PassQueen
a.k.a dumpsqueen.com

Java EE 7 Application Developer

Oracle 1z0-900

Version Demo

Total Demo Questions: 18

Total Premium Questions: 185

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Java EE Architecture	29
Topic 2, Java Persistence API	14
Topic 3, Enterprise JavaBeans	13
Topic 4, JavaServer Faces	11
Topic 5, Contexts and Dependency Injection for the Java EE Platform	9
Topic 6, Java API for WebSocket	8
Topic 7, JSON Processing	2
Topic 8, Java Message Service API	6
Topic 9, Java EE Web Services	15
Topic 10, Mix Questions	78
Total	185

QUESTION NO: 1

Your client has decided that Discrete Manufacturing will be implemented at a future stage, so any new supply from current manufacturing will be added to current inventory by the Open Transactions Interface. The immediate requirement is to go live with Inventory and Order Management.

Which two seeded transaction types can be omitted for material status control?

A. All Transaction Types Related to Work in Process

B. Average Cost Update

C. Backflush Subinventory Transfer

D. Miscellaneous Issues and Receipts

E. All Internal Transactions

ANSWER: A C

Explanation:

All Transaction Types Related to Work in Process and **Backflush Subinventory Transfer** can be omitted because the implementation does not yet include Discrete Manufacturing or its Work in Process processing. Material status control uses transaction-type controls to determine which inventory transactions are permitted for material in a particular status. The implementation should therefore include transaction types required by the applications that are live, particularly Inventory and Order Management, while excluding seeded transaction types that exclusively support functionality not being deployed.

Work in Process transaction types are intended for manufacturing execution activities, such as moving, completing, returning, issuing, and receiving material in relation to WIP jobs. A backflush subinventory transfer is likewise a manufacturing-oriented transaction: backflushing automatically consumes component material as part of production reporting and can transfer material from the designated supply subinventory. Since manufacturing supply is instead being introduced into inventory through the Open Transactions Interface, neither WIP execution nor backflush processing needs to be authorized by the material-status setup at go-live.

Oracle's Inventory documentation describes material status control as a mechanism for restricting inventory transactions by status and identifies transaction types as part of those controls. See the Oracle Inventory User's Guide: Material Status Control and the Oracle Inventory User's Guide.

QUESTION NO: 2

The Applications Development Framework Desktop Integration (ADFdi) user interface is supported in Oracle Fusion Receiving, Inventory Management, and Shipping products.

Identify four ADFdi user interfaces that are supported.

A. Upload ASN or ASBN

B. Manage Lot/Serial Interface

C. Manage Inventory Transaction Correction in Spreadsheet

D. Review Receipts Interface

E. Review Count Interface Records

F. Manage Shipment Message Interface

ANSWER: B C D F

Explanation:

ADF Desktop Integration (ADFdi) provides spreadsheet-based work areas that connect Microsoft Excel to Oracle Fusion applications. For the Receiving, Inventory Management, and Shipping business flows in the question, the supported interfaces are **Manage Lot/Serial Interface**, **Manage Inventory Transaction Correction in Spreadsheet**, **Review Receipts Interface**, and **Manage Shipment Message Interface**. These work areas support operational processing in which users need to review interface data, correct transaction-related information in a spreadsheet-oriented experience, and manage inbound or outbound integration records. In particular, lot and serial data must be available for controlled inventory transactions; transaction corrections require a spreadsheet work area suited to reviewing and updating correction information; receipt-interface records must be reviewed as part of receiving integration processing; and shipment-message interface records support shipping-message integration processing. ADFdi is intended to combine Excel productivity features with application-controlled validation and service-based interaction, rather than functioning as an independent spreadsheet import utility. Oracle documents the ADF Desktop Integration architecture and its Excel-integrated application model in the Oracle ADF Desktop Integration Developer's Guide. Product documentation and current functional guides are available through the Oracle Fusion Cloud SCM documentation library.

QUESTION NO: 3

Which JSON-P class should you use to read an entire JSON object from an input stream by using the object model API?

