PassQueen
a.k.a dumpsqueen.com

OMG-Certified UML Professional Intermediate Exam

OMG OMG-OCUP-200

Version Demo

Total Demo Questions: 15

Total Premium Questions: 158

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, UML 2.5 Basic Concepts	2
Topic 2, UML 2.5 Structure	49
Topic 3, UML 2.5 Behavior	93
Topic 4, UML 2.5 Cross-Cutting Concepts	14
Total	158

QUESTION NO: 1

What is NOT true about a roles and role bindings?

A. A role binding is an association.

B. The same connectable element may be bound to multiple roles in a single collaboration occurrence.

C. A role binding maps a connectable element to a role in a collaboration occurrence.

D. The same object may play roles in multiple collaborations.

E. A role typed by an interface specifies a set of features required by a participant in a collaboration.

ANSWER: A

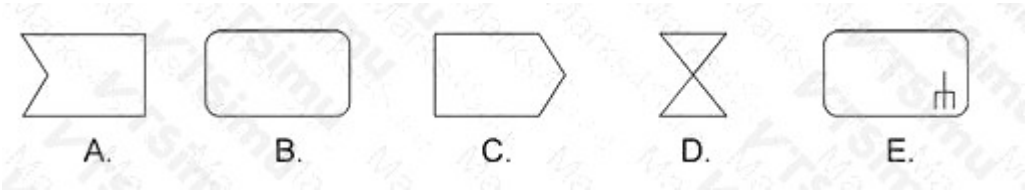
Explanation:

“A role binding is an association.” is the statement that is not true. In UML, a role binding is modeled as a *Dependency*, not as an Association. It is used within a CollaborationUse, which represents a particular occurrence or application of a Collaboration. The binding connects a participating ConnectableElement in the surrounding context with a role, represented by a ConnectableElement, defined by the Collaboration. This mapping identifies which actual participant fulfills each collaboration role for that particular use.

This distinction is important because an Association describes a structural relationship between classifiers and may own properties representing its ends. A role binding has a different purpose: it is a directed semantic mapping that relates an actual participating element to the role it realizes in the collaboration occurrence. UML therefore defines the RoleBinding metaclass as a specialization of Dependency and gives it the semantics of binding a ConnectableElement to a role in a CollaborationUse. The normative UML specification’s CollaborationUse and Dependency definitions provide this basis; see the OMG UML 2.5.1 specification and the UML 2.5.1 overview.

QUESTION NO: 2 - (SIMULATION)

What symbol depicts a SendSignalAction?



ANSWER: See the explanation for the answer

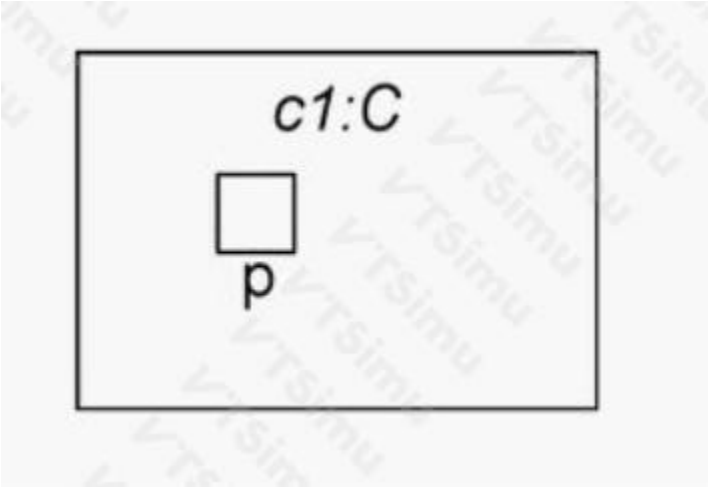
Explanation:

Answer: C.

A UML `SendSignalAction` is depicted as a convex pentagon-like action symbol with a pointed right side, as shown in option C. It represents dispatching a signal asynchronously.

QUESTION NO: 3

What does the composite structure notation in the exhibit show?



