE(result) TYPE tt_average_product_ratings.
26
```

```
29  CLASS zcl_ve_average_rating_mj IMPLEMENTATION.
30
31    METHOD if_sadl_exit_calc_element_read~calculate.
32      DATA(products) = CORRESPONDING tt_products( it_original_data ).
33
34      DATA(average_product_ratings) = read_average_rating4products( products ).
35
36      products = map_average_ratings2products( average_product_ratings = average_product_ratings
37                                               products                = products ).
38
39      ct_calculated_data = CORRESPONDING #( products ).
40    ENDMETHOD.
41
42    METHOD read_average_rating4products.
43      " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
44      READ ENTITIES OF Z_C_Product_M_MJ
45        ENTITY Product BY \_Rating
46        FIELDS ( Rating )
47        WITH CORRESPONDING #( products )
48        RESULT FINAL(ratings).
49
50      " Calculate the average rating for each product by DB selec
51      SELECT Product,
52             AVG( rating AS DEC( 2, 1 ) ) AS average_rating
53        FROM @ratings AS r
54        GROUP BY Product
55        INTO TABLE @result.
56    ENDMETHOD.
```



- ## Refactor (Decoupling of DB dependency)

  - **For READ operations:**

  Using CDS Test Environment standard class *cl_cds_test_environment*

  - **For CUD operations:**

  + SQL Test Environment standard class cl_sql_test_environment

- **REFACTOR: Mocking ENTITY access**

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

  PRIVATE SECTION.
    CLASS-DATA:
      cds_test_environment TYPE REF TO if_cds_test_environment,
      ratings              TYPE STANDARD TABLE OF zrating_mj.

    CLASS-METHODS:
      class_setup,
      class_teardown.

    DATA cut TYPE REF TO if_sadl_exit_calc_element_read.

    METHODS:
      setup,
      teardown.

    METHODS:
      " GIVEN: Input data for Product DXTR1000 WHEN: Calculate average rating THEN: ...
      should_calculate_correctly FOR TESTING RAISING cx_static_check.

ENDCLASS.
```

- **REFACTOR: Mocking ENTITY access**

```
4  CLASS ltcl_average_rating DEFINITION FINAL FOR TESTING
5    DURATION SHORT
6    RISK LEVEL HARMLESS.
7
8    PRIVATE SECTION.
9      CLASS-DATA:
10       cds_test_environment TYPE REF TO if_cds_test_environment,
11       ratings              TYPE STANDARD TABLE OF zrating_mj.
12
13     CLASS-METHODS:
14       class_setup,
15       class_teardown.
16
17     DATA cut TYPE REF TO if_sadl_exit_calc_element_read.
```

```
30   METHOD class_setup.
31     " create the test doubles for the underlying CDS entities
32     cds_test_environment = cl_cds_test_environment=>create(
33                       i_for_entity = 'Z_C_RATING_M_MJ'
34                       i_dependency_list = VALUE #( ( name = 'ZRATING_MJ'  type = 'TABLE' ) ) ).
35     ratings = VALUE #( ( rating_uuid = '1' product = 'DXTR1000' rating = 4 )
36                        ( rating_uuid = '2' product = 'DXTR1000' rating = 5 )
37                        ( rating_uuid = '3' product = 'DXTR2000' rating = 5 ) ).
38   ENDMETHOD.
39
40   METHOD class_teardown.
41     " remove test doubles
42     cds_test_environment->destroy(  ).
43   ENDMETHOD.
```

```
EN: Calculate average rating THEN: ...
ING cx_static_check.
```

- **REFACTOR: Mocking ENTITY access**

```
4  CLASS ltcl_average_rating DEFINITION FINAL FOR TESTING
5    DURATION SHORT
6    RISK LEVEL HARMLESS.
7
8    PRIVATE SECTION.
9      CLASS-DATA:
10       cds_test_environment TYPE REF TO if_cds_test_environment,
11       ratings              TYPE STANDARD TABLE OF zrating_mj.
12
13     CLASS-METHODS:
14       class_setup,
15       class_teardown.
16
17     DATA cut TYPE REF TO if_sadl_exit_calc_element_read.
```

```
30   METHOD class_setup.
31     " create the test doubles for the underlying CDS entities
32     cds_test_environment = cl_cds_test_environment=>create(
33                       i_for_entity = 'Z_C_RATING_M_MJ'
34                       i_dependency_list = VALUE #( ( name = 'ZRATING_MJ'  type = 'TABLE' ) ) ).
35     ratings = VALUE #( ( rating_uuid = '1' product = 'DXTR1000' rating = 4 )
36                        ( rating_uuid = '2' product = 'DXTR1000' rating = 5 )
37                        ( rating_uuid = '3' product = 'DXTR2000' rating = 5 ) ).
38   ENDMETHOD.
39
40   METHOD class_teardown.
41     " remove test doubles
42     cds_test_environment->destroy(  ).
43   ENDMETHOD.
```

```
45   METHOD setup.
46     cut = NEW zcl_ve_average_rating_mj( ).
47
48     " clear the test doubles per test
49     cds_test_environment->clear_doubles( ).
50
51     " insert test data into test doubles
52     cds_test_environment->insert_test_data( ratings ).
53   ENDMETHOD.
54
55   METHOD teardown.
56     ROLLBACK ENTITIES.                                    "#EC CI_ROLLBACK
57   ENDMETHOD.
```

```
EN: Calculate average rating THEN: ...
ING cx_static_check.
```

