Fachhochschule Südwestfalen

University of Applied Sciences

ERP Application Programming (Summer Semester 2025)

Assignment No. 2

Dipl.-Inform. Alfred Kersting

Tasniha Fahmin Chowdhury

Matriculation Number: 30346633.

Table of Contents

1. Introduction

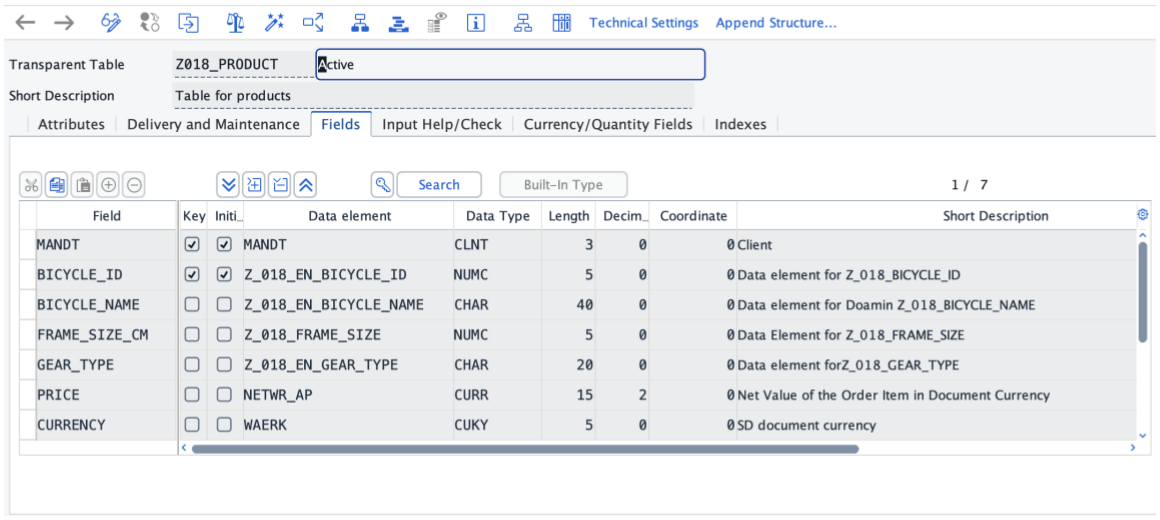
This report documents the design, development, and execution of an ABAP Object-Oriented program created for Assignment 2 of the ABAP programming course. The assignment focuses on developing a report that demonstrates object-oriented programming (OOP) principles using a custom local class, interacting with data elements defined in the SAP Data Dictionary, and outputting structured data on the screen. The project was completed entirely in ABAP and adheres to best practices including modular design, data encapsulation, and formatted output. The program is based on data from the custom table **Z018_PRODUCT**, which stores detailed product information.

2. Objective of the Assignment

The goal of this assignment was to apply ABAP Objects in a practical use case by developing a report that uses a local class to encapsulate product data and output it in a clean, object-wise format. Specific requirements included creating a class **LCL_PRODUCTS** with constructor logic, instance methods to set and get attributes, and class-level data to track the number of created product instances. The report had to generate multiple instances of this class and display their details using the WRITE command. The program structure required modularization using an INCLUDE statement, and robustness against user errors. This report explains how these objectives were achieved.

3. Database Table Description

The program is based on the database table Z018_PRODUCT, which was defined in Assignment 1 and reused here as per the assignment requirements. The table includes the following fields: **MANDT (client), BICYCLE_ID (key), BICYCLE_NAME, FRAME_SIZE_CM, GEAR_TYPE, PRICE,** and **CURRENCY**. These fields are bound to custom data elements such as **Z_018_EN_BICYCLE_ID, Z_018_FRAME_SIZE,** and standard types like **NETWR_AP** and **WAERK**. The table structure ensures that each product is uniquely identified and associated with a detailed description and pricing information. The data model was reused without modification, ensuring consistency and reusability of the Data Dictionary definitions.


Technical Settings Append Structure...

Transparent Table Z018_PRODUCT Active

Short Description Table for products

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields | Indexes

Search Built-In Type 1 / 7

Field	Key	Initi..	Data element	Data Type	Length	Decim..	Coordinate	Short Description
MANDT	✓	✓	MANDT	CLNT	3	0	0	Client
BICYCLE_ID	✓	✓	Z_018_EN_BICYCLE_ID	NUMC	5	0	0	Data element for Z_018_BICYCLE_ID
BICYCLE_NAME	☐	☐	Z_018_EN_BICYCLE_NAME	CHAR	40	0	0	Data element for Doamin Z_018_BICYCLE_NAME
FRAME_SIZE_CM	☐	☐	Z_018_FRAME_SIZE	NUMC	5	0	0	Data Element for Z_018_FRAME_SIZE
GEAR_TYPE	☐	☐	Z_018_EN_GEAR_TYPE	CHAR	20	0	0	Data element forZ_018_GEAR_TYPE
PRICE	☐	☐	NETWR_AP	CURR	15	2	0	Net Value of the Order Item in Document Currency
CURRENCY	☐	☐	WAERK	CUKY	5	0	0	SD document currency

