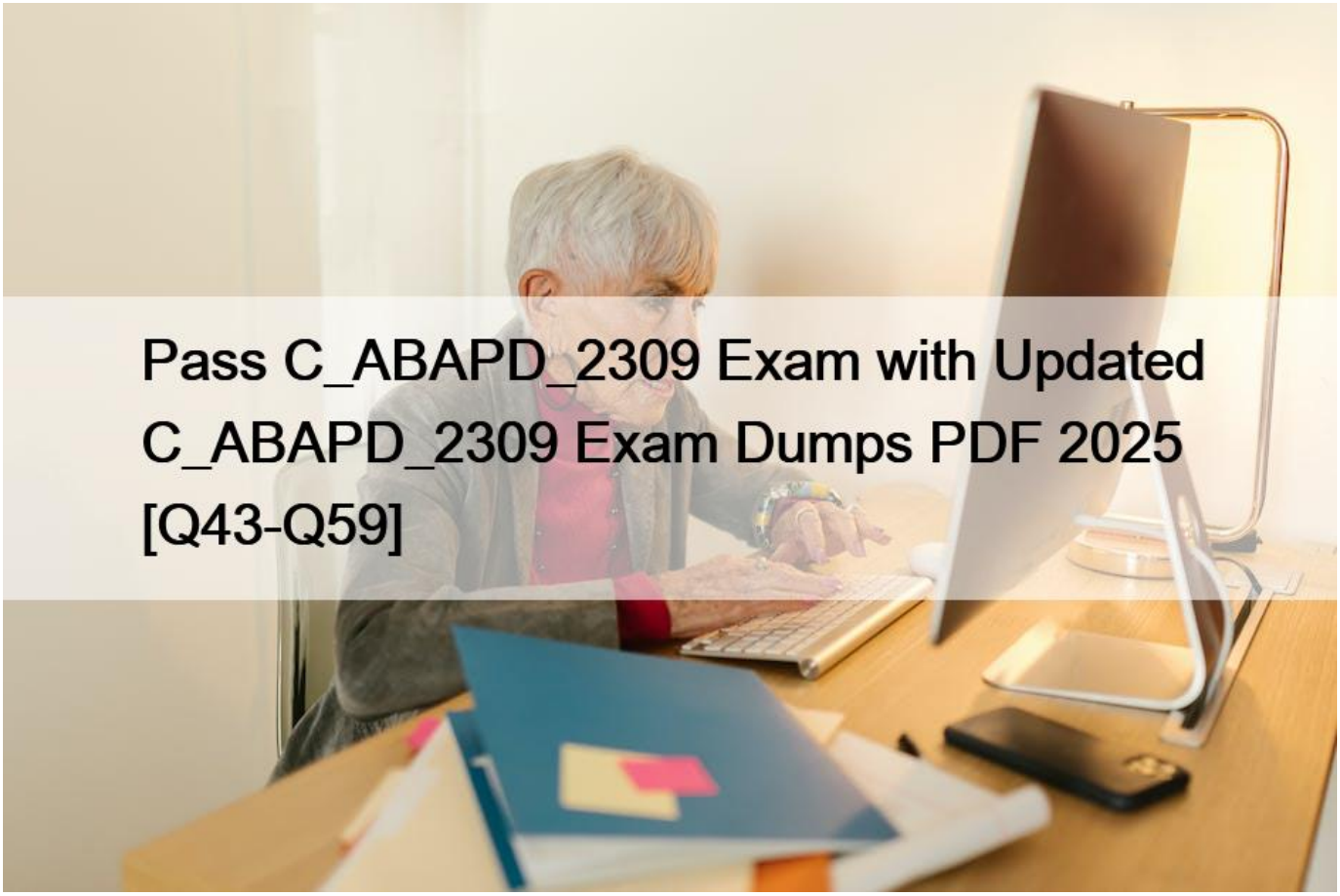
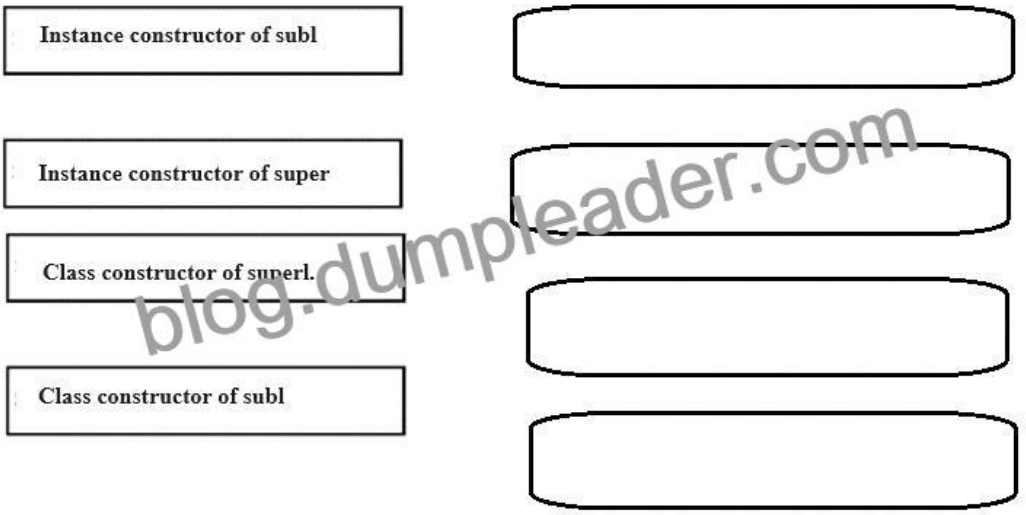
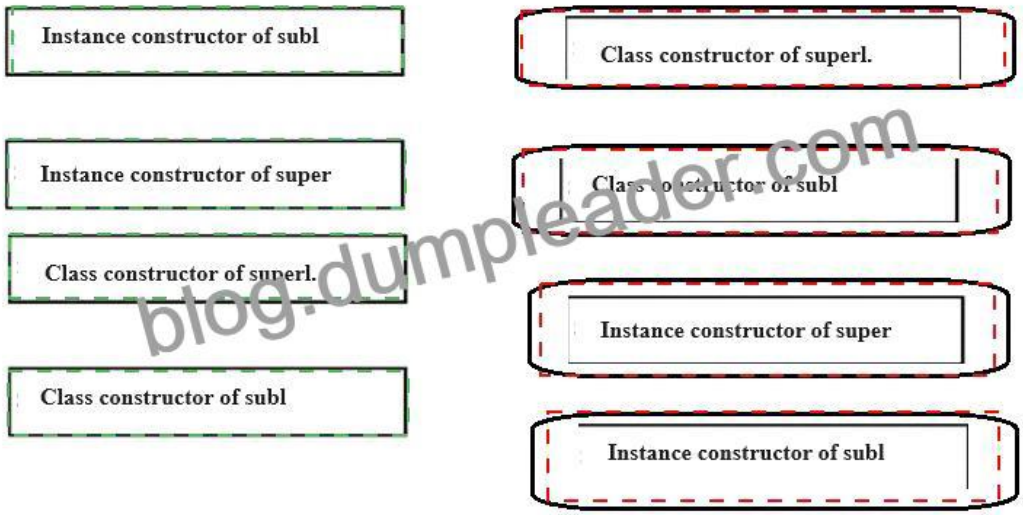
Pass C_ABAPD_2309 Exam with Updated C_ABAPD_2309 Exam Dumps PDF 2025 [Q43-Q59



