DUMPSBOSS.

OMG Certified UML Professional (OCUP 2) - Intermediate Level

OMG OMG-OCUP2-INT200

Version Demo

Total Demo Questions: 15

Total Premium Questions: 150

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Common Structure	4
Topic 2, Simple Classifiers	1
Topic 3, Structured Classifiers	21
Topic 4, Components	18
Topic 5, Deployments	7
Topic 6, Activities	22
Topic 7, Actions	18
Topic 8, Interactions	20
Topic 9, State Machines	24
Topic 10, Use Cases	2
Topic 11, Profiles	13
Total	150

QUESTION NO: 1

What is true of a required interface associated with a port?

A. identifies the services that the object owning the port expects of objects connected via that port

B. represents an interface that must be defined in the same package in which the classifier owning the port is defined

C. identifies the services that the object owning the port can offer to other objects connected via that port

D. represents an interface that must be defined within the classifier that owns the port

ANSWER: A

Explanation:

The statement “identifies the services that the object owning the port expects of objects connected via that port” is correct. In UML composite structures, a port is an interaction point through which an encapsulated classifier communicates with its environment or with other parts. Interfaces associated with a port describe the contractual features involved in those interactions. A required interface specifies capabilities that the port’s owning classifier needs to obtain from an external collaborating object. In practical terms, the owning object depends on those services and expects a connector through that port to lead to an object that provides them.

This distinction supports substitutability and clear separation of responsibilities: the internal behavior of the owning classifier can rely on the required contract without needing to know the identity or implementation details of the connected provider. UML commonly depicts a required interface using a socket notation at the port, emphasizing that the port needs a compatible service provider. The OMG UML specification defines required interfaces as interfaces specifying services required from the environment, in contrast to provided interfaces, which specify services offered to the environment. See the OMG UML specification’s composite-structure and port semantics in the OMG UML 2.5.1 Specification and the current UML specification landing page at OMG UML.

QUESTION NO: 2

What statements are true about an AcceptEventAction? (Choose two.)

A. It waits for an event occurrence that matches one of its triggers.

B. It may provide values on result output pins.

C. It must invoke an Operation before accepting an event.

D. It is a kind of object node.

E. It can accept only SignalEvents.

ANSWER: A B

Explanation:

An AcceptEventAction waits for the occurrence of an event that matches one of its triggers. It does not invoke a behavior or operation; instead, it accepts an event occurrence from the executing object and enables subsequent activity execution when a matching occurrence is received. Its triggers may refer to events such as signals, calls, time events, or change events.

An AcceptEventAction may provide result OutputPins. These pins make values carried by the accepted event occurrence available to the rest of the activity. For example, where the accepted event is a SignalEvent, values of the received signal can be supplied on result pins. The UML specification defines AcceptEventAction and its result pins in the Actions package; see the OMG UML 2.5.1 Specification.

QUESTION NO: 3

In the exhibit, what are the valid traces for Cont1?