4. Program Architecture

The solution is built in a modular fashion, separating concerns between the main report program and the object class definition. The main report **(Z018_ASSIGNMENT2)** includes a single local class **LCL_PRODUCTS**, defined and implemented inside an include file named **Z018_ASS2_LCL_PRODUCTS**. The class handles data encapsulation and output, while the main program is responsible for creating objects, managing object references in an internal table, and displaying the output in descending order. This structure aligns with standard ABAP architecture principles of separation of concerns and clean design.

5. INCLUDE z018_ass2_lcl_products

The **INCLUDE z018_ass2_lcl_products** contains the definition and implementation of the local class **lcl_products**, which encapsulates all logic related to product data management. This object-oriented design helps keep the main program clean and modular by delegating responsibilities such as storing product attributes, displaying data, and counting object instances. The class supports operations via methods like **set_attributes, get_attributes**, and a class method **get_number_of_products**, along with a constructor that auto-increments an internal counter. This include structure allows for reusability, better readability, and maintains separation of concerns, aligning with good ABAP development practices.

6. Class Definition and Implementation

The local class lcl_products, defined in the include file **z018_ass2_lcl_products**, handles all product-related logic in a structured way. It has private attributes like bicycle ID, name, gear type, frame size, price, and currency, which are set using the **set_attributes** method. The **get_attributes** method displays these values with proper formatting, such as left justification for currency. A **constructor** automatically increases the total product count using a class-level variable, and this count can be retrieved using the **class method get_number_of_products**. The class structure ensures clear separation between data and behavior, supporting clean and organized design.

```
*&---------------------------------------------------------------------*
*& Include z018_ass2_lcl_products
*       CLASS lcl_products DEFINITION
" Definition of the local class LCL_PRODUCTS
CLASS lcl_products DEFINITION.
  PUBLIC SECTION.    " Publicly accessible methods and class methods
    METHODS:
      constructor,  " Constructor method called when object is created
      " Method to set all product attributes using importing parameters
      set_attributes
        IMPORTING
          im_bicycle_id    TYPE z_018_en_bicycle_id
          im_bicycle_name  TYPE z_018_en_bicycle_name
          im_gear_type     TYPE z_018_en_gear_type
          im_frame_size_cm TYPE z_018_frame_size
          im_price         TYPE netwr_ap
          im_currency      TYPE waerk,
      get_attributes.  " Method to display all product attributes

    " Class-method to get total number of created product objects
    CLASS-METHODS: get_number_of_products
    EXPORTING no_of_products TYPE i.

  PRIVATE SECTION.    " Variables and class-data accessible only within this class
    DATA:
      bicycle_id    TYPE z_018_en_bicycle_id,
      bicycle_name  TYPE z_018_en_bicycle_name,
      gear_type     TYPE z_018_en_gear_type,
      frame_size_cm TYPE z_018_frame_size,
      price         TYPE netwr_ap,
      currency      TYPE waerk.

    CLASS-DATA: number_of_products TYPE i.  " Shared class-level variable to count
objects
ENDCLASS.
```

This part provides the actual code logic for each method declared in the definition. The **constructor** method increments the product counter. The **set_attributes** method assigns values to the product attributes, while the get_attributes method displays the stored data in a formatted way. The class method **get_number_of_products** returns the total number of created product objects.

```
"Implementation of the class methods
CLASS lcl_products IMPLEMENTATION.

  " Constructor method: increments class-level counter on each object creation
  METHOD constructor.
    number_of_products = number_of_products + 1.
  ENDMETHOD.

  " Class-method to return the number of product objects created
  METHOD get_number_of_products.
    no_of_products = number_of_products.
```

```
 ENDMETHOD.


 " Sets all the product details using the importing parameters
 METHOD set_attributes.
   bicycle_id     = im_bicycle_id.
   bicycle_name   = im_bicycle_name.
   gear_type      = im_gear_type.
   frame_size_cm  = im_frame_size_cm.
   price          = im_price.
   currency       = im_currency.
 ENDMETHOD.


 " Displays the product attributes in a structured format
 METHOD get_attributes.
   WRITE:
   / 'Product ID:     ',   bicycle_id,
   / 'Product Name:   ',   bicycle_name,
   / 'Gear Type:      ',   gear_type,
   / 'Frame Size:     ',   frame_size_cm, 'cm',
   / 'Product Price:  ',   price LEFT-JUSTIFIED,
   / 'Currency        ',   currency LEFT-JUSTIFIED.
 ENDMETHOD.


ENDCLASS.
```

7. Main Report Design and Execution

The main report includes the class and controls the program flow. Ten product objects of table **Z018_PRODUCT** were manually entered using CREATE OBJECT, and their attributes were assigned using the **SET_ATTRIBUTES** method. These objects were stored in an internal table of references, simulating a SELECT query from the **Z018_PRODUCT** table. The products are displayed in descending order by looping from the last index to the first, with each object labeled (e.g., "Object: 10") followed by its product details.