- # REFACTOR: Mocking ENTITY access

  - **Test method not changed**

```
59  METHOD should_calculate_correctly.
60    DATA products        TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
61    DATA calculated_data TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
62    DATA expected_values TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
63
64    " Set Input Data
65    products        = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black'   AverageRating = 0 ) ).
66    calculated_data = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black'   AverageRating = 4 ) ).
67
68    " Set Expected Result
69    expected_values = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black'   AverageRating = '4.5' ) ).
70
71    " Execute calculation
72    cut->calculate( EXPORTING it_original_data           = products
73                              it_requested_calc_elements = VALUE if_sadl_exit_calc_element_read=>tt_elements( )
74                    CHANGING  ct_calculated_data         = calculated_data ).
75
76    " Compare expected and calculated data
77    cl_abap_unit_assert=>assert_equals( exp = expected_values
78                                        act = calculated_data ).
79  ENDMETHOD.
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Probl | Prope | Templ | Trans | abap | ABAP | Searc | ATC P | ABAP | Quick | Git Re | Cons | Histor | ABAP | abap | ABAP

[TRL] ZCL_VE_AVERAGE_RATING_MJ (20:20:13) Methods: 1 Duration: 1.830 s 0 0 0 1 0

How to know what to mock?

```
32    cds_test_environment = cl_cds_test_environment=>create(
33                         i_for_entity = 'Z_C_RATING_M_MJ'
34                         i_dependency_list = VALUE #( ( name = 'ZRATING_MJ'  type = 'TABLE' ) ) ).
```

```
1  projection;
2  strict ( 1 );
3
4  define behavior for Z_C_Product_M_MJ alias Product
5  {
6    use create;
7    use update;
8    use delete;
9
10   use association _Rating { create; }
11 }
12
13 define behavior for Z_C_Rating_M_MJ alias Rating
14 {
15   use update;
16   use delete;
17
18   use action setStatusToCompleted;
19
20   use association _Product;
21 }
```

```
61      READ ENTITIES OF Z_C_Product_M_MJ
62        ENTITY Product BY \_Rating
63        FIELDS ( Rating )
64        WITH CORRESPONDING #( products )
65        RESULT FINAL(ratings).
```


```
11 define view entity Z_C_RATING_M_MJ
12   as projection on Z_I_RATING_MJ
13 {
14   key RatingUUID,
15       Product,
16       Name,
17       Email,
18       Rating,
19       Review,
20       CreatedBy,
21       CreatedAt,
22       LastChangedBy,
23       LastChangedAt,
24       Status,
25       StatusCriticality,
26
27       /* Associations */
28       _Product : redirected to parent Z_C_Product_M_MJ
29 }
```