Pass C_ABAPD_2309 Exam with Updated C_ABAPD_2309 Exam Dumps PDF 2025
C_ABAPD_2309 Exam Dumps - Free Demo & 365 Day Updates

Q43. You have a superclass superl and a subclass subl of superl. Each class has an instance constructor and a static constructor. The first statement of your program creates an instance of subl. In which sequence will the constructors be executed?





Explanation

The sequence in which the constructors will be executed is as follows:

Class constructor of superl. This is because the class constructor is a static method that is executed automatically before the class is accessed for the first time. The class constructor is used to initialize the static attributes and components of the class. The class constructor of the superclass is executed before the class constructor of the subclass, as the subclass inherits the static components of the superclass12 Class constructor of subl. This is because the class constructor is a static method that is executed automatically before the class is accessed for the first time. The class constructor is used to initialize the static attributes and components of the class. The class constructor of the subclass is executed after the class constructor of the superclass, as the subclass inherits the static components of the superclass12 Instance constructor of superl. This is because the instance constructor is an instance method that is executed automatically when an instance of the class is created using the statement CREATE OBJECT.

The instance constructor is used to initialize the instance attributes and components of the class. The instance constructor of the superclass is executed before the instance constructor of the subclass, as the subclass inherits the instance components of the superclass. The instance constructor of the subclass must call the instance constructor of the superclass explicitly using super->constructor, unless the superclass is the root node object12 Instance constructor of subl. This is because the instance constructor is an instance method that is executed automatically when an instance of the class is created using the statement CREATE OBJECT.

The instance constructor is used to initialize the instance attributes and components of the class. The instance constructor of the subclass is executed after the instance constructor of the superclass, as the subclass inherits the instance components of the superclass. The instance constructor of the subclass must call the instance constructor of the superclass explicitly using super->constructor, unless the superclass is the root node object12 References: Constructors of Classes – ABAP Keyword Documentation, METHODS – constructor – ABAP Keyword Documentation

Q44. For what kind of applications would you consider using on-stack developer extensions? Note: There are 2 correct answers to this question.
* Applications that provide APIs for side by side SAP BTP apps
* Applications that access SAP S/4HANA data using complex SQL
* Applications that integrate data from several different systems
* Applications that run separate from SAP S/4HANA
Explanation

On-stack developer extensibility is a type of extensibility that allows you to create development projects directly on the SAP S/4HANA Cloud technology stack. It gives you the opportunity to develop cloud-ready and upgrade-stable custom ABAP applications and services inside the SAP S/4HANA Cloud, public edition system. You can use the ABAP Development Tools in Eclipse to create and deploy your on-stack extensions.