A. p is a port on a part of C, which is not shown.

B. p is a hidden port of C.

C. p is a port providing a system service.

D. p is a behavior port of C.

ANSWER: B

Explanation:

The notation shows that *p is a hidden port of C.* In a UML composite-structure diagram, a Port is a property of an encapsulated classifier, such as classifier *C*, and it represents an interaction point through which the classifier can communicate with its environment or with its internal parts. A normally exposed port is drawn as a small square placed on the boundary of its owning classifier. The exhibit instead places the small port symbol *p* within the boundary of *C*, rather than on that boundary. UML defines this presentation as a hidden Port.

A hidden port remains owned by the classifier and can participate in the classifier's internal structure and connector configuration, but it is not presented as an externally visible interaction point. The visual distinction is important: boundary placement expresses visibility at the classifier's external interface, whereas placement inside the classifier indicates that the port is hidden. This is a matter of UML notation and does not by itself state that the port provides a particular service or that it dispatches received requests directly to the classifier's behavior. The normative UML composite-structure notation and Port semantics are specified in the OMG UML specification, particularly the Composite Structures package: OMG UML 2.5.1 Specification (PDF). The OMG UML specification landing page is available at OMG UML.

QUESTION NO: 4

What behavior may be specified for an object node? (Choose two)

A. selection behavior

B. ordering behavior

C. entry behavior

D. exit behavior

E. transition effect

ANSWER: A B

Explanation:

An ObjectNode may specify a selection behavior and an ordering behavior. The selection behavior determines how tokens are selected for transfer when more than one token is available. The ordering behavior determines the order in which tokens are offered by the object node, such as first-in-first-out or another defined ordering.

These behaviors support precise modeling of object-token handling in an activity. They do not make the object node an action or a control node. Object nodes hold tokens and participate in object flows, whereas actions execute behavior and control nodes coordinate control flow. See the ObjectNode definition in the OMG UML 2.5.1 specification.

QUESTION NO: 5

How many arrows can point from a flow final node?

A. any number

B. none

C. two

D. one

ANSWER: B

Explanation:

The correct answer is *none*. In a UML activity diagram, a FlowFinalNode terminates the particular flow that reaches it. It is a kind of FinalNode used to end one path of execution without necessarily ending all other concurrent flows in the containing activity. Because the arriving token is consumed and that flow stops at the node, a FlowFinalNode cannot have outgoing activity edges. Thus, no arrows may leave it.

This is distinct from the practical effect of merely pausing or redirecting a flow: the node is explicitly a termination point for that token path. An incoming control-flow or object-flow edge can deliver a token to the FlowFinalNode, but the node provides no continuation edge on which that token could be offered. This rule makes the diagram's termination semantics unambiguous, particularly when other flows in the same activity continue independently.

The UML specification defines FlowFinalNode within the activity-modeling semantics and specifies the structural constraint that its outgoing edges are empty. See the OMG UML 2.5.1 specification, especially the activity-node definitions, and the OMG's UML specification overview.

QUESTION NO: 6

What may an OpaqueExpression specify? (Choose two)

A. one or more expression bodies

B. the languages of the expression bodies

C. one or more owned Operations

D. the Regions of a state machine

E. the ends of an Association

ANSWER: A B

Explanation:

An OpaqueExpression is a ValueSpecification whose meaning is given in one or more languages. Its `body` property contains textual expressions, and its `language` property identifies the languages in which the corresponding bodies are written. For example, an OpaqueExpression might contain an OCL body used as a constraint or guard.

The body and language entries correspond by position when more than one is provided. An OpaqueExpression does not itself define a UML operation or a state-machine region. The UML specification defines these properties in its Common Structure concepts. See the OMG UML 2.5.1 specification.

QUESTION NO: 7

Which statements are true about signal-based communication? (Choose two)

A. A Signal is a kind of Classifier.

B. A Reception specifies that a classifier can receive a Signal.

C. A Signal must define an operation that returns a value.

D. Sending a Signal is a synchronous operation call.

E. A Reception is a kind of Association.

ANSWER: A B

Explanation:

A Signal is a kind of Classifier that specifies a named asynchronous communication. It may own attributes that define the data carried by occurrences of the Signal. A Reception is a behavioral feature that specifies that instances of a classifier are prepared to receive a particular Signal.