```
4  define view entity Z_I_RATING_MJ
5    as select from zrating_mj
6    association to parent Z_I_Product_MJ
7  {
8    key rating_uuid     as RatingUUID,
9        product         as Product,
10       name            as Name,
11       email           as Email,
12       rating          as Rating,
13       review          as Review,
14       status          as Status,
```

- **Refactoring is FUN and a super Learning “Tool” ☺**

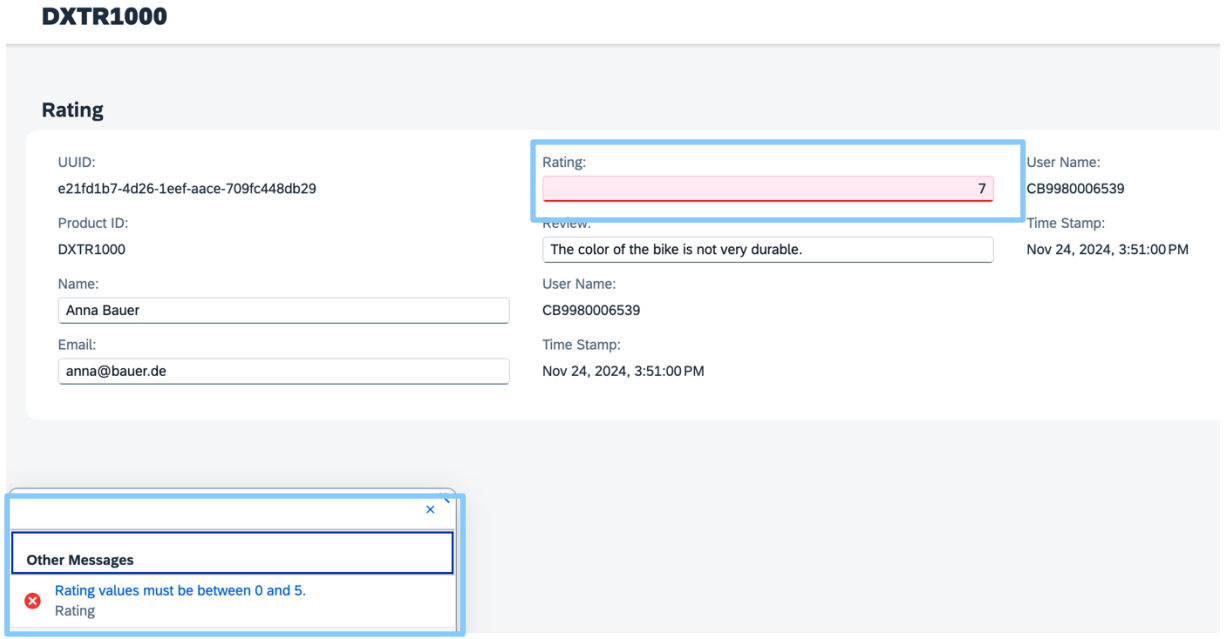
```
54  METHOD map_average_ratings2products.
55
56    result = products.
57
58    LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
59      <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.
60    ENDLOOP.
61
62  ENDMETHOD.
63
```

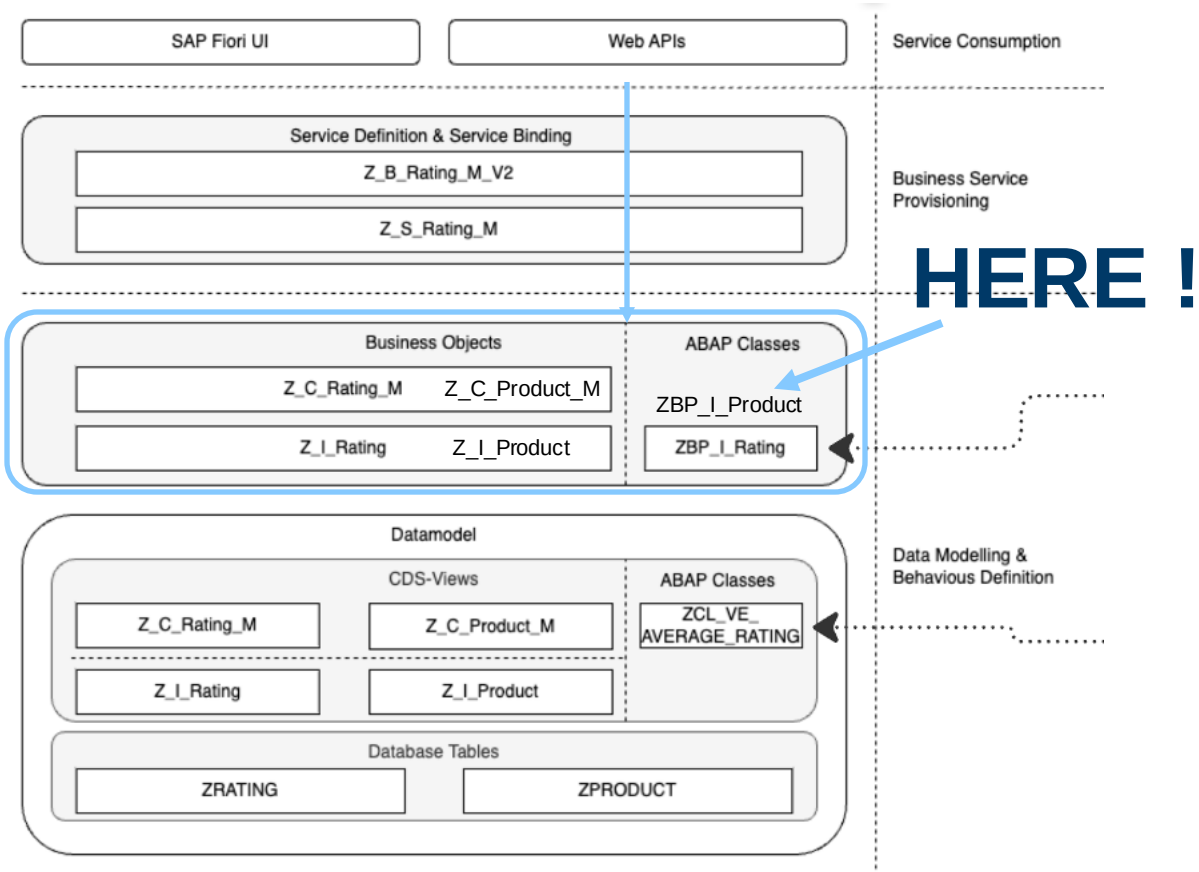
```
58  METHOD map_average_ratings2products.