A. `JsonReader`

B. `JsonGenerator`

C. `JsonObjectBuilder`

D. `JsonWriter`

ANSWER: A

Explanation:

Use `JsonReader`. The JSON-P object model API reads a JSON document into an in-memory object model, such as `JsonObject` or `JsonArray`. A reader is created through `Json.createReader(...)`, and `readObject()` returns the JSON object represented by the input.

This differs from `JsonParser`, which belongs to the streaming API and exposes a forward-only sequence of parsing events. The object model is appropriate when application code needs convenient random access to the fully read JSON structure. See the JsonReader API documentation and the Oracle Java EE 7 Tutorial: JSON Processing.

QUESTION NO: 4

How do you specify a default error page in your web.xml file?

A.
```
<on-error>
<location>/general-error.html<location>
</on-error>
```

B.
```
<error-page>
<error-code>*</error-code>
<location>/general-error.html</location>
</error-page>
```

C.
```
<on-error>
<error-code>*</error-code>
<location>/general-error.html</location>
</on-error>
```

D.
```
<error-page>
<location>/general-error.html</location>
</error-page>
```

A. Option A

B.
java.lang.Throwable
/error.jsp

C. Option C

D. Option D

ANSWER: B

Explanation:

A default application error page is declared with an `<error-page>` element containing an `<exception-type>` of `java.lang.Throwable` and a `<location>` for the resource that renders the error response. `Throwable` is the common superclass for Java exceptions and errors, so this mapping provides a general fallback for uncaught failures that reach the web container. The location is a context-relative path, such as `/error.jsp`, and the target resource can use the standard servlet error request attributes to present useful error information or a user-friendly message.

The Servlet deployment descriptor supports error-page mappings by exception type and by HTTP status code. A mapping based on `java.lang.Throwable` is appropriate when “default” means the general error page for unexpected exceptions rather than a page dedicated to one status, such as a missing-resource response. The application server uses the deployment descriptor mapping while processing an error and dispatches to the configured location. See Oracle’s Java EE tutorial discussion of deployment descriptors at Servlet deployment descriptors and the Servlet specification error-handling rules at Java Servlet specification.

QUESTION NO: 5

Given the code fragments:

```
//line 1
public class ContactId implements Serializable {
    private String countryCode;
    private String phoneNumber;
    // remaining implementation
}

@Entity
//line 2
public class Contact implements Serializable {
    @NotNull
    @Id
    protected String countryCode;
    @NotNull
    @Id
    protected String phoneNumber;
    // remaining implementation
}
```

Which action completes this composite primary key implementation?

A. Add @IdClass annotation at line 1.

B. Add @Embeddable annotation at line 1 and replace both @Id annotations with @EmbeddedId annotations.

C. Add @IdClass(ContactId.class) annotation at line 2.

D. Add @Embeddable annotation at line 1 and @EmbeddedId(ContactId.class) at line 2.

ANSWER: C

Explanation:

The correct action is to add `@IdClass(ContactId.class)` at line 2, on the entity class. This is the `IdClass` composite-primary-key strategy: the entity retains an `@Id` annotation on each persistent attribute that participates in its primary key, while `ContactId` supplies the separate primary-key class. The class named in `@IdClass` must correspond to the entity's primary-key attributes: it contains fields or properties with the same names and compatible Java types as the entity attributes annotated with `@Id`. It must also meet the normal primary-key-class requirements, including being public, serializable, and defining appropriate `equals` and `hashCode` behavior. Applying the annotation to the entity declaration associates the two individual identifier attributes with `ContactId` as one logical composite key. This is distinct from the embedded-id approach, which uses one entity attribute of an embeddable key type and annotates that single attribute with `@EmbeddedId`. Oracle's Java EE tutorial describes `@IdClass` as the approach in which the entity has multiple `@Id` fields and references the key class through the entity-level annotation. See the Oracle Java EE 7 Tutorial: Primary Keys and the Java EE 7 @IdClass API documentation.

QUESTION NO: 6

Which two tasks must be defined to a set up the Facilities Schedule? (Choose two.)