On-stack developer extensibility is suitable for the following kinds of applications:

Applications that provide APIs for side by side SAP BTP apps. On-stack developer extensibility allows you to create OData services or RESTful APIs based on CDS view entities or projection views. These services or APIs can expose SAP S/4HANA data and logic to other applications that run on the SAP Business Technology Platform (SAP BTP) or other platforms. This way, you can create a loosely coupled integration between your SAP S/4HANA system and your side by side SAP BTP apps.

Applications that access SAP S/4HANA data using complex SQL. On-stack developer extensibility allows you to use ABAP SQL to access SAP S/4HANA data using complex queries, such as joins, aggregations, filters, parameters, and code pushdown techniques. You can also use ABAP SQL to perform data manipulation operations, such as insert, update, delete, and upsert. This way, you can create applications that require advanced data processing and analysis on SAP S/4HANA data.

The other kinds of applications are not suitable for on-stack developer extensibility, as they have different requirements and challenges. These kinds of applications are:

Applications that integrate data from several different systems. On-stack developer extensibility is not meant for creating applications that integrate data from multiple sources, such as other SAP systems, third-party systems, or cloud services. This is because on-stack developer extensibility does not support remote access or data replication, and it may cause performance or security issues. For this kind of applications, you should use side by side extensibility, which allows you to create applications that run on the SAP BTP and communicate with the SAP S/4HANA system via public APIs or events.

Applications that run separate from SAP S/4HANA. On-stack developer extensibility is not meant for creating applications that run independently from the SAP S/4HANA system, such as standalone apps, microservices, or web apps. This is because on-stack

developer extensibility requires a tight coupling with the SAP S/4HANA system, and it may limit the scalability, flexibility, and portability of the applications. For this kind of applications, you should use side by side extensibility, which allows you to create applications that run on the SAP BTP and leverage the cloud-native features and services of the platform.

References: Developer Extensibility in SAP S/4HANA Cloud ABAP Environment, SAP S/4HANA Extensibility – Simplified Guide for Beginners

Q45. Which RESTful Application Programming object can be used to organize the display of fields in an app?
* Data model view
* Metadata extension
* Service definition
* Projection view

A metadata extension is a RESTful Application Programming object that can be used to organize the display of fields in an app. A metadata extension is a CDS view that annotates another CDS view with UI annotations, such as labels, icons, or facets. These annotations define how the data should be presented in the app, such as which fields should be shown on the object page, which fields should be editable, or which fields should be used for filtering or sorting. A metadata extension can also be used to add custom actions or validations to the app12. References: 1: Refine the Object Page with Annotations | SAP Tutorials 2: ABAP RAP : Enabling custom actions with a dialog for additional input fields | SAP Blogs

Q46. Which of the following string functions are predicate functions? Note: There are 2 correct answers to this question.
* find_any_not_of()
* contains_any_of()
* count_any_of()
* matchesQ

Explanation

String functions are expressions that can be used to manipulate character-like data in ABAP. String functions can be either predicate functions or non-predicate functions. Predicate functions are string functions that return a truth value (true or false) for a condition of the argument text. Non-predicate functions are string functions that return a character-like result for an operation on the argument text1.

The following string functions are predicate functions:

B). contains_any_of(): This function returns true if the argument text contains at least one of the characters specified in the character set. For example, the following expression returns true, because the text ‘ABAP’ contains at least one of the characters ‘A’, ‘B’, or ‘C’:

contains_any_of(val = ‘ABAP’ set = ‘ABC’).

D). matches(): This function returns true if the argument text matches the pattern specified in the regular expression. For example, the following expression returns true, because the text ‘ABAP’ matches the pattern that consists of four uppercase letters:

matches(val = ‘ABAP’ regex = ‘[A-Z]{4}’).

The following string functions are not predicate functions, because they return a character-like result, not a truth value:

A). find_any_not_of(): This function returns the position of the first character in the argument text that is not contained in the character set. If no such character is found, the function returns 0. For example, the following expression returns 3, because the third character of the text ‘ABAP’ is not contained in the character set ‘ABC’:

find_any_not_of(val = ‘ABAP’ set = ‘ABC’).

C). count_any_of(): This function returns the number of characters in the argument text that are contained in the character set. For example, the following expression returns 2, because there are two characters in the text ‘ABAP’ that are contained in the character set ‘ABC’:

count_any_of(val = ‘ABAP’ set = ‘ABC’).