59    result = products.
60
61    LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
62      <product>-AverageRating = VALUE #( average_product_ratings[ Product = <product>-ProductId ]-average_rating OPTIONAL ).
63    ENDLOOP.
64  ENDMETHOD.
```

«Power of three»

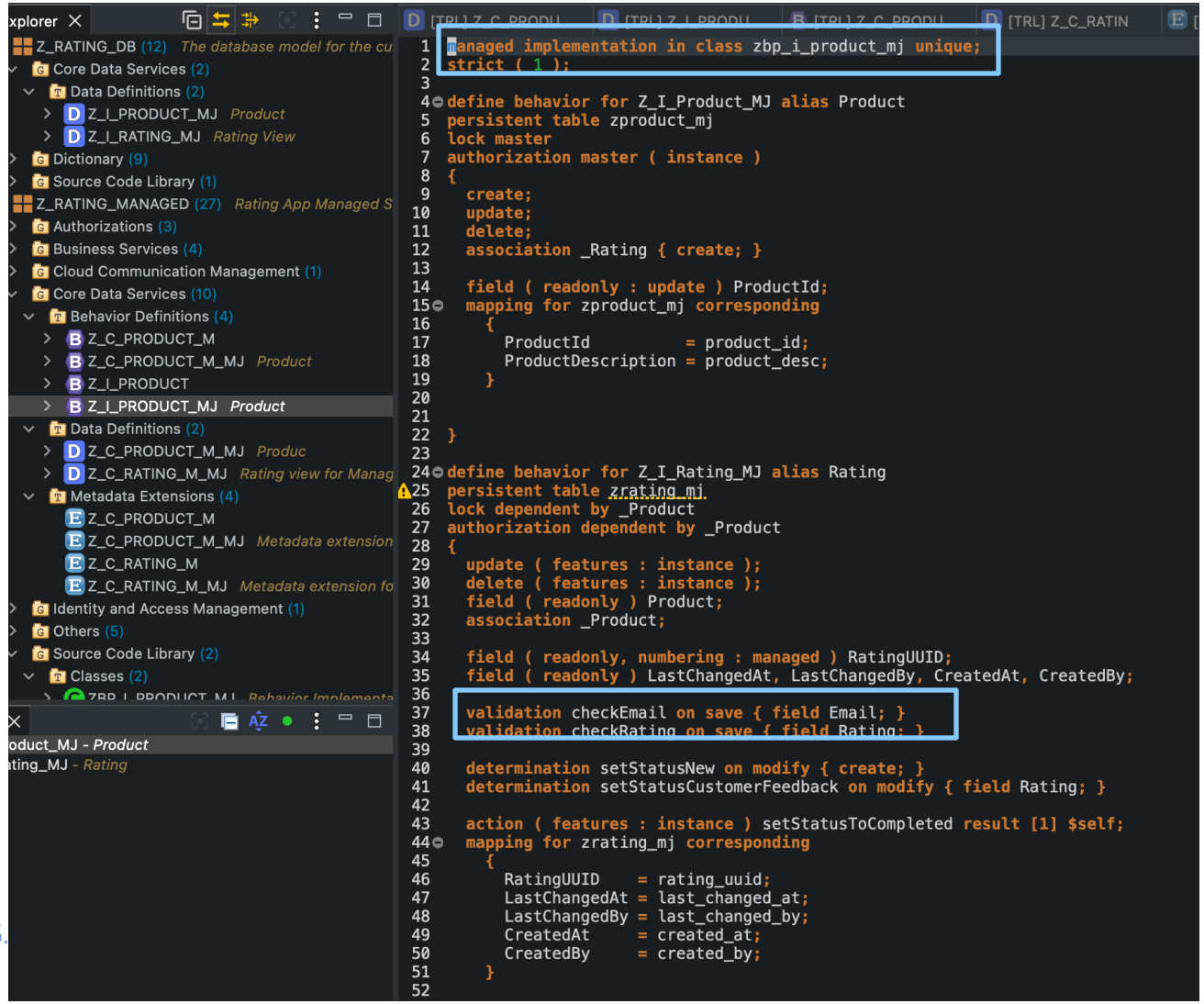
Showcase – Check Rating

… please use numbers between 1 and 5!

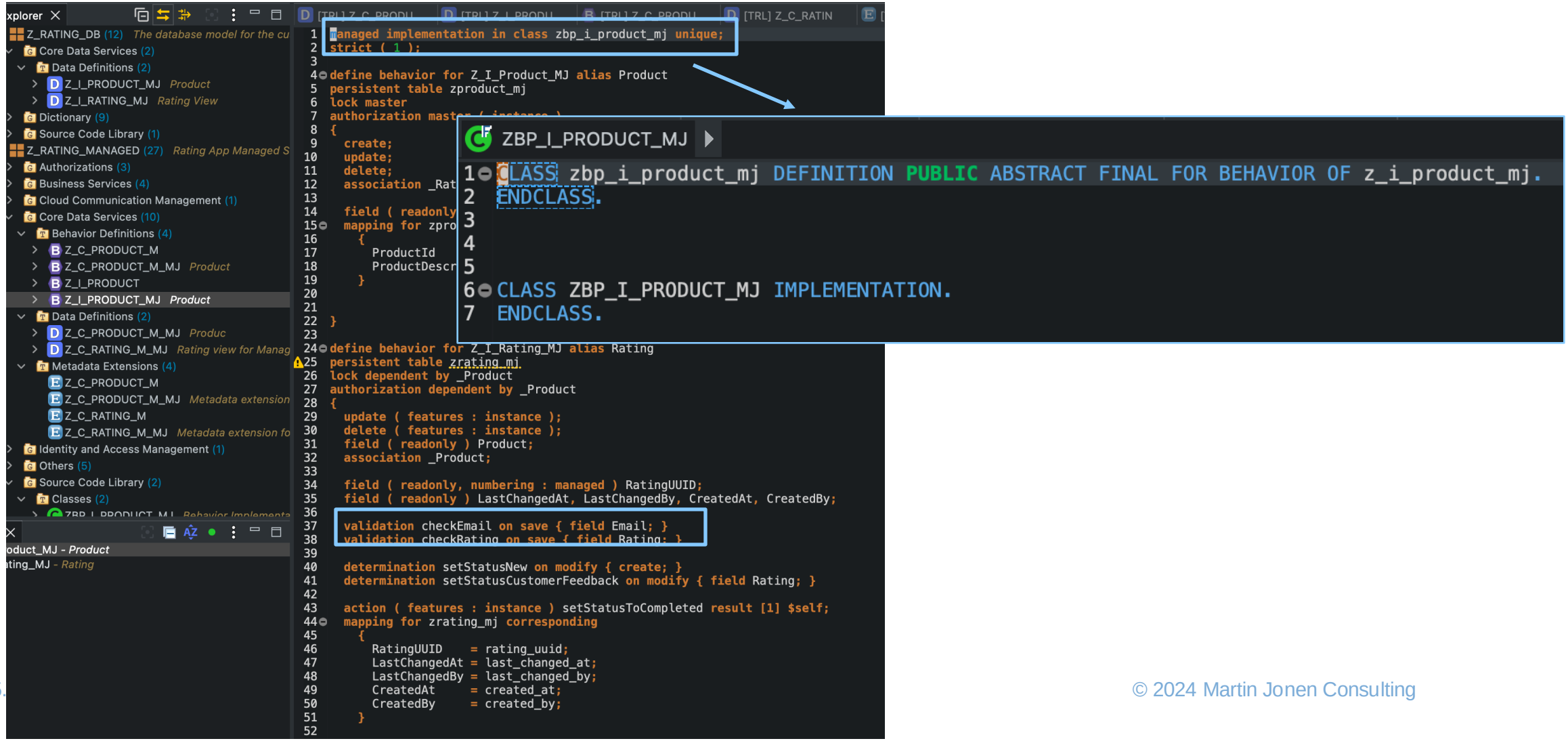




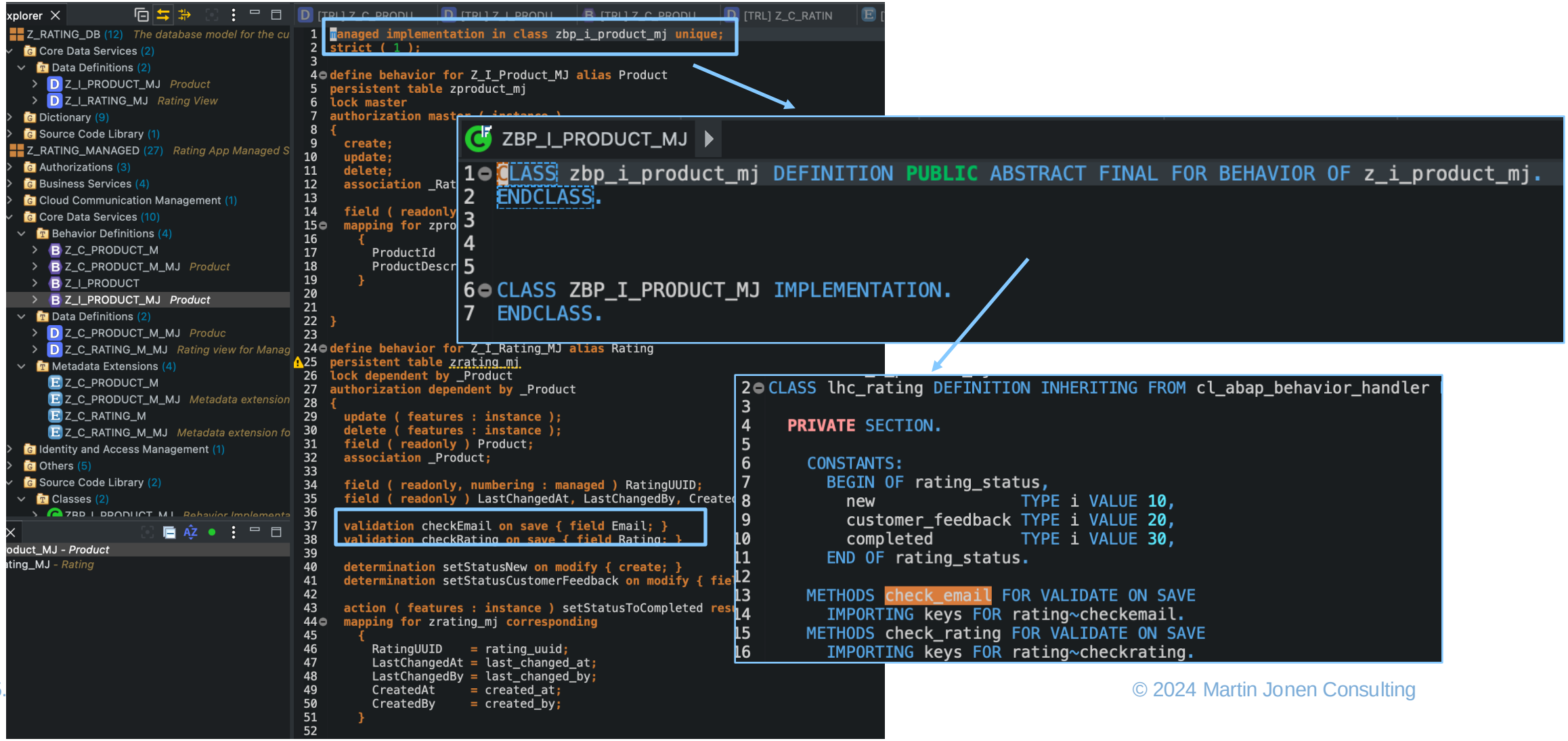
Testing the Bahavior Definition/Business Object logic



Testing the Bahavior Definition/Business Object logic



Testing the Bahavior Definition/Business Object logic



How to UNIT test this scenario?

- **Abstract class is not accessible, but …**
  - **FRIENDS are giving us a helping hand**

ZBP_I_PRODUCT_MJ ▸ LHC_RATING ▸ LTCL_I_PRODUCT_MJ