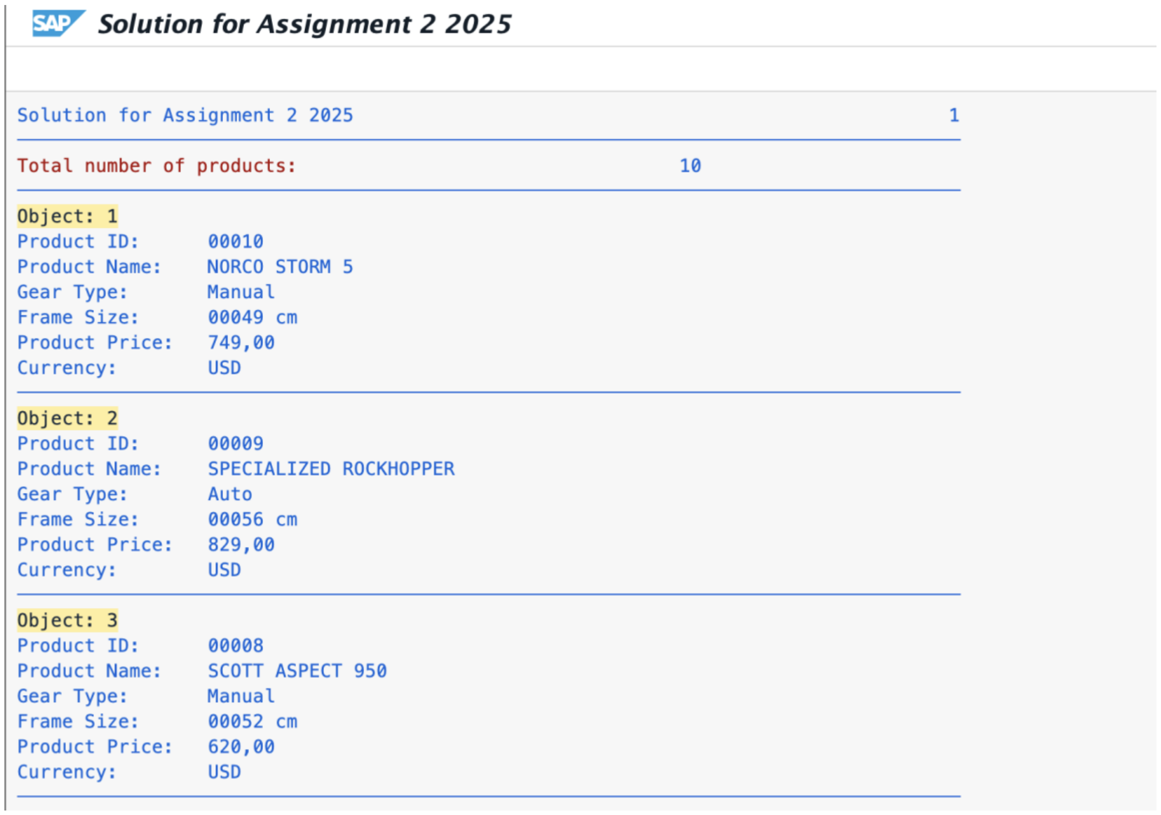
```
*&---------------------------------------------------------------------*
*& Report z018_assignment2
*&---------------------------------------------------------------------*
REPORT z018_assignment2.

" Include the local class definition and implementation from external file
INCLUDE z018_ass2_lcl_products.
" Declare variables and internal table
DATA: b_product        TYPE REF TO lcl_products,
      it_product_list   TYPE TABLE OF REF TO lcl_products,
      lv_no_of_products TYPE i.

START-OF-SELECTION.

" Create 10 product objects with attributes and store in internal table

  CREATE OBJECT b_product.
  b_product->set_attributes(
    im_bicycle_id     = '00010'
    im_bicycle_name   = 'NORCO STORM 5'
    im_frame_size_cm = '00049'
    im_gear_type      = 'Manual'
    im_price         = '749.00'
    im_currency      = 'USD' ).
  APPEND b_product TO it_product_list.

  CREATE OBJECT b_product.
  b_product->set_attributes(
    im_bicycle_id    = '00009'
    im_bicycle_name  = 'SPECIALIZED ROCKHOPPER'
    im_frame_size_cm = '00056'
    im_gear_type     = 'Auto'
    im_price         = '829.00'
    im_currency      = 'USD' ).
  APPEND b_product TO it_product_list.
  CREATE OBJECT b_product.
  b_product->set_attributes(
    im_bicycle_id    = '00008'
    im_bicycle_name  = 'SCOTT ASPECT 950'
    im_frame_size_cm = '00052'
    im_gear_type     = 'Manual'
    im_price         = '620.00'
    im_currency      = 'USD' ).
  APPEND b_product TO it_product_list.

  CREATE OBJECT b_product.
  b_product->set_attributes(
    im_bicycle_id    = '00007'
    im_bicycle_name  = 'POLYGON CASCADE 4'
    im_frame_size_cm = '00051'
    im_gear_type     = 'Auto'
    im_price         = '710.00'