References: 1: String Functions – ABAP Keyword Documentation

Q47. Which extensibility type does SAP recommend you use to enhance the existing UI for an SAP Fiori app?
* Key user
* Classic
* Side-by-side
* Developer

According to the SAP clean core extensibility and ABAP cloud topic, SAP recommends using developer extensibility to enhance the existing UI for an SAP Fiori app. Developer extensibility allows you to use the UI adaptation editor in SAP Web IDE to modify the UI layout, add or remove fields, and bind them to the data model. You can also use the SAPUI5 framework to create custom controls, views, and controllers. Developer extensibility is based on the in-app extensibility concept, which means that the extensions are part of the same application and are deployed together with the app. Developer extensibility requires developer skills and access to the source code of the app. References: SAP Learning Hub, SAP S/4HANA Cloud Extensibility – In-App Extensibility, SAP Fiori: Extensibility

Q48. What RESTful Application Programming feature is used to ensure the uniqueness of a semantic key?
* Validation
* Action
* Determination

The RESTful Application Programming feature that is used to ensure the uniqueness of a semantic key is determination. A determination is a type of behavior implementation that defines a logic that is executed automatically when certain events occur, such as create, update, delete, or activate. A determination can be used to calculate or derive values for certain fields, such as semantic keys, based on other fields or external sources. A determination can also be used to check the uniqueness of a semantic key by comparing it with the existing values in the database or the transaction buffer. A determination can use the ABAP SQL or the EML syntax to access and manipulate data. A determination can be defined using the DETERMINE action clause in the behavior definition of a CDS view entity or a projection view. A determination can also be annotated with the @ObjectModel.determination annotation to specify the event, the timing, and the scope of the determination12 The other RESTful Application Programming features are not used to ensure the uniqueness of a semantic key, but have different purposes and effects. These features are:

* Validation: A validation is a type of behavior implementation that defines a logic that is executed automatically when certain events occur, such as create, update, delete, or activate. A validation can be used to check the consistency and correctness of the data, such as mandatory fields, data types, value ranges, or business rules. A validation can use the ABAP SQL or the EML syntax to access and manipulate data. A validation can be defined using the VALIDATE action clause in the behavior definition of a CDS view entity or a projection view. A validation can also be annotated with the

@ObjectModel.validation annotation to specify the event, the timing, and the scope of the validation12

* Action: An action is a type of behavior implementation that defines a logic that is executed explicitly by the user or the application. An action can be used to perform a specific business operation, such as creating, updating, deleting, or activating an entity instance, or triggering a workflow or a notification.

An action can use the ABAP SQL or the EML syntax to access and manipulate data. An action can be defined using the ACTION clause in the behavior definition of a CDS view entity or a projection view. An action can also be annotated with the @ObjectModel.action annotation to specify the name, the description, the parameters, and the visibility of the action12 References: Behavior Implementation – ABAP Keyword Documentation, Behavior Definition – ABAP Keyword Documentation

Q49. In ABAP SQL, which of the following can be assigned an alias? Note: There are 2 correct answers to this question.
* order criterion (from order by clause)
* field (from field list)
* database table
* group criterion (from group by clause)

In ABAP SQL, an alias is a temporary name that can be assigned to a field or a database table in a query. An alias can be used to make the query more readable, to avoid name conflicts, or to access fields or tables with long names. An alias is created with the AS keyword and is only valid for the duration of the query1.

The following are examples of how to assign an alias to a field or a database table in ABAP SQL:

B) field (from field list): A field is a column of a table or a view that contains data of a certain type. A field can be assigned an alias in the field list of a SELECT statement, which specifies the fields that are selected from the data source. For example, the following query assigns the alias name to the field carrname of the table scarr:

SELECT carrid, carrname AS name FROM scarr.

The alias name can be used instead of carrname in other clauses of the query, such as WHERE, GROUP BY, ORDER BY, and so on2.

C) database table: A database table is a collection of data that is organized in rows and columns. A database table can be assigned an alias in the FROM clause of a SELECT statement, which specifies the data source that is selected from. For example, the following query assigns the alias c to the table scarr:

SELECT c.carrid, c.carrname FROM scarr AS c.

The alias c can be used instead of scarr in other clauses of the query, such as WHERE, JOIN, GROUP BY, ORDER BY, and so on3.

The following are not valid for assigning an alias in ABAP SQL:

A) order criterion (from order by clause): An order criterion is a field or an expression that is used to sort the result set of a query in ascending or descending order. An order criterion cannot be assigned an alias in the ORDER BY clause of a SELECT statement, because the alias is not visible in this clause. The alias can only be used in the clauses that follow the clause where it is defined1.

D) group criterion (from group by clause): A group criterion is a field or an expression that is used to group the result set of a query into subsets that share the same values. A group criterion cannot be assigned an alias in the GROUP BY clause of a SELECT statement, because the alias is not visible in this clause. The alias can only be used in the clauses that follow the clause where it is defined1.

Q50. What are some of the reasons that Core Data Services are preferable to the classical approach to data modeling? Note: There are 2 correct answers to this question.
* They implement code pushdown.
* They avoid data transfer completely.
* They transfer computational results to the application server.
* They compute results on the application server.