```
1  CLASS ltcl_i_product_mj DEFINITION DEFERRED FOR TESTING.
2  CLASS lhc_rating DEFINITION INHERITING FROM cl_abap_behavior_handler FRIENDS ltcl_i_product_mj.
3
```

- **THINK**
  - **Test for Rating 5**

ZBP_I_PRODUCT_MJ ▸ LHC_RATING ▸ LTCL_I_PRODUCT_MJ

```
1  CLASS ltcl_i_product_mj DEFINITION DEFERRED FOR TESTING.
2  CLASS lhc_rating DEFINITION INHERITING FROM cl_abap_behavior_handler FRIENDS ltcl_i_product_mj.
3
4    PRIVATE SECTION.
5
6      CONSTANTS:
7        BEGIN OF rating_status,
8          new               TYPE i VALUE 10,
9          customer_feedback TYPE i VALUE 20,
10         completed         TYPE i VALUE 30,
11       END OF rating_status.
12
13     METHODS check_email FOR VALIDATE ON SAVE
14       IMPORTING keys FOR rating~checkemail.
15     METHODS check_rating FOR VALIDATE ON SAVE
```

```
METHOD check_rating.
  READ ENTITIES OF z_i_product_mj IN LOCAL MODE
      ENTITY rating
        FIELDS ( rating )
        WITH CORRESPONDING #( keys )
      RESULT DATA(ratings).

  LOOP AT ratings ASSIGNING FIELD-SYMBOL(<rating>).
    IF <rating>-rating < 0 OR <rating>-rating > 5.
      APPEND VALUE #( %key = <rating>-%key ) TO failed-rating.