    im_currency      = 'USD' ).
  APPEND b_product TO it_product_list.
    CREATE OBJECT b_product.
  b_product->set_attributes(
    im_bicycle_id    = '00006'
    im_bicycle_name  = 'CUBE AIM RACE'
    im_frame_size_cm = '00045'
    im_gear_type     = 'Manual'
    im_price         = '800.00'
    im_currency      = 'USD' ).
  APPEND b_product TO it_product_list.
  CREATE OBJECT b_product.
  b_product->set_attributes(
    im_bicycle_id    = '00005'
    im_bicycle_name  = 'TURBO JET'
    im_frame_size_cm = '00052'
    im_gear_type     = 'Auto'
    im_price         = '759.00'
    im_currency      = 'USD' ).
  APPEND b_product TO it_product_list.
  CREATE OBJECT b_product.
  b_product->set_attributes(
    im_bicycle_id    = '00004'
    im_bicycle_name  = 'CITY SWIFT'
    im_frame_size_cm = '00056'
    im_gear_type     = 'Manual'
    im_price         = '850.00'
    im_currency      = 'USD' ).
  APPEND b_product TO it_product_list.
```

```
CREATE OBJECT b_product.
b_product->set_attributes(
  im_bicycle_id    = '00003'
  im_bicycle_name  = 'ECO RIDE'
  im_frame_size_cm = '00050'
  im_gear_type     = 'Manual'
  im_price         = '800.00'
  im_currency      = 'USD' ).
APPEND b_product TO it_product_list.

CREATE OBJECT b_product.
b_product->set_attributes(
  im_bicycle_id    = '00002'
  im_bicycle_name  = 'MOUNTAIN KING'
  im_frame_size_cm = '00058'
  im_gear_type     = 'Auto'
  im_price         = '780.00'
  im_currency      = 'USD' ).
APPEND b_product TO it_product_list.

CREATE OBJECT b_product.
b_product->set_attributes(
  im_bicycle_id    = '00001'
  im_bicycle_name  = 'ROAD RUNNER '
  im_frame_size_cm = '00054'
  im_gear_type     = 'Manual'
  im_price         = '670.00'
  im_currency      = 'USD' ).
APPEND b_product TO it_product_list.
```

