UF4. Componentes de acceso a datos.

Actividades PAC01 (Solución):

1. Realiza las actividades 1, 2, 3 y 4 de la página 300 del libro:

1. ¿Qué es un componente? Cita algunas ventajas e inconvenientes del uso de componentes.

Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Ventajas del uso de componentes:

- Reutilización del software.
- Disminución de la complejidad del software.
- Mejora del mantenimiento del sistema. Los errores son más fáciles de detectar.
- Incremento de la calidad de software. Un componente puede ser construido y luego mejorado.

Inconvenientes del uso de componentes:

- Solo existen en algunos campos y no siempre se pueden encontrar los componentes adecuados para cada proyecto.
- Falta de estándares y de procesos de certificación que garanticen la calidad de los componentes.

2. ¿Qué características debe tener un componente?

- Independiente de la plataforma: Hardware, Software, Sistema Operativo.
- Identificable: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
  Autocontenido: Un componente no debe requerir de la utilización de otros componentes para llevar a cabo la función por la cual fue diseñado.
- Puede ser reemplazado por otro componente: Se puede remplazar por nuevas versiones u otro componente que lo mejore.
- Con acceso solamente a través de su interfaz: Una interfaz define el conjunto de

operaciones que un componente puede realizar; Estas operaciones se llaman también servicios o responsabilidades. Las interfaces proveen un mecanismo para interconectar componentes y controlar las dependencias entre ellos.
- Sus servicios no varían: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.
- Bien documentado: Un componente debe estar correctamente documentado para *facilitar su búsqueda* si se quiere actualizar, integrar con otros, adaptarlo, etc.
- Es genérico: Sus servicios deben servir para varias aplicaciones.
- Reutilizado dinámicamente: Puede ser cargado en tiempo de ejecución en una aplicación.
- Se distribuye como un paquete: En este paquete se almacenen todos los elementos que lo constituyen.

3. ¿Cuál es la diferencia entre el modelo de componentes y la plataforma de componentes?

La forma concreta de especificar, implementar o empaquetar un componente depende de la tecnología utilizada. Las tecnologías usadas en componentes incluyen dos elementos:

- Modelo de componentes. Especifica las reglas de diseño que deben obedecer los componentes, sus interfaces y la interacción entre componentes.
- Plataforma de componentes. Es la infraestructura de software requerida para la ejecución de aplicaciones basadas en componentes. Se basan en un determinado modelo de componentes.

4. ¿Qué características debe cumplir un JavaBean?

Un JavaBean ha de cumplir las siguientes características:

- Introspección. Mecanismo mediante el cual los propios JavaBeans proporcionan información sobre sus características (propiedades, métodos y eventos). Los Beans soportan la introspección de dos formas: utilizando patrones de nombrado (que son como reglas para nombrar las características del Bean) y proporcionando las características mediante una clase *Bean Information* relacionada.
- Manejo de eventos. Los Beans se comunican con otros Beans utilizando los eventos. Un Bean puede tener interés en recibir y responder a eventos enviados por otro Bean.
- Propiedades. Las propiedades definen las características de apariencia y comportamiento de un Bean que pueden ser modificadas durante el diseño de los componentes.
- Persistencia. Permite a los Beans almacenar su estado y restaurarlo posteriormente. Se basa en la serialización.
- Personalización. Los programadores pueden alterar la apariencia y conducta del

Bean durante el diseño.

2. Realiza la actividad 1 de la página 283.

1. Crea un proyecto en Eclipse para probar la librería (*LibreriaJava1.jar*). Define varios productos y cambia el stock actual en alguno de ellos. Visualiza por cada producto su stock mínimo.

Podéis encontrar el proyecto Eclipse con la propuesta de solución en la carpeta PruebaLibreriaJava2

3. Realiza la actividad 2 de la página 290 que nos solicita (lo explicamos porque quizá en el enunciado del libro no acaba de quedar del todo claro):

a) modificar la clase llenarVentas para que recoja el ID de Producto y la Cantidad a través de la línea de comandos.

b) modificar de nuevo la clase llenarVentas para que le solicite al usuario que introduzca el ID de producto y la Cantidad a través de la entrada estándar.

Enviar ambas opciones.

Podéis encontrar la propuesta de solución en la carpeta Actividad 3. El fichero LlenarVentas_v2 se corresponde con el apartado a) y LlenarVentas_v3 se corresponde con el apartado b).