A. Manage Facility Workday Patterns

B. Manage Facility Operations

C. Manage Facility Holidays

D. Manage Facility Time Cards

E. Manage Facility Shifts

ANSWER: A E

Explanation:

A Facilities Schedule is built from the recurring working-time structure assigned to a facility. **Manage Facility Workday Patterns** is required because it defines the regular days and hours during which the facility operates. This pattern establishes the calendar framework used to determine normal availability across the week. **Manage Facility Shifts** is also required because shifts define the specific operational periods within that working-time framework, such as daytime, evening, or overnight coverage. Together, workday patterns and shifts provide the foundational calendar data needed for a facility schedule to represent when work can be planned and performed. Oracle Warehouse Management setup documentation describes facility configuration and scheduling as part of the operational setup used to support warehouse activities and resource availability. For Oracle product documentation and the current Warehouse Management documentation library, see the Oracle Warehouse Management documentation and the Oracle Cloud Applications documentation portal.

QUESTION NO: 7

Which three replenishment count types are used for Periodic Automated Replenishment (PAR)?

A. Order Economic Quantity (EQQ): Generates a replenishment request for the calculated quantities based on inventory policies to maintain the PAR quantity.

B. Reorder Point: Generates a replenishment request for the order quantity based on days of cover for the item lead time.

C. Order Quantity: Generates a replenishment request for the order quantity entered for the replenishment count.

D. Order PAR: Generates a replenishment requisition for the PAR-level quantity.

E. On-hand Quantity: Generates a replenishment requisition for the difference between the on-hand quantity and the PAR-level.

ANSWER: C D E

Explanation:

Periodic Automatic Replenishment (PAR) supports three count types that determine how Oracle calculates the replenishment quantity for a PAR location and item. **Order Quantity: Generates a replenishment request for the order quantity entered for the replenishment count.** is used when the counter specifies the exact quantity to request; the entered value becomes the requested replenishment quantity. **Order PAR: Generates a replenishment requisition for the PAR-level quantity.** supports requesting the location's defined PAR quantity directly. **On-hand Quantity: Generates a replenishment requisition for the difference between the on-hand quantity and the PAR-level.** is used when the counter records what is currently available, allowing the application to replenish the location back up to its configured PAR level. These count types provide the operational flexibility needed for periodic replenishment: request a fixed amount, request the established target level, or calculate the shortfall from a physical on-hand count. Oracle documents PAR as an inventory replenishment process for maintaining stock at designated locations; see the Oracle Fusion Cloud SCM documentation and Oracle's documentation search for PAR.

QUESTION NO: 8

You defined an Inventory Organization that is meant to track the contract manufacturing process outsourced to a supplier.

Where do you mention that this Inventory Organization is meant for one particular supplier?

A. Manage Inventory Organization Locations

B. Manage Subinventories

C. Manage Inventory Organizations

D. Manage Contract Manufacturing Relationships

E. Manage Item Organizations

ANSWER: D

Explanation:

Manage Contract Manufacturing Relationships is the correct setup task because it establishes the business relationship between an internal inventory organization used for contract manufacturing and the specific external supplier that performs the work. A contract-manufacturing inventory organization represents inventory that is owned, planned, or tracked by the enterprise while manufacturing activities are performed at the supplier's site. The relationship setup is where Oracle Fusion Cloud SCM identifies the supplier and supplier site associated with that contract-manufacturing organization, enabling the application to use the appropriate trading-partner context for outsourced manufacturing transactions and supply-chain processing.

This association is important because the inventory organization by itself defines an inventory-tracking entity, but the contract-manufacturing relationship supplies the supplier-specific connection required for the outsourced process. Oracle's manufacturing implementation documentation describes contract manufacturing as a process involving an enterprise and a supplier, with the relationship configuration linking the relevant organizations and supplier details. See Oracle's Manage Contract Manufacturing Relationships documentation and the Contract Manufacturing overview.

QUESTION NO: 9

A third-party service equipment is used for various purposes in a manufacturing facility and needs to be overhauled once a year. This equipment is charged based on its consumption without a project dependency.