      APPEND VALUE #( %key = <rating>-%key
                      %msg = new_message( id       = 'ZM_RATING_M'
                                          number   = '002'
                                          severity = if_abap_behv_message=>severity-error )
                   %element-rating = if_abap_behv=>mk-on ) TO reported-rating.
    ENDIF.
  ENDLOOP.
ENDMETHOD.
```

- **GREEN**

ZBP_I_PRODUCT_MJ ▸ LTCL_I_PRODUCT_MJ ▸ SHOULD_ACCEPT_RATING_5

```
1  CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2    DURATION SHORT
3    RISK LEVEL HARMLESS.
4
5    PRIVATE SECTION.
6      CLASS-DATA:
7        cut            TYPE REF TO lhc_rating.  " Must be static, because local class is static
8
9      CLASS-METHODS:
10       "! Instantiate class under test and setup test double frameworks
11       class_setup,
12       "! Destroy test environments and test doubles
13       class_teardown.
14
15     METHODS:
16       "! Reset test doubles
17       setup,
18       "! Reset transactional buffer
19       teardown.
20     METHODS:
21       " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
22       should_accept_rating_5 FOR TESTING RAISING cx_static_check.
23
24 ENDCLASS.
```

- **GREEN**

ZBP_I_PRODUCT_MJ ▸ LTCL_I_PRODUCT_MJ ▸ SHOULD_ACCEPT_RATING_5

```
1  CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2    DURATION SHORT
3    RISK LEVEL HARMLESS.
4
5    PRIVATE SECTION.
6      CLASS-DATA:
7        cut            TYPE REF TO lhc_rating.  " Must be static, because local class is static
8
9      CLASS-METHODS:
```

```
METHOD class_setup.
  CREATE OBJECT cut FOR TESTING.
ENDMETHOD.
```

```
15     METHODS:
16       "! Reset test doubles
17       setup,
18       "! Reset transactional buffer
19       teardown.
20     METHODS:
21       " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
22       should_accept_rating_5 FOR TESTING RAISING cx_static_check.
23
24 ENDCLASS.
```

- **GREEN**

ZBP_I_PRODUCT_MJ ▸ LTCL_I_PRODUCT_MJ ▸ SHOULD_ACCEPT_RATING_5

```
1  CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2    DURATION SHORT
3    RISK LEVEL HARMLESS.
4
5    PRIVATE SECTION.
6      CLASS-DATA:
7        cut            TYPE REF TO lhc_rating.  " Must be static, because local class is static
8
9      CLASS-METHODS:
```

```
METHOD class_setup.
  CREATE OBJECT cut FOR TESTING.
ENDMETHOD.
```

```
15     METHODS:
16       "! Reset test doubles
17       setup,
18       "! Reset transactional buffer
19       teardown.
20     METHODS:
21       " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
22       should_accept_rating_5 FOR TESTING RAISING cx_static_check.
23
```

```
42  METHOD should_accept_rating_5.
43    DATA keys     TYPE TABLE FOR VALIDATION z_i_product_mj\\rating~checkrating.
44    DATA failed   TYPE RESPONSE FOR FAILED LATE z_i_product_mj.
45    DATA reported TYPE RESPONSE FOR REPORTED LATE  z_i_product_mj.
46
47    " Call method to be tested
48    keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID
49    cut->check_rating( EXPORTING keys     = CORRESPONDING #( keys )
50                       CHANGING  failed   = failed
51                                 reported = reported ).
52
53    cl_abap_unit_assert=>assert_initial( failed ).
54    cl_abap_unit_assert=>assert_initial( reported ).
55  ENDMETHOD.
```

- **GREEN**

ZBP_I_PRODUCT_MJ ▸ LTCL_I_PRODUCT_MJ ▸ SHOULD_ACCEPT_RATING_5

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

  PRIVATE SECTION.
    CLASS-DATA:
      cut            TYPE REF TO lhc_rating.  " Must be static, because local class is static

    CLASS-METHODS:
      class_setup.
    METHODS:
      "! Reset test doubles
      setup,
      "! Reset transactional buffer
      teardown.
    METHODS:
      " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
      should_accept_rating_5 FOR TESTING RAISING cx_static_check.
```

```
METHOD class_setup.
  CREATE OBJECT cut FOR TESTING.
ENDMETHOD.
```

```
METHOD should_accept_rating_5.
  DATA keys     TYPE TABLE FOR VALIDATION z_i_product_mj\\rating~checkrating.
  DATA failed   TYPE RESPONSE FOR FAILED LATE z_i_product_mj.
  DATA reported TYPE RESPONSE FOR REPORTED LATE  z_i_product_mj.

  " Call method to be tested
  keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID
  cut->check_rating( EXPORTING keys     = CORRESPONDING #( keys )
                     CHANGING  failed   = failed
                               reported = reported ).

  cl_abap_unit_assert=>assert_initial( failed ).
  cl_abap_unit_assert=>assert_initial( reported ).
ENDMETHOD.
```

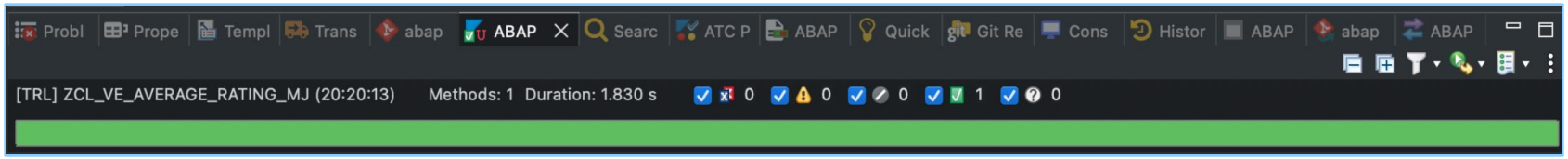
[TRL] ZCL_VE_AVERAGE_RATING_MJ (20:20:13) Methods: 1 Duration: 1.830 s 0 0 0 1 0

