C_ABAPD_2309 Q&As

SAP Certified Associate - Back-End Developer - ABAP Cloud

Pass SAP C_ABAPD_2309 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

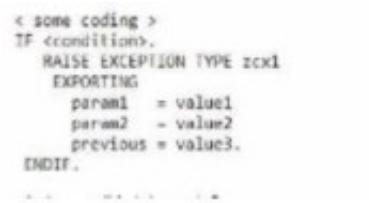
https://www.pass4itsure.com/c_abapd_2309.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by SAP Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1



(Sorry, we do not have a more clear image. If we have a better resource for the image, we will update this one immediately.)

What are valid statements? Note: There are 2 correct answers to this question.

A. "zcxl" is a dictionary structure, and "paraml" and "param2" are this structure.

B. "paraml11 and "param2" are predefined names.

C. The code creates an exception object and raises an exception.

D. "previous" expects the reference to a previous exception

Correct Answer: CD

The code snippet in the image is an example of using the RAISE EXCEPTION statement to raise a class-based exception and create a corresponding exception object. The code snippet also uses the EXPORTING addition to pass parameters to the instance constructor of the exception class12. Some of the valid statements about the code snippet are: The code creates an exception object and raises an exception: This is true. The RAISE EXCEPTION statement raises the exception linked to the exception class zcxl and generates a corresponding exception object. The exception object contains the information about the exception, such as the message, the source position, and the previous exception12. "previous" expects the reference to a previous exception: This is true. The previous parameter is a predefined parameter of the instance constructor of the exception class cx_root, which is the root class of all class-based exceptions. The previous parameter expects the reference to a previous exception objectthat was caught during exception handling. The previous parameter can beused to chain multiple exceptions and preserve the original cause of the exception12. You cannot do any of the following: "zcxl" is a dictionary structure, and "paraml" and "param2" are this structure: This is false. zcxl is not a dictionary structure, but a user-defined exception class that inherits from the predefined exception class cx_static_check. param1 and param2 are not components of this structure, but input parameters of the instance constructor of the exception class zcxl. The input parameters can be used to pass additional information to the exception object, such as the values that caused the exception12. "paraml" and "param2" are predefined names: This is false. param1 and param2 are not predefined names, but user-defined names that can be chosen arbitrarily. However, they must match the names of the input parameters of the instance constructor of the exception class zcxl. The names of the input parameters can be declared in the interface of the exception class using the RAISING addition12. References: 1: RAISE EXCEPTION - ABAP Keyword Documentation - SAP Online Help 2:

Class-Based Exceptions - ABAP Keyword Documentation - SAP Online Help

QUESTION 2

(Sorry, we do not have a more clear image. If we have a better resource for the image, we will update this one immediately.)

Using ABAP SQL, which select statement selects the mat field on line #17?

A. SELECT mat FROM Material...

B. SELECT mat FROM demo_sales_cds_so_i_ve...

C. SELECT mat FROM demo_sales_so_i...

D. SELECT mat FROM demo sales cds material ve...

Correct Answer: B

Using ABAP SQL, the select statement that selects the mat field on line #17 is:

SELECT mat FROM demo_sales_cds_so_i_ve...

This statement selects the mat field from the CDS view demo_sales_cds_so_i_ve, which is defined on line #1. The CDS view demo_sales_cds_so_i_ve is a projection view that projects the fields of the CDS view demo_sales_cds_so_i, which

is defined on line #2. The CDS view demo_sales_cds_so_i is a join view that joins the fields of the database table demo_sales_so_i, which is defined on line #3, and the CDS view demo_sales_cds_material_ve, which is defined on line #4.

The CDS view demo_sales_cds_material_ve is a value help view that provides value help for the material field of the database table demo_sales_so_i. The mat field is an alias for the material field of the database table demo_sales_so_i,

which is defined on line #91.