At this point, it should be transferred out of the vendor's warehouse for maintenance and then returned back in the future depending upon the requirement.

The material stock transfer should happen with an offline approval and with an account alias as a logical reference.

What is the ideal way to handle this process?

A. Transfer Order

B. Requisitions

C. Move Request

D. Miscellaneous Transaction

E. Subinventory Transfer

ANSWER: D

Explanation:

Miscellaneous Transaction is the appropriate process because it supports inventory movements that do not arise from a sales order, purchase order, transfer order, requisition, or project. In this scenario, the equipment is issued from inventory for an external maintenance purpose and is later received back when required. A miscellaneous issue records the material leaving the vendor warehouse, while a miscellaneous receipt can record its return.

Account aliases are specifically useful in miscellaneous transactions as logical, user-friendly references for predefined accounting combinations. This lets the organization associate the issue or receipt with the appropriate consumption, maintenance, or clearing account without requiring a project dependency. The transaction can therefore be processed directly by authorized inventory personnel under the organization's offline approval procedure, while preserving the necessary inventory and accounting audit trail.

Oracle Inventory Management describes miscellaneous transactions as a way to perform inventory receipts and issues for business events outside standard order-based flows. Oracle also documents account aliases as named references that simplify selection of account combinations during such transactions. See the Oracle Fusion Cloud Inventory Management documentation and the Account Aliases REST resource documentation.

QUESTION NO: 10

Given the code fragment from a Facelet page:

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:pt="http://xmlns.jcp.org/jsf/passthrough"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:jsf="http://xmlns.jcp.org/jsf"
      xmlns:ui="http://java.sun.com/jsf/facelets">
       <h:form id="searchForm">
            <!-- Line 1 -->
            <h:commandButton value="Search"
                  action="#{searchMB.search}"/>
      </h:form>
</ui:composition>
```

On Line 1, you ae asked to insert a search box that displays the text "Search Here" via a placeholder.

Assume searchMB is a valid Managed Bean.

Which two options enable you to create a search box with a placeholder attribute on Line 1? (Choose two.)

A. < h:inputText value="#(searchMB.query)" > < f:param name="placeholder" value="Search Here"/ > < /h:inputText >

B. < h:inputText value="#(searchMB.query)" placeholder="Search here"/ >

C. < input jsf:id="search" placeholder="Search here" jsf:value="# (searchMB.query)" > < /input >

D. < h:inputText pt:placeholder="Search Here" value="#(searchMB.query)"/ >

E. < input id="search" jsf:placeholder="Search Here" value="$(searchMB.query)" > < /input >

ANSWER: C D

Explanation:

The valid approaches use the JavaServer Faces 2.2 HTML-friendly markup feature or the pass-through-attributes feature. With `<input jsf:id="search" placeholder="Search here" jsf:value="# (searchMB.query)"> </input>`, the ordinary HTML `input` element supplies the browser-level `placeholder` attribute, while the `jsf:id` and `jsf:value` attributes cause Facelets to associate the element with a JSF input component and bind its submitted value to the managed-bean property. In actual Facelets source, the value expression uses the standard `#{searchMB.query}` syntax. This pattern lets developers retain familiar HTML markup while still using JSF component processing and model binding.

`<h:inputText pt:placeholder="Search Here" value="#(searchMB.query)"/>` is also valid because the `pt:` namespace designates a pass-through attribute. JSF renders that attribute unchanged to the generated HTML input element, producing `placeholder="Search Here"`. Pass-through attributes are specifically intended to expose HTML5 attributes that are not native properties of a JSF component. The page must declare the JSF and pass-through namespaces, typically `xmlns:jsf="http://xmlns.jcp.org/jsf"` and `xmlns:pt="http://xmlns.jcp.org/jsf/passthrough"`. See the Java EE 7 Tutorial: HTML5-Friendly Markup and the JSF pass-through attributes documentation.

QUESTION NO: 11

Given the set of navigation rules:

```
<navigation-rule>
   <from-view-id>list-widgets</from-view-id>
         <navigation-case>
            <from-outcome>add</from-outcome>
            <to-view-id>add-widget</to-view-id>
         </navigation-case>
  </navigation-rule>
  <navigation-rule>
         <from-view-id>*</from-view-id>
         <navigation-case>
                  <from-outcome>home</from-outcome>
                  <to-view-id>home</to-view-id>
         </navigation-case>
         <navigation-case>
                  <from-outcome>logout</from-outcome>
                  <to-view-id>goodbye</to-view-id>
  </navigation-rule>
  <navigation-rule>
         <from-view-id>home</from-view-id>
         <navigation-case>
                  <from-outcome>dashboard</from-outcome>
                  <to-view-id>dashboard</to-view-id>
         </navigation-case>
         <navigation-case>
                  <from-outcome>list</from-outcome>
                  <to-view-id>list-widgets</to-view-id>
         </navigation-case>
  </navigation-rule>
```

Which two define a valid flow of view IDs through the application? (Choose two.)

A. home > goodbye > list-widgets

B. dashboard > home > add-widget > list-widgets

C. list-widgets > add-widget > home > dashboard > home

D. home > list-widgets > add-widget > goodbye

ANSWER: C D

Explanation:

The valid flows are **list-widgets > add-widget > home > dashboard > home** and **home > list-widgets > add-widget > goodbye**. Each consecutive pair in these sequences follows a navigation outcome that is available from the current view in the supplied navigation-rule set. A JSF navigation rule is evaluated using the current view ID together with the action outcome; when a matching navigation case is found, JSF renders the target view ID. Consequently, a flow is valid only when every transition can be resolved from the view immediately before it, rather than merely containing view IDs that exist somewhere in the application.

The first valid sequence demonstrates that navigation can return to **home** after reaching **dashboard**, provided that the relevant navigation case is defined. The second sequence follows the available progression from **home** through the widget-listing and widget-addition views and then reaches **goodbye**. This is consistent with the JSF navigation model, in which navigation cases map outcomes from a source view to a destination view. See Oracle's Java EE 7 Tutorial: Configuring Navigation Rules and the Java EE 7 Tutorial: Navigation Model.

QUESTION NO: 12

Your client wants certain inventory items of high importance to be counted periodically to improve the average level of inventory accuracy.

What type of counting would you recommend?

A. Physical Count

B. Manual Count

C. Serialized Count

D. Cycle Count

E. Blind Count

F. Zero Quantity Count

ANSWER: D

Explanation:

Cycle Count is the appropriate recommendation because cycle counting is an ongoing inventory-control process in which selected items are counted at regular intervals rather than requiring a complete inventory count at one time. This approach lets an organization prioritize high-importance or high-value items and count them more frequently, helping identify and correct inventory-record discrepancies promptly. As a result, inventory accuracy improves continuously while operational disruption is minimized.

Oracle Inventory Management supports cycle counting as a method for maintaining inventory accuracy through periodic counts of specified items. Cycle count setup can assign items to a cycle count and define count intervals, allowing greater attention to items that have the highest business importance. In practice, organizations commonly use classification-based counting frequencies, so critical inventory is verified more often than less significant stock. This directly matches the requirement to count certain important items periodically in order to raise the overall average inventory-accuracy level. Oracle documentation describes cycle counting and its use for scheduled, recurring inventory verification in Oracle Fusion Cloud SCM: Cycle Counting. Further inventory-management guidance is available from Oracle Fusion Cloud SCM Inventory Management.

QUESTION NO: 13

What are two outcomes when a lot expires on an item with lot control enabled?

A. It remains an inventory, but is not considered on-hand when the user is performing min-max or reorder point planning calculations.

B. It is issued out of stores.

C. It cannot be transacted.

D. It is not included in cycle counts.

E. It cannot be reserved for a date beyond the expiration date.

ANSWER: A E

Explanation:

An expired lot remains recorded in inventory and continues to be part of the physical on-hand balance; expiration does not itself remove the material from inventory. However, Oracle Inventory planning logic excludes expired quantity from the usable on-hand supply considered by min-max and reorder-point replenishment calculations. This prevents planning from treating material that has passed its shelf-life date as inventory that can satisfy future demand. Oracle's inventory documentation describes lot expiration as an inventory control that affects availability and replenishment treatment while retaining lot traceability and quantity records.

Reservations also honor the lot's expiration date. A reservation for lot-controlled material must not allocate a lot to a demand date after that lot has expired. This ensures that material committed to an order, work requirement, or other demand is valid on the date for which it is reserved. Together, these controls preserve accurate inventory balances while preventing expired material from inflating replenishment supply or being committed for future fulfillment. See Oracle's documentation on lot and serial number control and inventory reservations.

QUESTION NO: 14

Given the code fragment:

```
<servlet-mapping>
   <servlet-name>firstServlet</servlet-name>
   <url-pattern>/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>secondServlet</servlet-name>
   <url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>thirdServlet</servlet-name>
   <url-pattern>/</url-pattern>
</servlet-mapping>
```

When the context root is requested http://host:port/context, how does the container resolve this mapping?

A. thirdServlet handles the request.

B. firstServlet handles the request.

C. The container throws an error at startup.

D. secondServlet handles the request.

ANSWER: C

Explanation:

The container throws an error at startup. Servlet URL-pattern mappings are established while the web application is being deployed, before requests are dispatched. A context-root request is represented by the empty-string mapping pattern, while other patterns can represent the default servlet path or path-prefix mappings. The deployment process must produce an unambiguous association between every URL pattern and the servlet that owns it. Where the code fragment declares the same effective mapping for more than one servlet, the container cannot choose a request target based on mapping precedence because the conflict exists at the mapping-definition level rather than at request-matching time. Consequently, the application configuration is invalid and the container must reject deployment or report the mapping conflict during startup.

It does not defer this issue until a request for the context root arrives, nor does it arbitrarily select one of the declared servlets. The Servlet specification defines URL-pattern mapping rules and requires servlet registration mapping conflicts to be detected; programmatic registration likewise reports conflicts through the set of mappings that could not be added. See the Java Servlet 3.1 specification and the ServletRegistration.Dynamic.addMapping API documentation.

QUESTION NO: 15

Which is a valid description of REST?

A. REST provides the same architectural properties as SOAP.

B. REST is a library that is part of JEE called JAX-RS.

C. REST is the conventional way of interacting with information resources.

D. REST is a Web Services standard supported by JEE and JAX-RS.

E. REST is an architectural style for designing networked applications around resources and a uniform interface.

ANSWER: E

Explanation:

REST is an architectural style for designing networked applications in which resources are identified by URIs and manipulated through a uniform interface, commonly the standard methods of HTTP. A RESTful service exposes resource representations, such as JSON or XML, and clients use those representations along with HTTP semantics to retrieve or change application state. The defining REST constraints include client-server separation, stateless requests, cacheability, a uniform interface, layered systems, and optional code on demand. These constraints promote loose coupling and scalability because each request carries the information needed for processing rather than relying on server-side conversational state.

In Java EE, JAX-RS is the Java API used to implement RESTful web services; it supplies annotations and runtime facilities for mapping HTTP requests to resource classes and methods. That API support does not redefine REST itself: REST remains the underlying architectural approach. Oracle's Java EE tutorial describes JAX-RS as an API for developing RESTful web services and explains resources and their URI-based interaction model. See the Java EE 7 Tutorial: JAX-RS and the REST architectural style description.

QUESTION NO: 16

Which three statements are true for an expense destination transfers that does NOT require a receipt at destination inventory organization?

A. There is no cost associated to the transaction.

B. There is no put away transaction in Inventory since the item is expensed.

C. A receipt is required on interorganization expense destination transfer orders between the from and to organizations.

D. The transfer order is considered received and delivered at the time of shipment.

E. The destination inventory is not incremented.

ANSWER: B D E

Explanation:

For an expense destination transfer that is configured so that no receipt is required at the destination inventory organization, the shipment transaction completes the destination-side processing automatically. Consequently, the transfer order is treated as both received and delivered when the source organization ships the material. There is no separate receiving or delivery activity awaiting action at the destination.

Because the destination is an expense destination, the material is not added to destination on-hand inventory. Therefore, the destination inventory balance is not incremented. For the same reason, Inventory Management does not require a put-away transaction: put-away is used to place received material into inventory storage, whereas expensed material is delivered for use or consumption rather than stocked as destination inventory.

This behavior supports a streamlined internal-material flow for supplies that must be issued from a source inventory organization but are not intended to become controlled on-hand stock in the destination organization. Oracle's transfer-order documentation describes how transfer orders manage shipment, receipt, and delivery processing, while its Inventory Management documentation describes inventory transaction and destination behavior. See the Oracle Fusion Cloud Inventory Management transfer-order documentation and the Oracle Fusion Cloud Inventory Management documentation.

QUESTION NO: 17

: 15

Given:

```
// line 1
public class Mymapper {
        @PrePassivate
        public void setPinDrop() {}
        @PostActivate
        public void checkLocation() {}
}
```

What code needs to be added to line 1 for MyMapper bean instances to be correctly passivated?

A. @Stateless @PassivationCapable

B. @Stateless

C. @Stateful @PassivationCapable

D. @Stateful

ANSWER: D

Explanation:

`@Stateful` is correct because it declares `MyMapper` as a stateful session bean, whose lifecycle includes container-managed passivation and activation. A stateful session bean retains conversational client-specific state across method invocations. When the container needs to conserve resources, it may passivate an idle bean instance by saving its conversational state, then later activate that instance when the client invokes it again. Declaring the bean with `@Stateful` therefore gives the container the EJB lifecycle semantics required to manage passivation. For a bean to be eligible for this lifecycle, its state and any relevant injected references must meet the EJB passivation requirements; the container validates and manages those requirements as part of deploying and operating the stateful session bean. The Java EE 7 Tutorial describes stateful session beans as preserving state for a particular client and notes that the container can passivate them to secondary storage. See Oracle's Stateful Session Beans documentation and the EJB 3.2 specification for the stateful-session-bean lifecycle and passivation rules.

QUESTION NO: 18

Your company is an automobile spares manufacturing organization, which follows a discrete process. It has its own manufacturing and distribution centers located globally.

It has these business units:

US - USA business unit

CAN - Canadian business unit

UK - UK business unit

MX - Mexican business unit

IND - India business unit

SPA - Spain business unit

FRA - France business unit

NL - Netherlands business unit

GER - Germany business unit

These are the inventory organizations that exist in each business unit:

Which two statements are true?

A. All inventory organizations do not need to be in the same business unit to assign items.

B. Item organizations are also supported to serve as inventory organizations.

C. Items cannot be assigned to inventory organizations across business units.

D. Items defined in the GM organization can be assigned to US1, US2, NL1, and MX1 inventory organizations.

E. Operational Attributes can be controlled only at the Master Level.

F. You can define an item in the IN1 inventory organization and assign it to the DE1 inventory organization.

ANSWER: A B D

Explanation:

Oracle Fusion item management separates the item-definition structure from the business-unit structure. An item master organization can maintain the centrally governed item definition and make that item available to eligible child item organizations, including inventory organizations belonging to different business units. Therefore, all inventory organizations do not need to belong to one business unit for item assignment; the governing requirement is the item-organization and item-master relationship used for the assignment. This design supports enterprises that centrally manage a common item catalog while conducting inventory and fulfillment operations in multiple legal or operational regions.

Item organizations are also supported to serve as inventory organizations. When an organization is enabled for inventory, it can hold on-hand balances and support inventory transactions while also participating in item-definition and assignment processes. In the stated organization model, the GM organization is the master source for the item definition, and US1, US2, NL1, and MX1 are eligible inventory organizations to which that GM-defined item can be assigned. This demonstrates centralized item governance with distribution to operational organizations across business units. Oracle describes the item-organization model in its Item Organizations documentation and provides related implementation guidance in the Inventory Organizations documentation.