A. either two p messages or two q messages

B. either p followed by q or q followed by p

C. only a p message followed by a q message

D. any combination of two p messages and two q messages

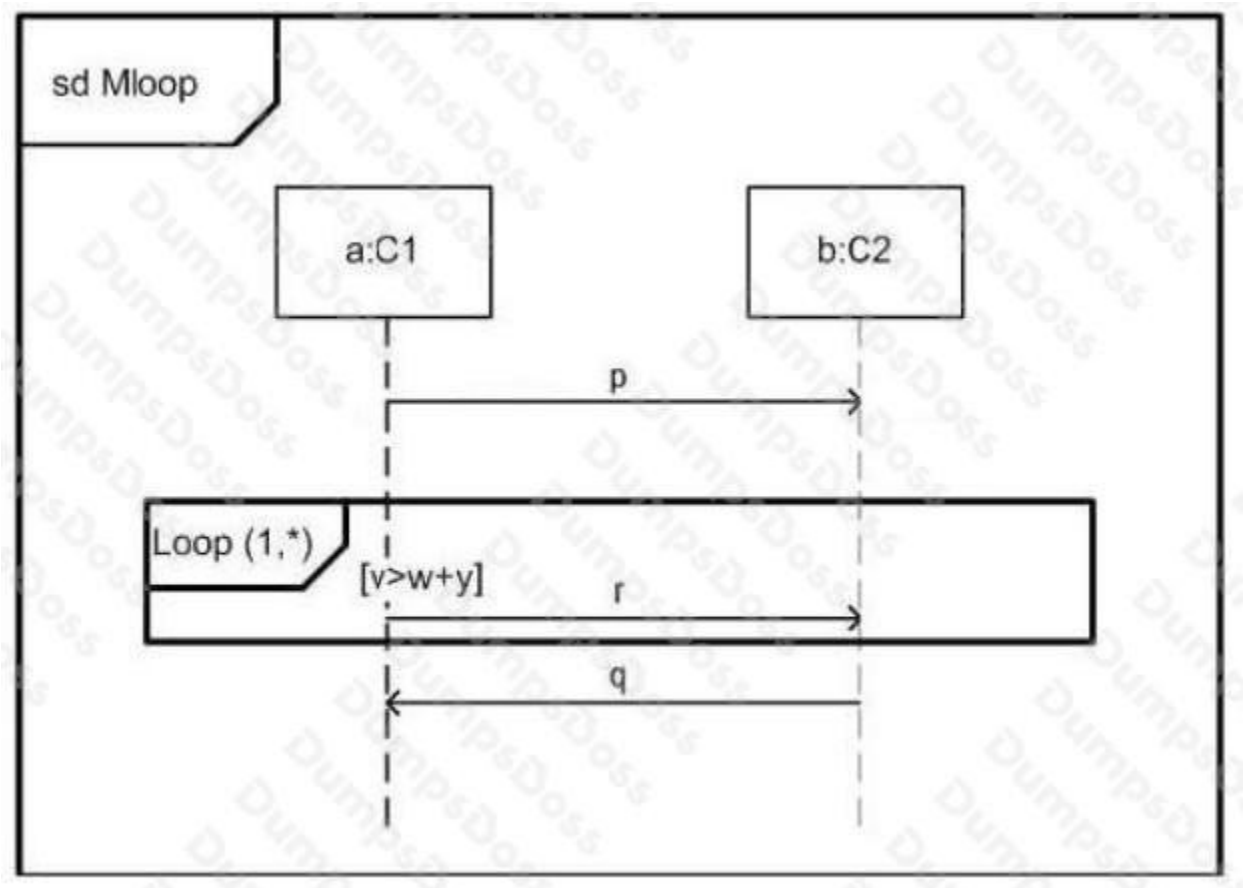
ANSWER: B

Explanation:

The valid traces are *either p followed by q or q followed by p*. Cont1 represents concurrent behavior in which the occurrences associated with *p* and *q* are not constrained to one fixed relative order. UML defines the semantics of a parallel combined fragment in terms of an interleaving (shuffle) of the traces contributed by its operands, while preserving the ordering that exists within each individual operand. Where each concurrent operand contributes one message occurrence, the resulting trace may therefore place the *p* message before the *q* message, or place the *q* message before the *p* message. Both sequences satisfy the concurrent ordering semantics represented by Cont1. This reflects an important UML interaction-modeling principle: concurrency permits alternative observable orderings unless an ordering constraint, such as a strict sequencing construct or an explicit general ordering, has been modeled. The trace set for Cont1 consequently contains exactly the two possible orderings of these two message events. The OMG UML specification defines combined-fragment operators and their trace semantics, including parallel composition, in its interaction-modeling clauses. See the OMG UML 2.5.1 specification and the OMG UML specification page.

QUESTION NO: 4

In the exhibit, what applies if the interaction constraint is false when the loop is entered?