The other options are not valid because:

A. SELECT mat FROM Material... is not valid because Material is not a valid data source in the given code. There is no CDS view or database table named Material. C. SELECT mat FROM demo_sales_so_i... is not valid because

demo_sales_so_i is not a valid data source in the given code. There is no CDS view named demo_sales_so_i, only a database table. To access a database table, the keyword TABLE must be used, such as SELECT mat FROM TABLE demo_sales_so_i... D. SELECT mat FROM demo sales cds material ve... is not valid because demo sales cds material ve is not a valid data source in the given code. There is no CDS view or database table named demo sales cds material ve. The correct name of the CDS view is demo_sales_cds_material_ve, with underscores instead of spaces.
References: 1: Projection Views - ABAP Keyword Documentation

QUESTION 3

Exhibit:

```
1  INTERFACE if1.
2      METHODS m1.
3  ENDINTERFACE.
4
5  CLASS cl1 DEFINITION.
6  PUBLIC SECTION.
7      INTERFACES if1.
8      METHODS m2.
9  ENDCLASS.
10 ...
11 *In a method of another class
12 DATA go_if1 TYPE REF TO if1.
13 DATA go_cl1 TYPE REF TO cl1.
14 go_cl1 = NEW #(...).
15 go_if1 = go_cl1.
```

What are valid statements? Note: There are 3 correct answers to this question.

A. go_if 1 may call method ml with go_ift->ml().

B. Instead of go ell = NEW #(...) you could use go ifl = NEW cll(. ...).

C. go_cll may call method ml with go_dl->ifl-ml().

D. Instead of go_cll = NEW #() you could use go_iff - NEW #(...).

E. go_ifl may call method m2 with go if->m2(...).

Correct Answer: ABE

The following are the explanations for each statement:

A: This statement is valid. go_ifl may call method ml with go_ifl->ml(). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable go_ifl. The class cll implements the interface ifl, which means that it provides an implementation of the method ml. The data object go_ifl is assigned to a new instance of the class cll using the NEW operator and the inline declaration operator @DATA. Therefore, when go_ifl->ml() is called, the implementation of the method ml in the class cll is executed123

B: This statement is valid. Instead of go_cll = NEW #(...) you could use go_ifl = NEW cll(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the interface ifl, which

means that it is compatible with the interface ifl. Therefore, go_ifl can be assigned to a new instance of the class cll using the NEW operator and the class name cll. The inline declaration operator @DATA is optional in this case, as go_ifl is already declared. The parentheses after the class name cll can be used to pass parameters to the constructor of the class cll, if any123

E: This statement is valid. go_ifl may call method m2 with go_ifl->m2(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the interface ifl, which means that it inherits all the components of the interface ifl. The class cll also defines amethod m2, which is a public method of the class cll. Therefore, go_ifl can call the method m2 using the reference variable go_ifl. The method m2 is not defined in the interface ifl, but it is accessible through the interface ifl, as the interface ifl is implemented by the class cll. The parentheses after the method name m2 can be used to pass parameters to the method m2, if any123 The other statements are not valid, as they have syntax errors or logical errors. These statements are:

C: This statement is not valid. go_cll may call method ml with go_cll->ifl~ml(). This is because go_cll is a data object of type REF TO cll, which is a reference to the class cll. The class cll implements the interface ifl, which means that it inherits all the components of the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable go_cll. However, the syntax for calling an interface method using a class reference is go_cll->ml(), not go_cll->ifl~ml (). The interface component selector ~ is only used when calling an interface method using an interface reference, such as go_ifl->ifl~ml(). Using the interface component selector ~ with a class reference will cause a syntax error123

D: This statement is not valid. Instead of go_cll = NEW #() you could use go_ifl = NEW #(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The interface ifl cannot be instantiated, as it does not have an implementation. Therefore, go_ifl cannot be assigned to a new instance of the interface ifl using the NEW operator and the inline declaration operator @DATA. This will cause a syntax error or a runtime error. To instantiate an interface, you need to use a class that implements the interface, such as the class cll123 References: INTERFACES - ABAP Keyword Documentation, CLASS - ABAP Keyword Documentation, NEW - ABAP Keyword Documentation

QUESTION 4

When processing a loop with the statement DO... ENDDO, what system variable contains the implicit loop counter?

A. sy-linno

B. sy-labix

C. sy-subrc

D. sy-index

Correct Answer: D

Explanation: When processing a loop with the statement DO... ENDDO, the system variable that contains the implicit loop counter is sy-index. The loop counter is a numeric value that indicates how many times the loop has been executed.

The loop counter is initialized to 1 before the first execution of the loop and is incremented by 1 after each execution. The loop counter can be used to control the number of loop iterations or to access the loop elements by index. The loop

counter can also be accessed or modified within the loop body, but this is not recommended as it may cause unexpected results or errors1.

For example, the following code snippet uses the loop counter sy-index to display the numbers from 1 to 10:

DO 10 TIMES. WRITE: / sy-index. ENDDO.