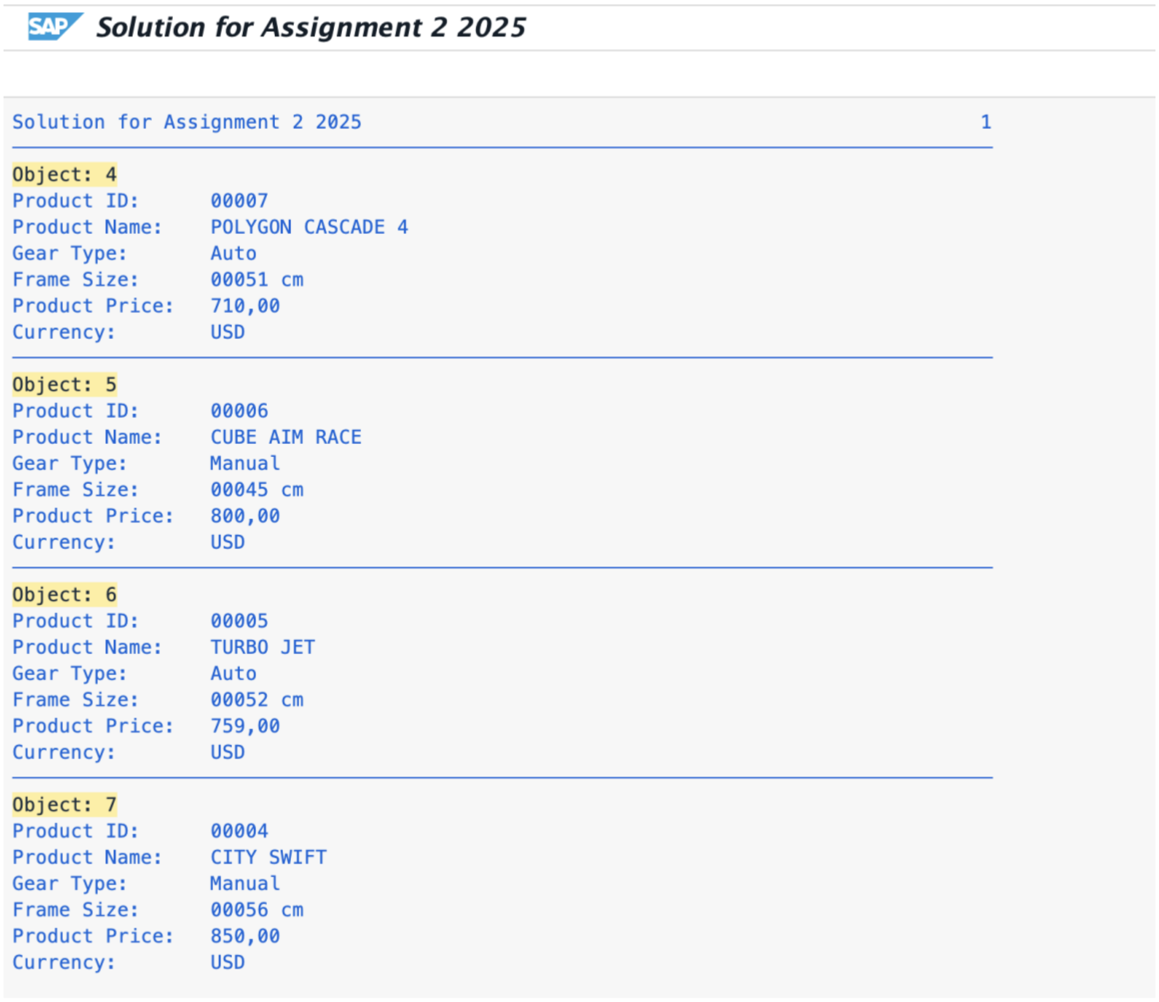
```
" Get and display total number of product objects

  CALL METHOD lcl_products=>get_number_of_products
    IMPORTING
      no_of_products = lv_no_of_products.
  WRITE: / TEXT-001 COLOR 6 INVERSE ON , lv_no_of_products.
  ULINE.

" Output each product with a counter
  DATA: lv_counter TYPE i VALUE 1.

 LOOP AT it_product_list INTO b_product.
  DATA(lv_text) = |Object: { lv_counter }|.
  WRITE: / lv_text COLOR 3.
  b_product->get_attributes( ).
  lv_counter = lv_counter + 1.
  ULINE.
ENDLOOP
```