A. All valid traces will include message r.

B. Valid traces include both the case where the loop iterates once, and when it iterates zero times.

C. The specification if illegal.

D. The loop body is not included in a valid trace.

ANSWER: A

Explanation:

All valid traces will include message r. The loop shown in the interaction has a minimum iteration count of one. UML loop combined fragments can specify both lower and upper iteration bounds, in addition to a Boolean guard. The lower bound is significant: it requires the loop operand to execute at least that many times. Consequently, when the interaction constraint is false on entry, that false result does not prevent the required first iteration. The guard is relevant to deciding whether further iterations are permitted after the minimum number of iterations has been satisfied. Since message r is part of the loop operand that must execute during the mandatory iteration, every valid trace contains r. This behavior lets a model express "execute at least once, then continue while the condition holds," rather than merely a conventional zero-or-more loop. UML defines the `loop` combined-fragment operator as repetition of its operand, with an interaction constraint that may include minimum and maximum iteration counts and a Boolean expression. See the OMG UML specification's definition of combined fragments and loop semantics: OMG UML 2.5.1 Specification. The current UML specification is also available from the OMG UML standards page: OMG Unified Modeling Language.

QUESTION NO: 5

What causes a ChangeEvent to occur?

A. Its Boolean change expression changes from false to true.

B. A transition guard evaluates to false.

C. An active state completes its doActivity.

D. A signal is received by the context object.

ANSWER: A

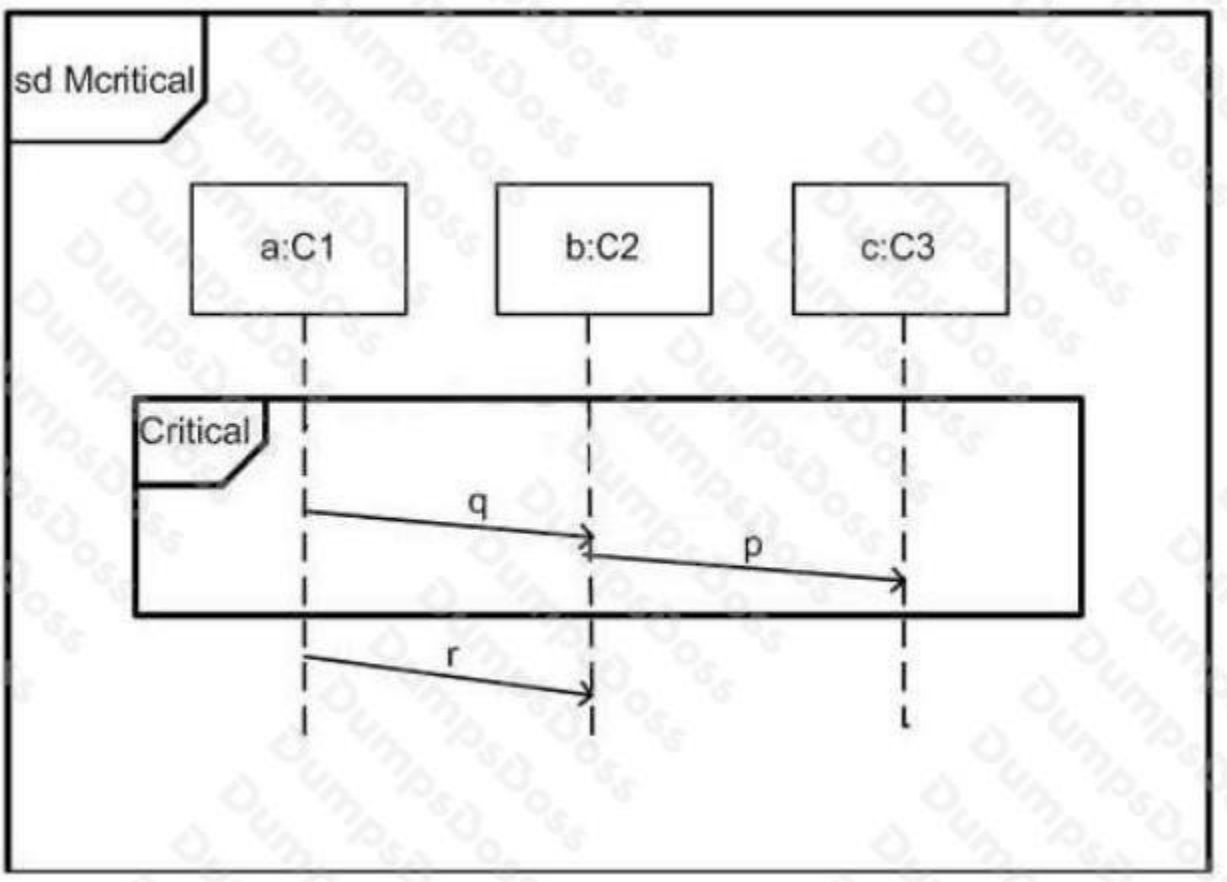
Explanation:

A ChangeEvent occurs when its change expression changes from false to true. The expression is a Boolean-valued ValueSpecification that is evaluated to determine whether the change event has occurred. The event is not triggered merely because the expression is evaluated, nor does it occur simply because the expression remains true.

Change events are useful for state-machine transitions whose triggering condition depends on a change in a modeled condition, such as a balance becoming negative or a resource becoming available. A guard on a transition is evaluated only when the transition is otherwise triggered, whereas a ChangeEvent itself supplies the triggering event when its Boolean condition becomes true. UML defines ChangeEvent and its `changeExpression` in the state-machine package of the OMG UML 2.5.1 Specification.

QUESTION NO: 6

In the exhibit, what is true for Mcritical?



A. r can be sent whenever p has been sent.

B. Whenever q and p have been sent, r can be sent.

C. There are legal traces according to Mcritical where r is absent.

D. The reception of p must precede sending of r.

ANSWER: D

Explanation:

The reception of p must precede sending of r. In the interaction named Mcritical, the relevant message-occurrence specifications are ordered so that the receive event for *p* occurs before the send event for *r*. Consequently, every valid trace of that interaction must preserve this precedence relationship. This is the essential meaning of an ordering constraint in UML interaction semantics: a trace is only valid when it respects the partial order induced by the occurrence specifications shown on the lifelines and by the enclosing interaction fragment.

The use of a *critical* combined fragment also ensures that the occurrences within its covered region are handled as a non-interleavable segment with respect to the relevant covered lifelines. It does not remove the ordering already established between the receipt of *p* and the sending of *r*. Therefore, any execution represented by Mcritical must first reach the reception of *p* before it can perform the send occurrence for *r*. UML defines message occurrence specifications and combined-fragment trace semantics in its interaction model; see the OMG UML 2.5.1 specification and the UML 2.5.1 specification overview.

QUESTION NO: 7

What does a structured node contain? (Choose two.)

A. messages

B. edges

C. classes

D. nodes

E. lifelines

F. States

ANSWER: B D

Explanation:

A structured node, formally represented in UML activity modeling by *StructuredActivityNode*, is a composite activity node that groups a set of contained activity-model elements into one structured region. Its containment properties include *node*, which holds the activity nodes owned by the structured node, and *edge*, which holds the activity edges owned by it. Thus, the structured region can include executable substeps such as actions or control nodes, together with the control-flow and object-flow connections that relate those substeps.

This ownership is important because the grouped nodes and edges form the internal behavior of the structured activity node. UML can therefore treat the region as a single activity node in an enclosing activity while retaining its internal flow semantics. The UML specification defines StructuredActivityNode as an ActivityNode with owned nodes and edges, and specifies that the contained edges connect elements within the structured node in accordance with the applicable well-formedness rules. See the OMG UML specification's activity package definitions at OMG UML 2.5.1 Specification (PDF) and the OMG UML specification landing page at OMG UML Specifications.

QUESTION NO: 8

Instances of the Communication Path class connect instances of which classes in a deployment diagram?