- # REFACTOR
  - **Extract “functional” method**

```
27      METHODS is_rating_valid IMPORTING rating        TYPE ze_rating
28                              RETURNING VALUE(result) TYPE abap_bool.
```

```
64  METHOD check_rating.
65    READ ENTITIES OF z_i_product_mj IN LOCAL MODE
66        ENTITY rating
67          FIELDS ( rating )
68          WITH CORRESPONDING #( keys )
69        RESULT DATA(ratings).
70
71    LOOP AT ratings ASSIGNING FIELD-SYMBOL(<rating>).
72      IF is_rating_valid( <rating>-rating ) = abap_false.
73        APPEND VALUE #( %key = <rating> %key ) TO failed-rating.
74
75        APPEND VALUE #( %key = <rating>-%key
76                        %msg = new_message( id       = 'ZM_RATING_M'
77                                            number   = '002'
78                                            severity = if_abap_behv_message=>severity-error )
79                     %element-rating = if_abap_behv=>mk-on ) TO reported-rating.
80      ENDIF.
81    ENDLOOP.
82  ENDMETHOD.
```

```
84  METHOD is_rating_valid.
85    result = xsdbool( rating >= 1 AND rating <= 5 ).
86  ENDMETHOD.
```



- **Refactor: Mocking ENTITY access**

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

  PRIVATE SECTION.
    CLASS-DATA:
      cut                  TYPE REF TO lhc_rating, " Must be static, because local class is static
      cds_test_environment TYPE REF TO if_cds_test_environment.

    CLASS-METHODS:
      "! Instantiate class under test and setup test double frameworks
      class_setup,
      "! Destroy test environments and test doubles
      class_teardown.

    DATA:
      keys             TYPE TABLE FOR VALIDATION z_i_product_mj\\rating~checkrating,
      rating_mock_data TYPE STANDARD TABLE OF zrating_mj,
      failed           TYPE RESPONSE FOR FAILED LATE z_i_product_mj,
      reported         TYPE RESPONSE FOR REPORTED LATE  z_i_product_mj.

    METHODS:
      "! Reset test doubles
      setup,
      "! Reset transactional buffer
      teardown.
    METHODS:
      " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
      should_accept_rating_5 FOR TESTING RAISING cx_static_check.

ENDCLASS.
```

- # Refactor: Mocking ENTITY access

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

  PRIVATE SECTION.
    CLASS-DATA:
      cut                  TYPE REF TO lhc_rating, " Must be static, because local class is static
      cds_test_environment TYPE REF TO if_cds_test_environment.
```

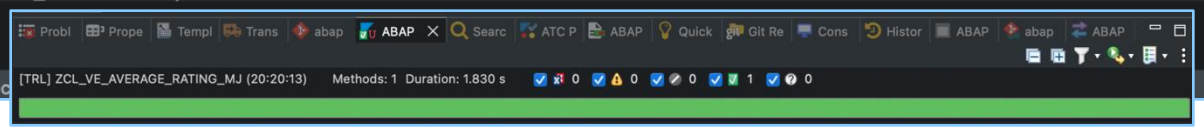
```
CLASS ltcl_i_product_mj IMPLEMENTATION.

  METHOD class_setup.
    CREATE OBJECT cut FOR TESTING.

    cds_test_environment = cl_cds_test_environment=>create( i_for_entity = 'Z_I_RATING_MJ'
                                                            i_dependency_list = VALUE #( ( name = 'ZRATING_MJ' type = 'TABLE' ) ) ).
  ENDMETHOD.

  METHOD class_teardown.
    cds_test_environment->destroy( ).
  ENDMETHOD.

  METHOD setup.
    cds_test_environment->clear_doubles( ).
  ENDMETHOD.

  METHOD teardown.
    ROLLBACK ENTITIES.                              "#EC CI_ROLLBACK
  ENDMETHOD.

  METHOD should_accept_rating_5.
    " Setup Rating Mock Data
    rating_mock_data = VALUE #( ( rating_uuid = 'E21FD1B74D261EEFAACE709FC448FB29' product = 'DXTR1000' rating = '5' ) ).
    cds_test_environment->insert_test_data( rating_mock_data ).

    " Call method to be tested
    keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID
    cut->check_rating( EXPORTING keys     = CORRESPONDING #( keys )
                       CHANGING  failed   = failed
                                 reported = reported ).

    cl_abap_unit_assert=>assert_initial( failed ).
    cl_abap_unit_assert=>assert_initial( reported ).
  ENDMETHOD.

ENDCLASS.
```

Global Class | Class-relevant Local Types | Local Types | **Test Classes** | Macros

- # Refactor: Mocking ENTITY access

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

  PRIVATE SECTION.
    CLASS-DATA:
      cut                  TYPE REF TO lhc_rating, " Must be static, because local class is static
      cds_test_environment TYPE REF TO if_cds_test_environment.
```

```
CLASS ltcl_i_product_mj IMPLEMENTATION.

  METHOD class_setup.
    CREATE OBJECT cut FOR TESTING.

    cds_test_environment = cl_cds_test_environment=>create( i_for_entity = 'Z_I_RATING_MJ'
                                                            i_dependency_list = VALUE #( ( name = 'ZRATING_MJ' type = 'TABLE' ) ) ).
  ENDMETHOD.

  METHOD class_teardown.
    cds_test_environment->destroy( ).
  ENDMETHOD.

  METHOD setup.
    cds_test_environment->clear_doubles( ).
  ENDMETHOD.

  METHOD teardown.
    ROLLBACK ENTITIES.                              "#EC CI_ROLLBACK
  ENDMETHOD.

  METHOD should_accept_rating_5.
    " Setup Rating Mock Data
    rating_mock_data = VALUE #( ( rating_uuid = 'E21FD1B74D261EEFAACE709FC448FB29' product = 'DXTR1000' rating = '5' ) ).
    cds_test_environment->insert_test_data( rating_mock_data ).