8. Program Output and Display

The report output is generated entirely using the WRITE command. At the start of the output, the total number of created product objects is displayed using the class method **get_number_of_products**. Following this, each product's details are printed in descending order with an identifying object number and a separating line (ULINE) for better readability. The output is color-coded: for example, the object number label (like "Object: 10") is highlighted using COLOR 3. Each product's price and currency are aligned to enhance readability using the **LEFT-JUSTIFIED** formatting.

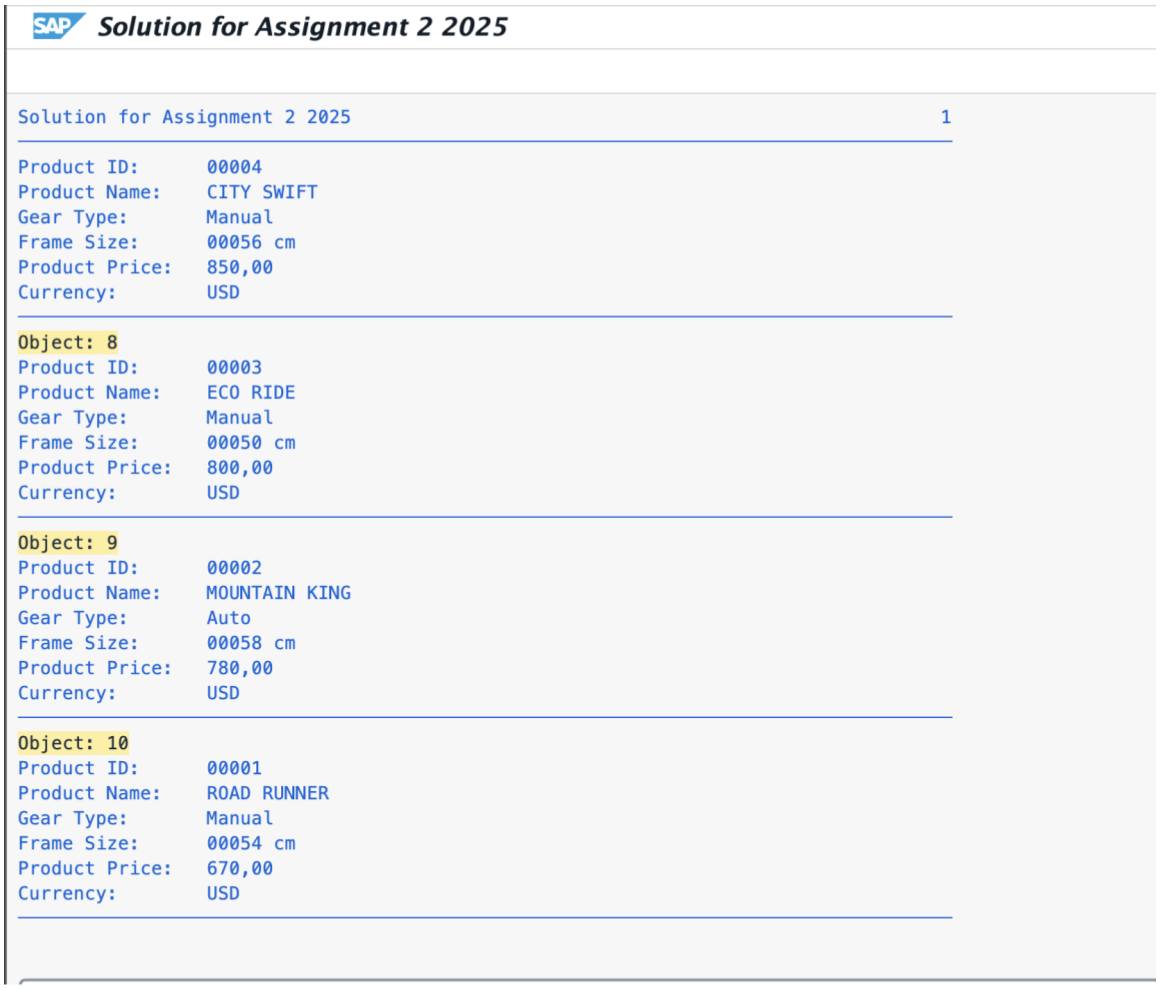


```
Solution for Assignment 2 2025

Solution for Assignment 2 2025                                                    1
------------------------------------------------------------------------------------
Total number of products:                                  10
------------------------------------------------------------------------------------
Object: 1
Product ID:      00010
Product Name:    NORCO STORM 5
Gear Type:       Manual
Frame Size:      00049 cm
Product Price:   749,00
Currency:        USD
------------------------------------------------------------------------------------
Object: 2
Product ID:      00009
Product Name:    SPECIALIZED ROCKHOPPER
Gear Type:       Auto
Frame Size:      00056 cm
Product Price:   829,00
Currency:        USD
------------------------------------------------------------------------------------
Object: 3
Product ID:      00008
Product Name:    SCOTT ASPECT 950
Gear Type:       Manual
Frame Size:      00052 cm
Product Price:   620,00
Currency:        USD
------------------------------------------------------------------------------------
```



Solution for Assignment 2 2025

```
Solution for Assignment 2 2025                                          1
--------------------------------------------------------------------------
Object: 4
Product ID:       00007
Product Name:     POLYGON CASCADE 4
Gear Type:        Auto
Frame Size:       00051 cm
Product Price:    710,00
Currency:         USD
--------------------------------------------------------------------------
Object: 5
Product ID:       00006
Product Name:     CUBE AIM RACE
Gear Type:        Manual
Frame Size:       00045 cm
Product Price:    800,00
Currency:         USD
--------------------------------------------------------------------------
Object: 6
Product ID:       00005
Product Name:     TURBO JET
Gear Type:        Auto
Frame Size:       00052 cm
Product Price:    759,00
Currency:         USD
--------------------------------------------------------------------------
Object: 7
Product ID:       00004
Product Name:     CITY SWIFT
Gear Type:        Manual
Frame Size:       00056 cm
Product Price:    850,00
Currency:         USD
```