A. Instance Specification and Deployment Specification

B. Deployment Target and Deployed Artifact

C. Node and Property

D. Node and Signal

E. two Nodes

ANSWER: E

Explanation:

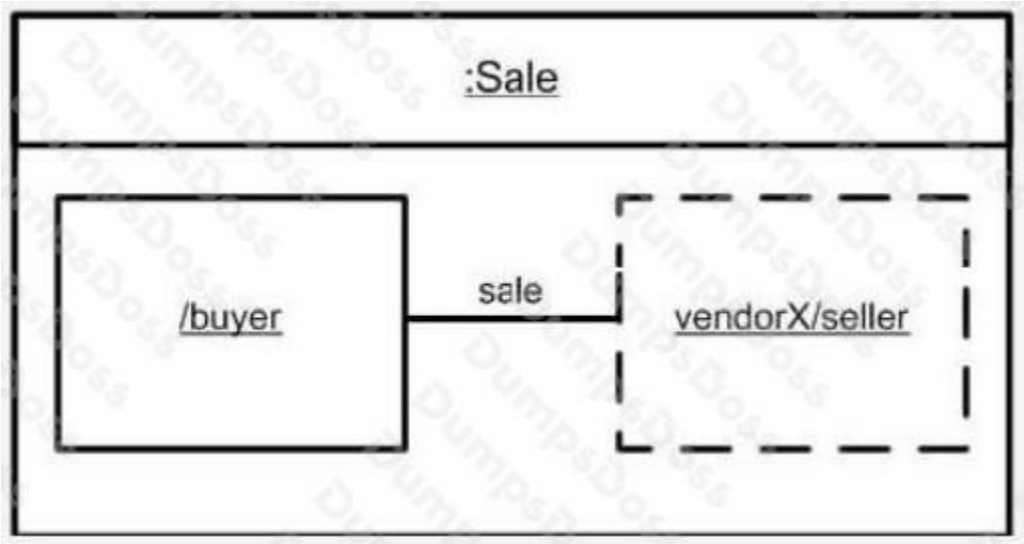
two Nodes is correct. In UML deployment modeling, a CommunicationPath is a specialized Association that represents a communication link between deployment targets. Its association ends are typed by Node, so each instance of a CommunicationPath connects instances of Node—commonly device nodes, execution-environment nodes, or combinations of these. For example, a physical server node may be connected to a router device node, or one execution environment may communicate with another through a modeled network path.

This relationship expresses the ability of the connected nodes to exchange information, rather than the deployment of an artifact onto a target. CommunicationPath may also carry properties that characterize the connection, such as protocol, bandwidth, latency, or other network characteristics. At the model level, the key point is that Node is the metaclass at both ends of the CommunicationPath association. This is why deployment diagrams use communication paths to show the topology and communication capability among the hardware and execution resources that host deployed software.

The UML specification defines CommunicationPath as an association between Nodes in its deployment package; see the OMG UML 2.5.1 specification, section 19.3: OMG UML 2.5.1 Specification (PDF). The normative machine-readable model is also available from OMG UML XMI.

QUESTION NO: 9

What is wrong with the Sale instance diagram shown in the exhibit?



A. Buyer instance name is missing.

B. Link name should be underlined.

C. Sale instance name is missing.

D. Link should be shown with a dashed line.

E. Types of the buyer and seller parts are missing.

ANSWER: B

Explanation:

“Link name should be underlined.” is correct because a link in an instance diagram is itself an *InstanceSpecification*: it represents an instance of an association connecting the participating object instances. UML permits such a link to be displayed as a solid line between the object symbols. When the link’s own name is displayed adjacent to that line, the name uses the standard instance notation and must be underlined. Underlining distinguishes the name of an individual runtime instance from a model-level classifier or association name, which is not underlined. Thus, if the exhibit labels the connecting link with a name but renders that label without underlining, its notation does not conform to the UML instance-specification presentation rules. This is a presentation issue for the named link itself; it does not depend on whether the connected objects have optional instance names or whether their classifiers are displayed. The UML specification defines an InstanceSpecification as a model element representing an instance in a modeled system and specifies the notation for links and their displayed names. See the OMG UML 2.5.1 specification, especially the InstanceSpecification notation in OMG UML 2.5.1, and the OMG UML specification landing page at OMG UML specifications.