Core Data Services (CDS) are preferable to the classical approach to data modeling for several reasons, but two of them are:

They implement code pushdown. Code pushdown is the principle of moving data-intensive logic from the application server to the database server, where the data resides. This reduces the data transfer between the application server and the database server, which improves the performance and scalability of the application. CDS enable code pushdown by allowing the definition of semantic data models and business logic in the database layer, using SQL and SQL-based expressions1.

They transfer computational results to the application server. CDS allow the application server to access the data and the logic defined in the database layer by using Open SQL statements. Open SQL is a standardized and simplified subset of SQL that can be used across different database platforms. Open SQL statements are translated into native SQL statements by the ABAP runtime environment and executed on the database server. The results of the computation are then transferred to the application server, where they can be further processed or displayed2.

Q51. What would be the correct expression to change a given string value ‘mr joe doe’ into ‘JOE’ in an ABAP SQL field list?
* SELECT FROM TABLE dbtabl FIELDS

Of1,

upper(left(‘mr joe doe’, 6)) AS f2_up_left, f3,
* SELECT FROM TABLE dbtabl FIELDS

Of1,

left(lower(substring(‘mr joe doe’, 4, 3)), 3) AS f2_left_lo_sub, f3,
* SELECT FROM TABLE dbtabl FIELDS

Of1,

substring(upper(‘mr joe doe’), 4, 3) AS f2_sub_up, f3,…
* SELECT FROM TABLE dbtabl FIELDS

Of1,

substring(lower(upper(‘mr joe doe’)), 4, 3) AS f2_sub_lo_up, f3,
Explanation

The correct expression to change a given string value ‘mr joe doe’ into ‘JOE’ in an ABAP SQL field list is

C). SELECT FROM TABLE dbtabl FIELDS Of1, substring(upper(‘mr joe doe’), 4, 3) AS f2_sub_up, f3,…

This expression uses the following SQL functions for strings12:

upper: This function converts all lowercase characters in a string to uppercase. For example, upper(‘mr joe doe’) returns ‘MR JOE DOE’.

substring: This function returns a substring of a given string starting from a specified position and with a specified length. For example, substring(‘MR JOE DOE’, 4, 3) returns ‘JOE’.

AS: This keyword assigns an alias or a temporary name to a field or an expression in the field list. For example, AS f2_sub_up assigns the name f2_sub_up to the expression substring(upper(‘mr joe doe’), 4, 3).

You cannot do any of the following:

A). SELECT FROM TABLE dbtabl FIELDS Of1, upper(left(‘mr joe doe’, 6)) AS f2_up_left, f3,…:

This expression uses the wrong SQL function for strings to get the desired result. The left function returns the leftmost characters of a string with a specified length, ignoring the trailing blanks. For example, left(‘mr joe doe’, 6) returns ‘mr joe’. Applying the upper function to this result returns ‘MR JOE’, which is not the same as ‘JOE’.

B). SELECT FROM TABLE dbtabl FIELDS Of1, left(lower(substring(‘mr joe doe’, 4, 3)), 3) AS f2_left_lo_sub, f3,…: This expression uses unnecessary and incorrect SQL functions for strings to get the desired result. The lower function converts all uppercase characters in a string to lowercase. For example, lower(substring(‘mr joe doe’, 4, 3)) returns ‘joe’. Applying the left function to this result with the same length returns ‘joe’ again, which is not the same as ‘JOE’.

D). SELECT FROM TABLE dbtabl FIELDS Of1, substring(lower(upper(‘mr joe doe’)), 4, 3) AS f2_sub_lo_up, f3,…: This expression uses unnecessary and incorrect SQL functions for strings to get the desired result. The lower function converts all uppercase characters in a string to lowercase, and the upper function converts all lowercase characters in a string to uppercase. Applying both functions to the same string cancels out the effect of each other and returns the original string. For example, lower(upper(‘mr joe doe’)) returns ‘mr joe doe’.

Applying the substring function to this result returns ‘joe’, which is not the same as ‘JOE’.

References: 1: SQL Functions for Strings – ABAP Keyword Documentation – SAP Online Help 2: sql_func – String Functions – ABAP Keyword Documentation – SAP Online Help

Q52. Refer to the Exhibit.

The class zcl_demo_class is in a software component with the language version set to “Standard ABAP”. The function module “ZF11 is in a software component with the language version set to “ABAP Cloud”. Both the class and function module are customer created. Regarding line #6, which of the following is a valid statement?

* ‘ZF1’ can be called whether it has been released or not for cloud development.
* ‘ZF1’ can be called via a wrapper that itself has been released for cloud development.
* ‘ZF1’ can be called via a wrapper that itself has not been released for cloud development.
* ‘ZF1’ must be released for cloud development to be called.