A signal occurrence is not an operation call and does not require the sender to wait for a return value. A `SendSignalAction` may be used in an activity to transmit a Signal to a target object. See the UML definitions of Signal and Reception in the OMG UML 2.5.1 specification.

QUESTION NO: 8

What does a structured node contain? (Choose two)

A. states

B. lifelines

C. classes

D. messages

E. nodes

F. edges

ANSWER: E F

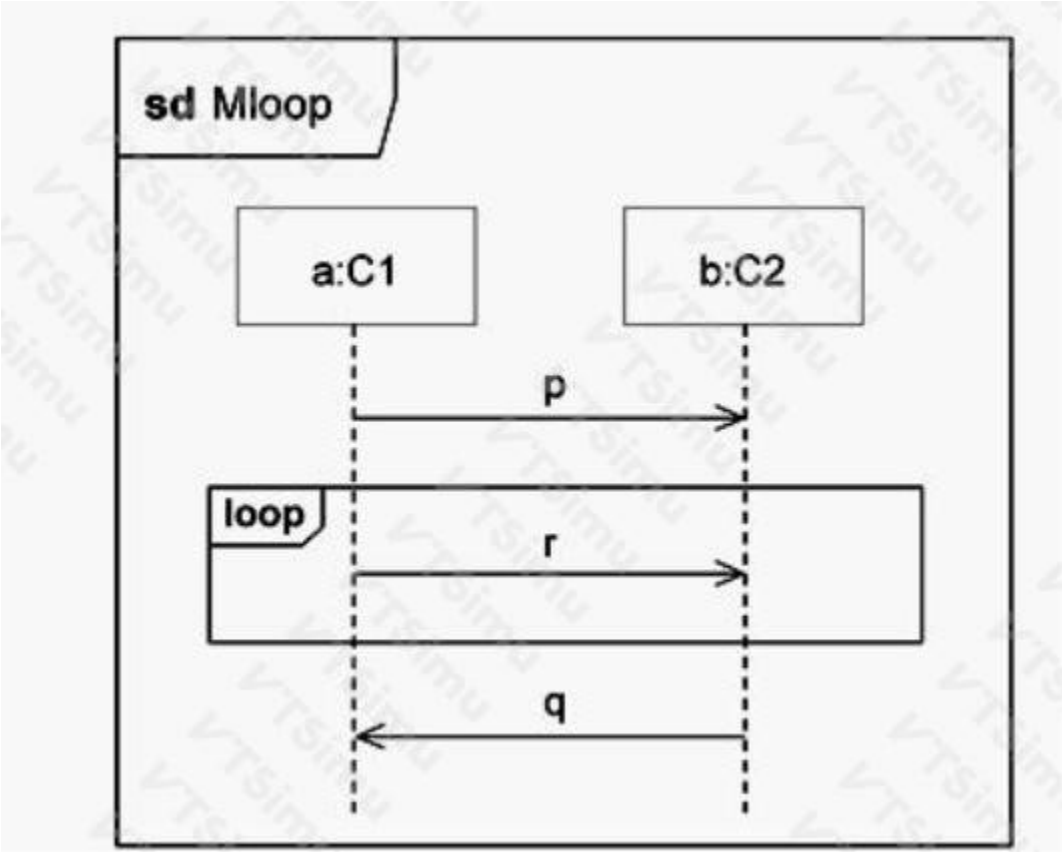
Explanation:

A structured node, formally a *StructuredActivityNode* in UML activity modeling, encapsulates a portion of an activity as a distinct executable region. Its owned contents are activity **nodes** and activity **edges**. Nodes represent the executable or control/data-flow elements within the region, such as actions, control nodes, and object nodes. Edges define the flows that connect those nodes, including control flows and object flows. This ownership gives the structured node a clear internal activity graph while allowing the region itself to behave as a single node in the enclosing activity.