QUESTION NO: 10

What statements are true about an ActivityPartition? (Choose two.)

A. It can group activity nodes and edges.

B. It can represent responsibility for activity nodes.

C. It is an executable activity node.

D. It receives tokens from incoming control flows.

E. It is the source or target of an object flow.

ANSWER: A B

Explanation:

An ActivityPartition groups activity nodes and edges according to a chosen criterion. It is commonly used to show responsibility, such as the organizational role, system component, or actor responsible for performing particular actions. Although often drawn as a swimlane, a partition is a grouping construct rather than an executable activity node.

Activity partitions do not receive or offer tokens, and control flows and object flows do not connect to a partition itself. Such edges connect the activity nodes that happen to be contained in different partitions. See the Activities package in the OMG UML 2.5.1 Specification.

QUESTION NO: 11

What are the kinds of structured nodes? (Choose two.)

A. partition

B. object node

C. loop

D. action

E. conditional

ANSWER: C E

Explanation:

A loop is a structured activity node used to represent repeated execution of a body part according to its setup, test, and body semantics. The node encapsulates the internal activity elements involved in iteration and provides explicit control over how values are supplied, tested, and passed through successive iterations. This makes it the UML activity-modeling construct for expressing iteration as a coordinated, self-contained portion of an activity.

A conditional is also a structured activity node. It contains one or more clauses, each of which can include a test and a body. The conditional node evaluates clause tests and executes the applicable clause body or bodies according to its defined behavior. Like a loop, it groups contained executable elements and defines control-flow semantics for that group, rather than merely serving as an individual action, an object-flow element, or a diagram partitioning mechanism. The UML specification identifies ConditionalNode and LoopNode among the concrete forms of StructuredActivityNode. See the OMG UML specification's activity section at OMG UML 2.5.1 specification and the UML specification landing page at OMG UML specification.

QUESTION NO: 12

What features may an Interface own? (Choose two.)

A. operations

B. receptions

C. parts

D. attributes

E. ports

ANSWER: A B

Explanation:

An Interface may own Operations and Receptions. Operations specify services that conforming classifiers provide, while Receptions specify the Signals that conforming classifiers are prepared to receive. Together, these features allow an Interface to describe both synchronous operation-based services and asynchronous signal reception capabilities.

An Interface is not a Class and does not own attributes or parts in the manner of a Class or StructuredClassifier. It may participate in relationships and be realized by classifiers, but those facts do not make associations or ports owned interface features. See the Interfaces package in the OMG UML 2.5.1 Specification.

QUESTION NO: 13

What statements are true about an Include relationship between use cases? (Choose two.)