Solution for Assignment 2 2025

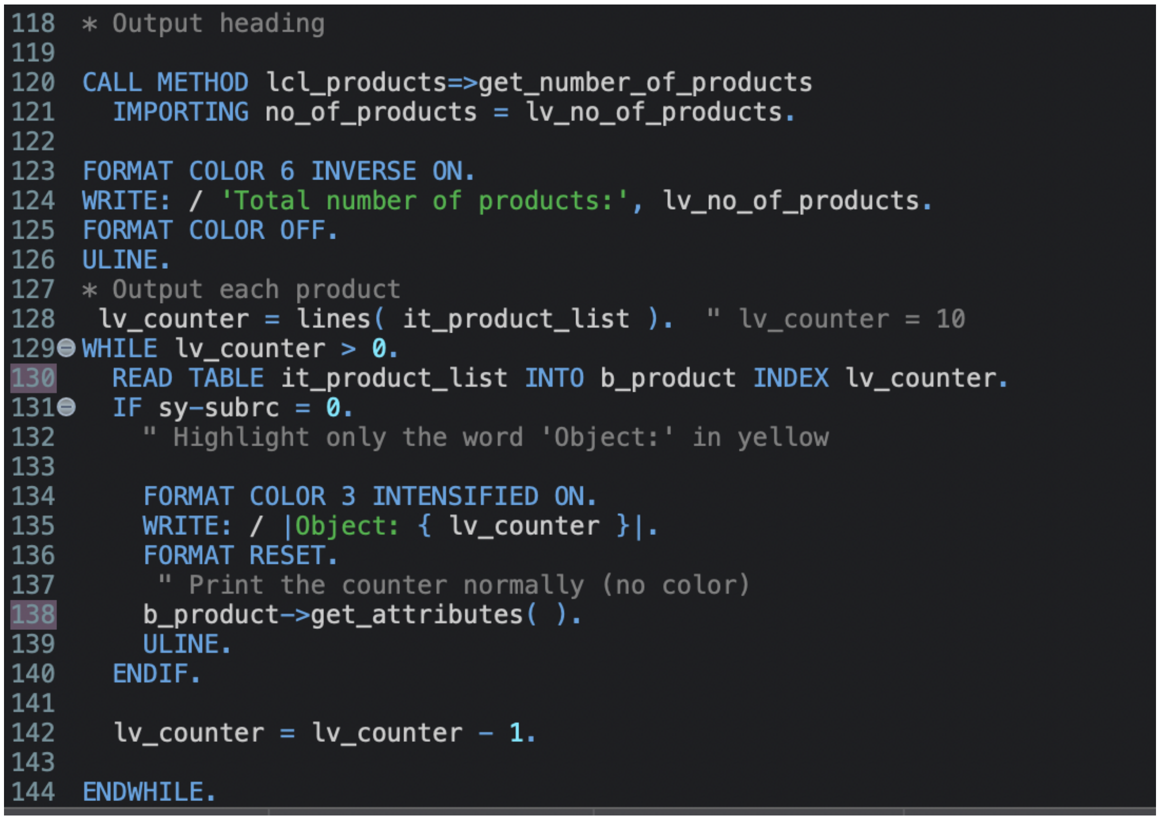
```
Solution for Assignment 2 2025                                           1
----------------------------------------------------------------------------
Product ID:       00004
Product Name:     CITY SWIFT
Gear Type:        Manual
Frame Size:       00056 cm
Product Price:    850,00
Currency:         USD
----------------------------------------------------------------------------
Object: 8
Product ID:       00003
Product Name:     ECO RIDE
Gear Type:        Manual
Frame Size:       00050 cm
Product Price:    800,00
Currency:         USD
----------------------------------------------------------------------------
Object: 9
Product ID:       00002
Product Name:     MOUNTAIN KING
Gear Type:        Auto
Frame Size:       00058 cm
Product Price:    780,00
Currency:         USD
----------------------------------------------------------------------------
Object: 10
Product ID:       00001
Product Name:     ROAD RUNNER
Gear Type:        Manual
Frame Size:       00054 cm
Product Price:    670,00
Currency:         USD
----------------------------------------------------------------------------
```

8. Testing and Validation

The program was tested successfully, and all components functioned as expected. Each of the ten manually created objects displayed the correct attribute values, confirming that the **SET_ATTRIBUTES** and **GET_ATTRIBUTES** methods worked correctly. The class counter returned a value of 10, matching the number of created instances. The descending loop correctly displayed all products from the last to the first, and the formatting and alignment were verified for visual clarity. No user input was required, but the design is robust enough to handle dynamic database inputs if needed in the future.