The function module ZF1 is in a software component with the language version set to “ABAP Cloud”. This means that it follows the ABAP Cloud Development Model, which requires the usage of public SAP APIs and extension points to access SAP functionality and data. These APIs and extension points are released by SAP and documented in the SAP API Business Hub1. Customer-created function modules are not part of the public SAP APIs and are not released for cloud development. Therefore, calling a function module directly from a class with the language version set to “Standard ABAP” is not allowed and will result in a syntax error. However, there is a possible way to call a function module indirectly from a class with the language version set to “Standard ABAP”:

Create a wrapper class or interface for the function module and release it for cloud development. A wrapper is a class or interface that encapsulates the function module and exposes its functionality through public methods or attributes. The wrapper must be created in a software component with the language version set to “ABAP Cloud” and must be marked as released for cloud development using the annotation @EndUserText.label. The wrapper can then be called from a class with the language version set to “Standard ABAP” using the public methods or attributes2.

For example, the following code snippet shows how to create a wrapper class for the function module ZF1 and call it from the class zcl_demo_class:

@EndUserText.label: ‘Wrapper for ZF1’ CLASS zcl_wrapper_zf1 DEFINITION PUBLIC FINAL CREATE PUBLIC. PUBLIC SECTION. CLASS-METHODS: call_zf1 IMPORTING iv_a TYPE i iv_b TYPE i EXPORTING ev_result TYPE i. ENDCLASS.

CLASS zcl_wrapper_zf1 IMPLEMENTATION. METHOD call_zf1. CALL FUNCTION ‘ZF1’ EXPORTING a = iv_a b = iv_b IMPORTING result = ev_result. ENDMETHOD. ENDCLASS.

CLASS zcl_demo_class DEFINITION. METHODS: m1. ENDCLASS.

CLASS zcl_demo_class IMPLEMENTATION. METHOD m1. DATA(lv_result) = zcl_wrapper_zf1=>call_zf1(iv_a = 2 iv_b = 3). WRITE: / lv_result. ENDMETHOD. ENDCLASS.

The output of this code is:

5

Q53. In an Access Control Object, which clauses are used? Note: There are 3 correct answers to this question.
* Where (to specify the access conditions)
* Crant (to identify the data source)
* Return code (to assign the return code of the authority check)
* Define role (to specify the role name)
* Revoke (to remove access to the data source)

An Access Control Object (ACO) is a CDS annotation that defines the access control rules for a CDS view entity. An ACO consists of one or more clauses that specify the role name, the data source, the access conditions, and the return code of the authority check12. Some of the clauses that are used in an ACO are:

Where (to specify the access conditions): This clause is used to define the logical expression that determines whether a user has access to the data source or not. The expression can use the fields of the data source, the parameters of the CDS view entity, or the predefined variables $user and $session. The expression can also use the functions check_authorization and check_role to perform additional authority checks12.

Define role (to specify the role name): This clause is used to assign a name to the role that is defined by the ACO. The role name must be unique within the namespace of the CDS view entity and must not contain any special characters. The role name can be

used to reference the ACO in other annotations, such as @AccessControl.authorizationCheck or @AccessControl.grant12.

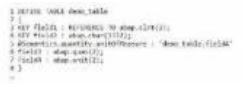
Revoke (to remove access to the data source): This clause is used to explicitly deny access to the data source for a user who meets the conditions of the where clause. The revoke clause overrides any grant clause that might grant access to the same user. The revoke clause can be used to implement the principle of least privilege or to enforce data segregation12.

You cannot do any of the following:

Grant (to identify the data source): This is not a valid clause in an ACO. The grant clause is a separate annotation that is used to grant access to a CDS view entity or a data source for a user who has a specific role. The grant clause can reference an ACO by its role name to apply the access conditions defined by the ACO12.

Return code (to assign the return code of the authority check): This is not a valid clause in an ACO. The return code of the authority check is a predefined variable that is set by the system after performing the access control check. The return code can be used in the where clause of the ACO to specify different access conditions based on the outcome of the check12.

Q54. Which field is defined incorrectly?



* field1
* field3
* field4
* field2

The field4 is defined incorrectly in the ABAP code snippet. The reason is that the data type c (character) cannot have a decimal places specification. The decimal places specification is only valid for the data types p (packed number) and f (floating point number)1. Therefore, the field4 definition should either omit the decimal places specification or change the data type to p or f.

Q55. In ABAP SQL, which of the following can be assigned an alias? Note: There are 2 correct answers to this question.

* order criterion (from order by clause)
* field (from field list)
* database table
* group criterion (from group by clause)

In ABAP SQL, an alias is a temporary name that can be assigned to a field or a database table in a query. An alias can be used to make the query more readable, to avoid name conflicts, or to access fields or tables with long names. An alias is created with the AS keyword and is only valid for the duration of the query1.

The following are examples of how to assign an alias to a field or a database table in ABAP SQL:

* B. field (from field list): A field is a column of a table or a view that contains data of a certain type. A field can be assigned an alias in the field list of a SELECT statement, which specifies the fields that are

* selected from the data source. For example, the following query assigns the alias name to the field carrname of the table scarr:

SELECT carrid, carrname AS name FROM scarr.

The alias name can be used instead of carrname in other clauses of the query, such as WHERE, GROUP BY, ORDER BY, and so on2.