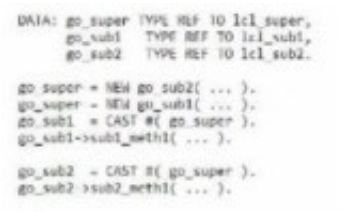
The output of this code is:

1 2 3 4 5 6 7 8 9 10

References: 1: DO - ABAP Keyword Documentation

QUESTION 5

Exhibit:



(Sorry, we do not have a more clear image. If we have a better resource for the image, we will update this one immediately.)

With Icl_super being superclass for Icl_subl and Icl_sub2 and with methods subl_methl and sub2_methl being subclass-specific methods of Id_subl or Icl_sub2, respectivel.

What will happen when executing these casts? Note: There are 2 correct answers to this question

A. go subl = CAST # go super), will not work

B. go_sub2 = CAST # go super), will work. go_subl CAST #go_super), will work

C. go_sub2 = CAST #(go_super). will not work.] go sub2->sub2 meth 1(...). will work

D. go_subl->subl_meth !(...)?w\\'ll work.

Correct Answer: AD

The following are the explanations for each statement:

A: This statement is correct. go_subl = CAST #(go_super) will not work. This is because go_subl is a data object of type REF TO cl_subl, which is a reference to the subclass cl_subl. go_super is a data object of type REF TO cl_super, which is a reference to the superclass cl_super. The CAST operator is used to perform a downcast or an upcast of a reference variable to another reference variable of a compatible type. A downcast is a conversion from a more general type to a more specific type, while an upcast is a conversion from a more specific type to a more general type. In this case, the CAST operator is trying to perform a downcast from go_super to go_subl, but this is notpossible, as go_super is not

pointing to an instance of cl_subl, but to an instance of cl_super. Therefore, the CAST operator will raise an exception CX_SY_MOVE_CAST_ERROR at runtime12

B: This statement is incorrect. go_sub2 = CAST #(go_super) will work. go_subl = CAST #(go_super) will not work. This is because go_sub2 is a data object of type REF TO cl_sub2, which is a reference to the subclass cl_sub2. go_super is a data object of type REF TO cl_super, which is a reference to the superclass cl_super. The CAST operator is used to perform a downcast or an upcast of a reference variable to another reference variable of a compatible type. A downcast is a conversion from a more general type to a more specific type, while an upcast is a conversion from a more specific type to a more general type. In this case, the CAST operator is trying to perform a downcast from go_super to go_sub2, and this is possible, as go_super is pointing to an instance of cl_sub2, which is a subclass of cl_super. Therefore, the CAST operator will assign the reference of go_super to go_sub2 without raising an exception. However, the CAST operator will not work for go_subl, as explained in statement A12

C: This statement is incorrect. go_sub2 = CAST #(go_super) will work. go_sub2- >sub2_meth1(...) will not work. This is because go_sub2 is a data object of type REF TO cl_sub2, which is a reference to the subclass cl_sub2. go_super is a data object of type REF TO cl_super, which is a reference to the superclass cl_super. The CAST operator is used to perform a downcast or an upcast of a reference variable to another reference variable of a compatible type. A downcast is a conversion from a more general type to a more specific type, while an upcast is a conversion from a more specific type to a more general type. In this case, the CAST operator is trying to perform a downcast from go_super to go_sub2, and this is possible, as go_super is pointing to an instance of cl_sub2, which is a subclass of cl_super. Therefore, the CAST operator will assign the reference of go_super to go_sub2 without raising an exception. However, the method call go_sub2- >sub2_meth1(...) will not work, as sub2_meth1 is a subclass-specific method of cl_sub2, which is not inherited by cl_super. Therefore, the method call will raise an exception CX_SY_DYN_CALL_ILLEGAL_METHOD at runtime123

D: This statement is correct. go_subl->subl_meth1(...) will work. This is because go_subl is a data object of type REF TO cl_subl, which is a reference to the subclass cl_subl. subl_meth1 is a subclass-specific method of cl_subl, which is not inherited by cl_super. Therefore, the method call go_subl->subl_meth1(...) will work, as go_subl is pointing to an instance of cl_subl, which has the method subl_meth1123 References: NEW - ABAP Keyword Documentation, CAST - ABAP Keyword Documentation, Method Call - ABAP Keyword Documentation