9. Problems Encountered and Improvements

Problem was encountered during the implementation of the output logic. Initially, the output section was written using a WHILE loop, as shown below:



```
* Output heading

CALL METHOD lcl_products=>get_number_of_products
  IMPORTING no_of_products = lv_no_of_products.

FORMAT COLOR 6 INVERSE ON.
WRITE: / 'Total number of products:', lv_no_of_products.
FORMAT COLOR OFF.
ULINE.
* Output each product
 lv_counter = lines( it_product_list ).  " lv_counter = 10
WHILE lv_counter > 0.
  READ TABLE it_product_list INTO b_product INDEX lv_counter.
  IF sy-subrc = 0.
    " Highlight only the word 'Object:' in yellow

    FORMAT COLOR 3 INTENSIFIED ON.
    WRITE: / |Object: { lv_counter }|.
    FORMAT RESET.
     " Print the counter normally (no color)
    b_product->get_attributes( ).
    ULINE.
  ENDIF.

  lv_counter = lv_counter - 1.

ENDWHILE.
```

This version was intended to display the product list in descending order by manually controlling the index with lv_counter. However, it introduced formatting issues and caused confusion in how the output was aligned, especially in relation to color highlighting for the "Object" label and inconsistent ordering when new objects were added. Additionally, in some test cases, object instances were not properly synchronized with their numbering.

To resolve these problems, the output logic was rewritten using a **LOOP AT** it_product_list **INTO** b_product statement, which offered more clarity, reduced complexity, and maintained consistent formatting.

```
" Get and display total number of product objects

 CALL METHOD lcl_products=>get_number_of_products
   IMPORTING
     no_of_products = lv_no_of_products.
 WRITE: / TEXT-001 COLOR 6 INVERSE ON , lv_no_of_products.
 ULINE.

" Output each product with a counter
 DATA: lv_counter TYPE i VALUE 1.

LOOP AT it_product_list INTO b_product.
 DATA(lv_text) = |Object: { lv_counter }|.
 WRITE: / lv_text COLOR 3.
 b_product->get_attributes( ).
 lv_counter = lv_counter + 1.
 ULINE.
ENDLOOP.
```

10. Conclusion

This assignment offered practical experience in combining class-based logic with procedural reporting. It highlighted key concepts like encapsulation, modular design, and structured output. Creating and using a custom class with constructors and methods demonstrated effective, maintainable programming aligned with real SAP practices.

11. Appendix

Full Source Code of Include (Include z018_ass2_lcl_products)

```
*& Include z018_ass2_lcl_products
*       CLASS lcl_products DEFINITION
" Definition of the local class LCL_PRODUCTS
CLASS lcl_products DEFINITION.
 PUBLIC SECTION.    " Publicly accessible methods and class methods
  METHODS:
   constructor,   " Constructor method called when object is created
   " Method to set all product attributes using importing parameters
   set_attributes
    IMPORTING
     im_bicycle_id    TYPE z_018_en_bicycle_id
     im_bicycle_name  TYPE z_018_en_bicycle_name
     im_gear_type     TYPE z_018_en_gear_type
     im_frame_size_cm TYPE z_018_frame_size
     im_price         TYPE netwr_ap
     im_currency      TYPE waerk,
   get_attributes.  " Method to display all product attributes

  " Class-method to get total number of created product objects
  CLASS-METHODS: get_number_of_products
   EXPORTING no_of_products TYPE i.

 PRIVATE SECTION.    " Variables and class-data accessible only within this class
  DATA:
      bicycle_id    TYPE z_018_en_bicycle_id,
      bicycle_name  TYPE z_018_en_bicycle_name,
      gear_type     TYPE z_018_en_gear_type,
      frame_size_cm TYPE z_018_frame_size,
      price         TYPE netwr_ap,
      currency      TYPE waerk.

    CLASS-DATA: number_of_products TYPE i.  " Shared class-level variable to count objects
ENDCLASS.

"Implementation of the class methods
CLASS lcl_products IMPLEMENTATION.

  " Constructor method: increments class-level counter on each object creation
  METHOD constructor.
    number_of_products = number_of_products + 1.
  ENDMETHOD.

  " Class-method to return the number of product objects created
  METHOD get_number_of_products.
    no_of_products = number_of_products.
  ENDMETHOD.

  " Sets all the product details using the importing parameters
  METHOD set_attributes.
    bicycle_id     = im_bicycle_id.
    bicycle_name   = im_bicycle_name.
    gear_type      = im_gear_type.
    frame_size_cm  = im_frame_size_cm.
    price          = im_price.
    currency       = im_currency.
  ENDMETHOD.
  " Displays the product attributes in a structured format
  METHOD get_attributes.
    WRITE:
    / 'Product ID:    ',   bicycle_id,
    / 'Product Name:  ',   bicycle_name,
    / 'Gear Type:     ',   gear_type,
    / 'Frame Size:    ',   frame_size_cm, 'cm',
    / 'Product Price: ',   price LEFT-JUSTIFIED,
    / 'Currency:      ',   currency LEFT-JUSTIFIED.
  ENDMETHOD.

ENDCLASS.
```