* C. database table: A database table is a collection of data that is organized in rows and columns. A database table can be assigned an alias in the FROM clause of a SELECT statement, which specifies the data source that is selected from. For example, the following query assigns the alias c to the table scarr:

SELECT c.carrid, c.carrname FROM scarr AS c.

The alias c can be used instead of scarr in other clauses of the query, such as WHERE, JOIN, GROUP BY, ORDER BY, and so on3.

The following are not valid for assigning an alias in ABAP SQL:

* A. order criterion (from order by clause): An order criterion is a field or an expression that is used to sort the result set of a query in ascending or descending order. An order criterion cannot be assigned an alias in the ORDER BY clause of a SELECT statement, because the alias is not visible in this clause. The alias can only be used in the clauses that follow the clause where it is defined1.

* D. group criterion (from group by clause): A group criterion is a field or an expression that is used to group the result set of a query into subsets that share the same values. A group criterion cannot be assigned an alias in the GROUP BY clause of a SELECT statement, because the alias is not visible in this clause. The alias can only be used in the clauses that follow the clause where it is defined1.

References: 1: ALIASES – ABAP Keyword Documentation 2: SELECT List – ABAP Keyword Documentation 3: FROM Clause – ABAP Keyword Documentation

Q56. In an Access Control Object, which clauses are used? Note: There are 3 correct answers to this question.
* Where (to specify the access conditions)
* Crant (to identify the data source)
* Return code (to assign the return code of the authority check)
* Define role (to specify the role name)
* Revoke (to remove access to the data source)

Explanation

An Access Control Object (ACO) is a CDS annotation that defines the access control rules for a CDS view entity. An ACO consists of one or more clauses that specify the role name, the data source, the access conditions, and the return code of the authority check12. Some of the clauses that are used in an ACO are:

Where (to specify the access conditions): This clause is used to define the logical expression that determines whether a user has access to the data source or not. The expression can use the fields of the data source, the parameters of the CDS view entity, or the predefined variables $user and $session. The expression can also use the functions check_authorization and check_role to perform additional authority checks12.

Define role (to specify the role name): This clause is used to assign a name to the role that is defined by the ACO. The role name

must be unique within the namespace of the CDS view entity and must not contain any special characters. The role name can be used to reference the ACO in other annotations, such as @AccessControl.authorizationCheck or @AccessControl.grant12.

Revoke (to remove access to the data source): This clause is used to explicitly deny access to the data source for a user who meets the conditions of the where clause. The revoke clause overrides any grant clause that might grant access to the same user. The revoke clause can be used to implement the principle of least privilege or to enforce data segregation12.

You cannot do any of the following:

Grant (to identify the data source): This is not a valid clause in an ACO. The grant clause is a separate annotation that is used to grant access to a CDS view entity or a data source for a user who has a specific role. The grant clause can reference an ACO by its role name to apply the access conditions defined by the ACO12.

Return code (to assign the return code of the authority check): This is not a valid clause in an ACO. The return code of the authority check is a predefined variable that is set by the system after performing the access control check. The return code can be used in the where clause of the ACO to specify different access conditions based on the outcome of the check12.

References: 1: Access Control Objects – ABAP Keyword Documentation – SAP Online Help 2: Access Control in Core Data Services (CDS) | SAP Help Portal

Q57. When processing a loop with the statement DO… ENDDO, what system variable contains the implicit loop counter?
* sy-linno
* sy-labix
* sy-subrc
* sy-index

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

Q58. Refer to the Exhibit.

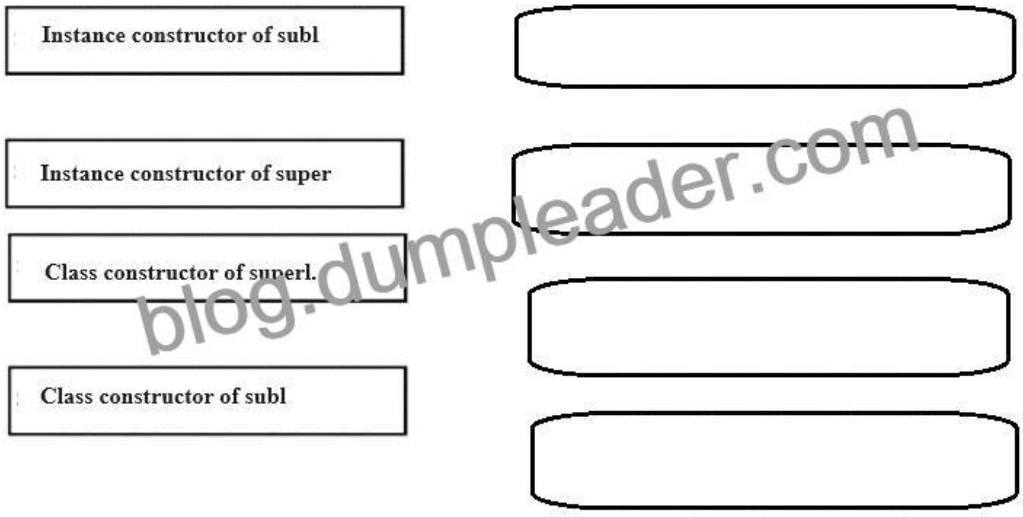
```
@AccessControl.authorizationCheck: #NOT_REQUIRED
DEFINE VIEW ENTITY demo_flight_info_union AS
    SELECT FROM Scustom
    {
        KEY id,
        KEY 'Customer' AS partner,
        name,
        city,
        country
    }
    UNION
        SELECT FROM stravelag
        {
            KEY agencynum AS id,
                'Agency'  AS partner,
                name,
                city,
                country
        }
```