    " Call method to be tested
    keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID
    cut->check_rating( EXPORTING keys     = CORRESPONDING #( keys )
                       CHANGING  failed   = failed
                                 reported = reported ).

    cl_abap_unit_assert=>assert_initial( failed ).
    cl_abap_unit_assert=>assert_initial( reported ).
  ENDMETHOD.

ENDCLASS.
```



- **THINK -> GREEN (2nd testcase)**

```
27      METHODS:
28        " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
29        should_accept_rating_5 FOR TESTING RAISING cx_static_check,
30        " GIVEN: Rating Input of value 6 WHEN: Check_Rating is executed THEN: ...
31        should_reject_rating_6 FOR TESTING RAISING cx_static_check.
32
```

```
71  METHOD should_reject_rating_6.
72    " Setup Rating Mock Data
73    rating_mock_data = VALUE #( ( rating_uuid = 'E21FD1B74D261EEFAACE709FC448FB29' product = 'DXTR1000' rating = '6' ) ).
74    cds_test_environment->insert_test_data( rating_mock_data ).
75    " Call method to be tested
76    keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID
77    cut->check_rating( EXPORTING keys     = CORRESPONDING #( keys )
78                       CHANGING  failed   = failed
79                                 reported = reported ).
80    cl_abap_unit_assert=>assert_not_initial( failed ).
81    cl_abap_unit_assert=>assert_not_initial( reported ).
82  ENDMETHOD.
```

Check values?

What we learned

... uff!

- **UNIT test framework** in context with business logic in RAP
  - via *Data Definitions*
  - via *Behavior Definition*

- How to **mock ENTITIY access** by SAP Standard Class *cl_cds_test_environment*

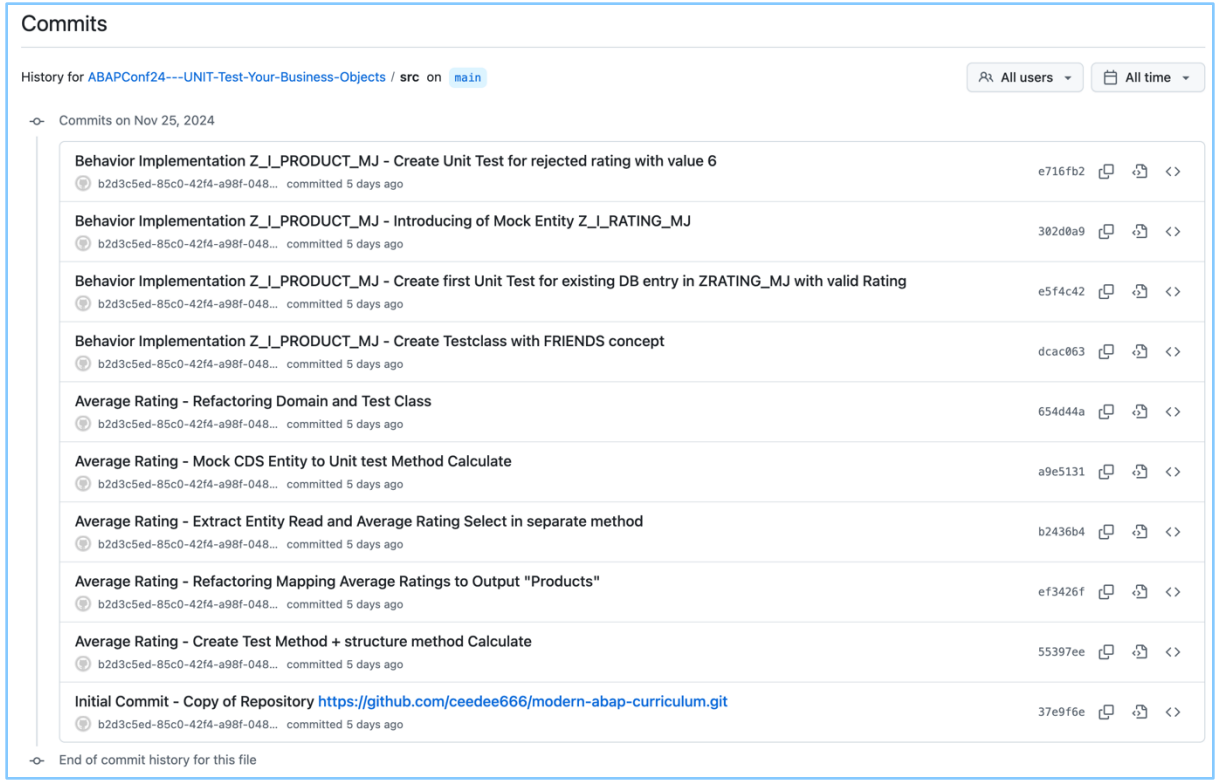
- **Refactoring technics:**
  - Extract (functional) method (information hiding)
  - «Power of three»

- **T**est **A**fter **D**evelopment for legacy code (not TDD)

What is there to discover

... puh!

- Try it out on your own
  - GitHub Repo: https://github.com/MaddinJay/ABAPConf24---UNIT-Test-Your-Business-Objects



What is there to discover

... puh!

- Mock the CUD ENTITY access and Business Objects itself
  - SHOW CASE - https://developers.sap.com/tutorials/abap-environment-rap100-unit-testing..html
  - Search for **/DMO/*TRAVEL*** Behavior Definitions **with UNIT Tests** 😲
  - https://github.com/SAP-samples/abap-cheat-sheets/blob/main/14_ABAP_Unit_Tests.md
- ACCCH-Challenge: Help me to update the Github
  - DM via LinkedIn

www.abapconf.org | #abapconf

December 5th 2024

abap conf2024

Thank you!

Unit Test your Business Objects in RAP

Martin Jonen

13:30 – 14:20 CET
Channel 1
English