A. It is directed from the including use case to the included use case.

B. It can factor common behavior into a separate use case.

C. It represents a conditional addition to the base use case.

D. It is directed from the included use case to the including use case.

E. It requires the including use case to specialize the included use case.

ANSWER: A B

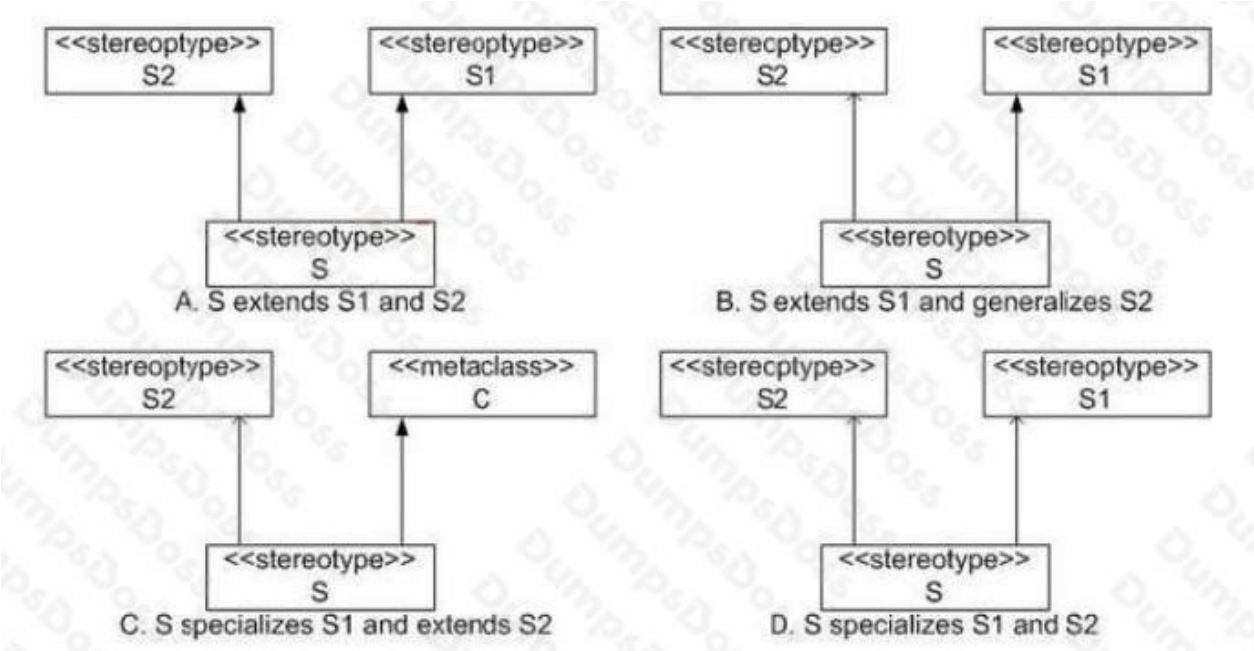
Explanation:

An Include relationship is directed from the including use case to the addition, which is the included use case. The including use case incorporates the behavior of the addition at a location defined in its own behavior. The direction is therefore not from the reused use case back to each use case that uses it.

Include is commonly used to factor out behavior that is shared by more than one use case. The included use case represents reusable behavior that is inserted whenever the including use case reaches the relevant inclusion point. This differs from an Extend relationship, which adds behavior conditionally to an extended use case. UML defines Include as a DirectedRelationship in the Use Cases package. See the OMG UML 2.5.1 Specification.

QUESTION NO: 14

Which is a correct representation of multiple extension and generalization/specialization? (Choose two.)



A. A

B. B

C. C

D. D

ANSWER: C D

Explanation:

The two selected representations correctly apply UML's directed relationship notation while allowing more than one relationship of the relevant kind. In a use-case model, an `<<extend>>` relationship is directed from the extending use case to the extended use case. This expresses optional or conditional behavior that augments a defined extension point or the behavior of the extended use case; several extending use cases may therefore target the same extended use case. Generalization is also directed: the arrowhead is placed at the more general classifier or use case, while each specialized element is at the tail end. A specialized use case inherits the behavior and relationships of its general use case and can add or refine behavior. UML permits multiple specialized elements to generalize the same general element, so diagrams can legitimately combine several extensions with a generalization/specialization structure when each connector's stereotype, direction, and endpoints preserve those semantics. This is consistent with the normative UML use-case relationship definitions in the OMG UML specification, especially the definitions of Extend and Generalization. See the OMG UML 2.5.1 specification and the OMG's UML specification page.

QUESTION NO: 15

What is the target input pin of a CallOperationAction used for?

A. the object on which the operation is invoked

B. the classifier that owns the operation

C. the value returned by the operation

D. the operation's implementation behavior

E. the signal that triggers the operation

ANSWER: A

Explanation:

The target input pin supplies the object on which the operation is invoked. A CallOperationAction invokes an Operation rather than a Behavior directly, so it requires a target object that is an instance of, or conforms to, the Operation's owning classifier. The remaining input pins correspond to the operation's input parameters, and output pins correspond to its output and return parameters.

The UML specification defines CallOperationAction as an InvocationAction with a target InputPin that provides the target object for the call. See the OMG UML 2.5.1 Specification.