The UML metamodel defines *StructuredActivityNode* with properties for contained *node* elements and contained *edge* elements. The node and edge collections express the internal structure of the region and support constructs such as conditional nodes, loop nodes, expansion regions, and sequence nodes. Therefore, *nodes* and *edges* are the two required answers. See the OMG UML specification's activity-modeling definitions in the OMG UML 2.5.1 specification and the corresponding UML 2.5.1 specification landing page.

QUESTION NO: 9

Assume !p denotes sending of p, ?p the reception of p. In the exhibit, what traces are valid? (Choose three)



A.

B.

C.

D.

E.

F.

ANSWER: A B C

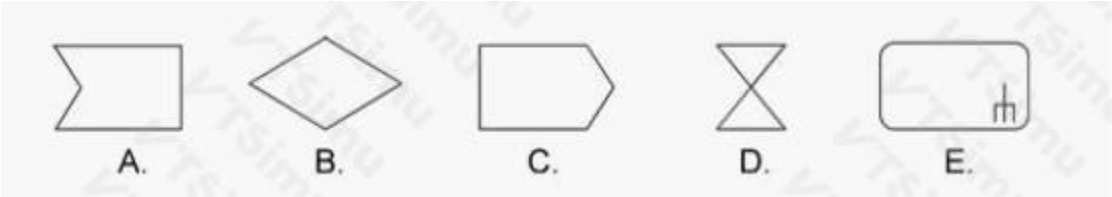
Explanation:

The valid traces are <!p, !r, ?p, ?r, !r, ?r, !r, ?r, !q, ?q>, <!p, ?p, !q, ?q>, and <!p, ?p, !r, !r, ?r, ?r, !q, ?q>. The interaction first establishes the required send/receive ordering for p and later for q: a send occurrence precedes its corresponding receive occurrence. The repeated r behavior is optional, so a trace may contain no r occurrences at all, as in <!p, ?p, !q, ?q>. Where r occurs, the repetition permits multiple instances. Because r is asynchronous, its send occurrence does not require the sender to wait for the corresponding reception before another r can be sent. Consequently, both an interleaving of r with completion of p and a pair of consecutive r sends followed by their receptions are permitted. These traces preserve each message's send-before-receive causality while respecting the ordering constraints imposed by the interaction. UML defines message occurrence specifications and the ordering semantics used to determine valid interaction traces; see the OMG UML 2.5.1 specification and the OMG UML specification overview.

QUESTION NO: 10

CORRECT TEXT

What symbol depicts a CallBehaviorAction?



A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

ANSWER: E

Explanation:

The symbol depicted by the selected image represents a CallBehaviorAction. In a UML activity diagram, a CallBehaviorAction is an action that invokes a Behavior, such as an Activity, Interaction, State Machine, or OpaqueBehavior. Its notation uses the normal action shape—a rounded rectangle—with the distinctive rake symbol in its lower-right corner. The rake indicates that the action is calling a separately defined behavior rather than merely representing an atomic or unspecified action. This notation lets a model reader recognize that the action has a decomposed or reusable behavioral definition that may be shown elsewhere in the model.

CallBehaviorAction is defined in the UML Actions package and includes a *behavior* property identifying the Behavior being invoked. The graphical rake is therefore significant: it communicates the relationship between the action node in the current activity and the invoked Behavior. The OMG UML specification describes the notation for CallBehaviorAction and its rake symbol in the Actions clause. See the official OMG UML 2.5.1 specification and the UML 2.5.1 specification overview.

QUESTION NO: 11

What is NOT a purpose of a port owned by a classifier?

A. provides a distinct point of interaction between the classifier and its environment

B. serves as an end point for connectors

C. hides the internals of that classifier from other classifiers

D. specifies an association to the classifier

ANSWER: D

Explanation:

The statement *specifies an association to the classifier* is not a purpose of a port. In UML, a Port is a Property of an EncapsulatedClassifier that identifies a distinct interaction point between that classifier and its environment. Its role is to make the classifier's externally visible interaction points explicit, including the interfaces that may be provided or required at

that point. A Port can also act as a connectable element, allowing Connectors to attach to it and route communications either to the classifier's internal parts or to its behavior.