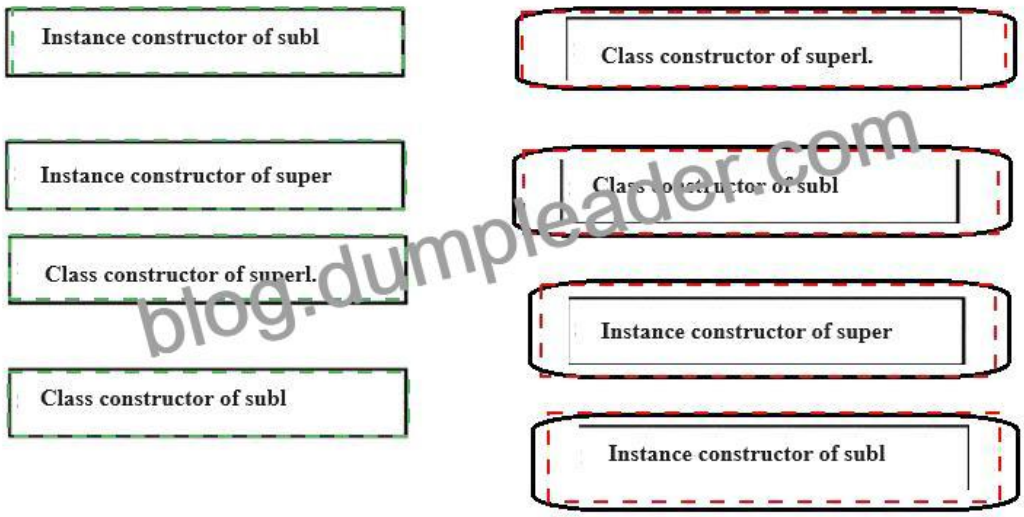

when you attempt to activate the definition, what will be the response?

* Activation error because the field names of the union do not match
* Activation error because the field types of the union do not match
* Activation error because the key fields of the union do not match
* Activation successful

The response will be an activation error because the field names of the union do not match. This is because the field names of the union must match in order for the definition to be activated. The union operator combines the result sets of two or more queries into a single result set. The queries that are joined by the union operator must have the same number and type of fields, and the fields must have the same names1. In the given code, the field names of the union do not match, because the first query has the fields carrname, connid, cityfrom, and cityto, while the second query has the fields carrname, carrier_id, cityfrom, and cityto. The field connid in the first query does not match the field carrier_id in the second query. Therefore, the definition cannot be activated.

Q59. You have a superclass superl and a subclass subl of superl. Each class has an instance constructor and a static constructor. The first statement of your program creates an instance of subl. In which sequence will the constructors be executed?





Explanation:

The sequence in which the constructors will be executed is as follows:

* Class constructor of superl. This is because the class constructor is a static method that is executed automatically before the class is accessed for the first time. The class constructor is used to initialize the static attributes and components of the class. The class constructor of the superclass is executed before the class constructor of the subclass, as the subclass inherits the static components of the superclass12

* Class constructor of subl. This is because the class constructor is a static method that is executed automatically before the class is accessed for the first time. The class constructor is used to initialize the static attributes and components of the class. The class constructor of the subclass is executed after the class constructor of the superclass, as the subclass inherits the static components of the superclass12

* Instance constructor of superl. This is because the instance constructor is an instance method that is executed automatically when an instance of the class is created using the statement CREATE OBJECT.

The instance constructor is used to initialize the instance attributes and components of the class. The instance constructor of the superclass is executed before the instance constructor of the subclass, as the subclass inherits the instance components of the superclass. The instance constructor of the subclass must call the instance constructor of the superclass explicitly using super->constructor, unless the superclass is the root node object12

* Instance constructor of subl. This is because the instance constructor is an instance method that is executed automatically when an instance of the class is created using the statement CREATE OBJECT.

The instance constructor is used to initialize the instance attributes and components of the class. The instance constructor of the subclass is executed after the instance constructor of the superclass, as the subclass inherits the instance components of the superclass. The instance constructor of the subclass must call the instance constructor of the superclass explicitly using super->constructor, unless the superclass is the root node object12 References: Constructors of Classes – ABAP Keyword Documentation, METHODS – constructor – ABAP Keyword Documentation

C_ABAPD_2309 Dumps - Pass Your Certification Exam: https://www.dumpleader.com/C_ABAPD_2309_exam.html]