An Association, by contrast, is a separate UML relationship concept: it describes a set of links between instances and is represented by Properties at its ends. Although a Port is itself a kind of Property and can have a type, multiplicity, and visibility, declaring a Port does not define an Association to its owning classifier. The ownership relationship is instead part of the classifier's structural ownership of the Port. Therefore, describing a port as specifying an association to the classifier confuses a Port's interaction-point role with UML's separate Association mechanism. See the OMG UML specification's definition of Port and its discussion of EncapsulatedClassifiers in OMG UML 2.5.1, and the official specification landing page at OMG UML 2.5.1 Specification.

QUESTION NO: 12

What does a run-to-completion processing for state machines mean?

A. The thread executing the state machine cannot be pre-empted by the scheduler.

B. The executions of orthogonal regions are serialized.

C. Interrupts are disabled while the state machine is running.

D. No other event will be processed until the current event is fully processed.

ANSWER: D

Explanation:

Run-to-completion processing means that, once a state machine starts dispatching an event, it completes the entire processing associated with that event before it dispatches the next event. This processing can include selecting and executing a transition, performing its exit behavior, transition effect, and entry behavior, as well as any resulting completion processing required to reach a stable state configuration. Events that arrive while this work is in progress may be placed in the event pool, but they are not dispatched to the state machine until the current run-to-completion step has finished. The concept is a semantic rule for how events are handled by the state machine; it does not prescribe operating-system scheduling, prohibit interrupts, or require a particular physical threading implementation. UML describes an event pool as holding events awaiting dispatch and specifies dispatching behavior in the state-machine execution model. See the OMG's UML 2.5.1 specification, especially the state-machine semantics and event dispatch material. A related OMG reference is the UML specification landing page, which provides the current formal specifications and associated machine-readable artifacts.

QUESTION NO: 13

What are valid ways to present a component's artifacts file, such as a jar file? (Choose two)

A. stereotype of class with keyword <> with a solid arrow pointing to the related component

B. an Artifact stereotyped <> with a dashed <> dependency pointing to the related component

C. stereotype of a component with keyword <> with a solid arrow pointing to the related component

D. stereotype of a component with keyword <> with a dashed arrow pointing to the related component

E. as a compartment of a related component

ANSWER: B E

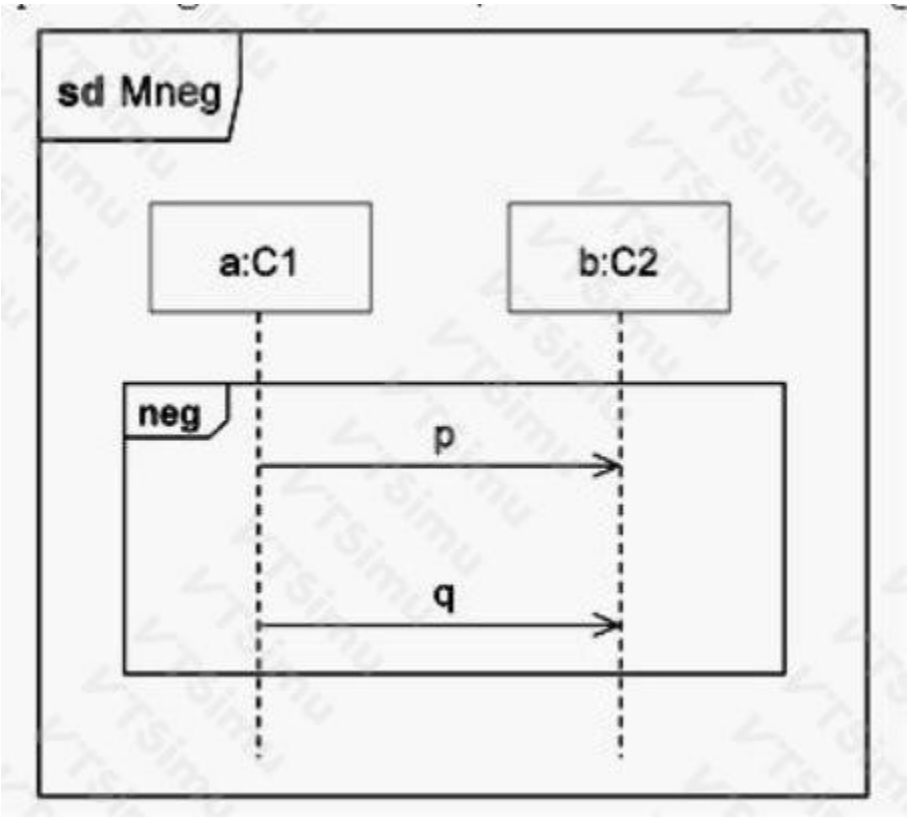
Explanation:

An archive such as a JAR can be modeled as an Artifact, optionally using an «archive» stereotype to communicate its file/archive nature. The Artifact is connected to the component it implements using a dashed «manifest» dependency directed from the Artifact to the component. A Manifestation indicates that the physical Artifact embodies or implements the model element, making this notation appropriate for showing a deployable or deliverable file associated with a component.

UML also permits artifacts associated with a component to be listed directly in a compartment of the component's notation. This is a compact presentation when the diagram's purpose is to show the component and the files that realize it, without needing to draw each Artifact and Manifestation relationship separately. In both presentations, the component remains the logical modular unit, while the archive/JAR is a physical Artifact that manifests it. The UML specification defines Artifact and Manifestation as part of its deployment-related modeling concepts and allows classifiers to be displayed with appropriate compartments. See the OMG UML 2.5.1 specification and the OMG UML specification overview.

QUESTION NO: 14

Assume that !p means sending message p and ?p receiving it. In the exhibit, what is true about Mneg?



A. Neither p nor q should be sent between a and b.

B. is an invalid trace according to Mneg.

C. p and q should not be sent concurrently from a to b.

D. is an invalid trace according to Mneg.

ANSWER: D

Explanation:

is an invalid trace according to Mneg. In a UML interaction, a `neg` combined fragment specifies an invalid, or forbidden, interaction trace. The events shown within the operand define the ordering pattern that must not occur in a valid execution of the enclosing interaction. Here, the relevant sequence has the sending of `p`, followed by the sending of `q`, and then the corresponding receives in the order `?p` followed by `?q`. Consequently, the trace `<!p, !q, ?p, ?q>` matches the prohibited behavior denoted by Mneg and is therefore invalid. This does not mean that the individual messages are universally prohibited; rather, it is this interaction occurrence and ordering represented by the negative fragment that is excluded. UML defines `neg` as a combined-fragment operator whose operand describes traces that are invalid, which is precisely the semantic basis for identifying this trace as forbidden. See the OMG UML 2.5.1 specification, particularly its Interaction and CombinedFragment semantics, and the OMG's UML specification page.

QUESTION NO: 15

What does an activity partition contain? (Choose two)

A. lifelines B. states

B. classes

C. edges

D. nodes

E. messages

ANSWER: C D

Explanation:

An activity partition contains **edges** and **nodes**. In UML, an Activity is modeled as a directed graph whose executable and control-modeling elements are ActivityNodes, connected by ActivityEdges. An ActivityPartition is an ActivityGroup used to divide that activity graph according to a responsibility, organizational unit, classifier, or other grouping criterion. This is the semantic basis for the familiar “swimlane” notation in an activity diagram.

The partition does not create a different kind of behavioral element; instead, it owns or references the relevant elements of the activity graph. Consequently, its contents can include ActivityNodes, such as actions, control nodes, object nodes, and pins, as well as ActivityEdges, such as control flows and object flows, when those edges belong to the partition. A partition may also be nested, allowing a modeler to refine responsibility groupings while retaining the underlying nodes and edges that express the activity’s behavior.

This follows the UML specification’s definition of ActivityPartition as an ActivityGroup and its explicit containment relationships for nodes and edges. See the OMG UML 2.5.1 specification and the OMG UML specification overview.