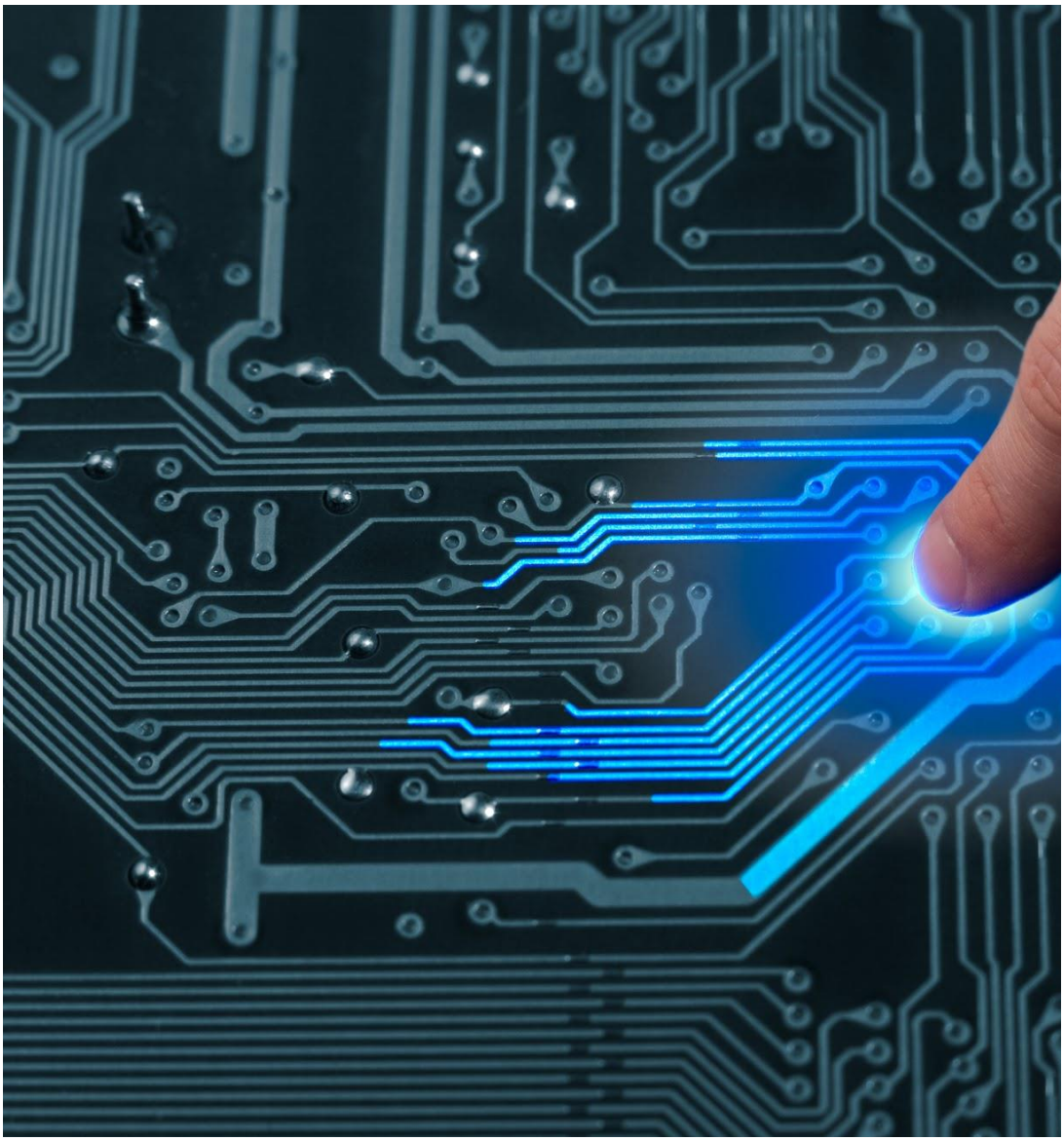
CFGS

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

ACCESO A DATOS

UF4. Componentes de acceso a datos

PAC 1: Componentes de acceso a datos



PAC 1: Componentes de acceso a datos

INFORMACIÓN
Para responder a las siguientes cuestiones deberás ayudarte del material didáctico y consultar internet.

Requisitos varios que deben cumplirse en vuestros trabajos:

- En los ejercicios, si se requieren de cálculos, estos deben aparecer en la respuesta que planteéis.
- Siempre que utilicéis información de Internet para responder / resolver alguna pregunta, tenéis que citar la fuente (la página web) de dónde habéis sacado aquella información.
- Siempre que utilicéis información del libro digital para responder / resolver alguna pregunta, tenéis que citar el tema y la página de dónde habéis sacado aquella información.
- No se aceptarán respuestas sacadas de Internet utilizando la metodología de copiar y pegar. Podéis utilizar Internet para localizar información, pero el redactado de las respuestas ha de ser vuestro.
- Las respuestas a las preguntas deben estar bien argumentadas, no se admiten respuestas escuetas o monosílabas.
- Si el proyecto no compila, la puntuación máxima será un 4.
- Se valorará la presentación, ortografía y gramática de vuestro trabajo hasta con un punto de la nota final.

Entrega de la PAC:

- Se debe entregar junto con el proyecto comprimido, una memoria en el que se expliquen los pasos realizados junto con los puntos más importantes del código explicados.

Ejercicio teórico

Ejercicio 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.

Ejercicio 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.

Ejercicio 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.

Ejercicio práctico

Ejercicio 4: ……..

Pues bien, ejecútalo y analiza la respuesta sobre pantalla a partir del código en las clases *Pedido*, *Producto* y *PruebaLibreriaJava1*.

Al ejecutar el programa, el mensaje sobre pantalla es

```
Stock anterior: 10
Stock actual: 2
REALIZAR PEDIDO EN PRODUCTO:Dabber Sur Femme 2011
```

que nos indica que hay que realizar un pedido del producto. En la inicialización del producto se ha fijado el valor mínimo de stock (*stockminimo*) de 3 artículos,

```
Producto producto = new Producto(1, "Dabber Sur Femme 2011", 10, 3, 16);
```

Se comienza con un stock de 10 artículos, y cuando se actualiza a 2 artículos (lo que significaría que se han vendido 8 artículos), salta el mensaje de aviso. Esto es resultado de que se lance el método *firePropertyChange* al cumplirse la condición (stockactual < getStockminimo()).

Responde (teniendo en cuenta las definiciones en el material didáctico):

- ¿Qué propiedades de los Beans creados son simples?

  Las propiedades simples se reconocen muy fácilmente porque los métodos asociados son setters y getters muy básicos y simples.

  Bean *Producto*: los atributos *descripcion, idproducto, stockminimo* y *pvp*.

Bean *Pedido*: los atributos numeropedido, producto, fecha, cantidad y pedir.

- ¿Qué propiedades de los Beans creados son indexadas?

  Las propiedades indexadas también son atributos del Bean que tienen asociados simples setters y getters, pero se diferencian de las propiedades simples en el hecho de que los métodos setters y getters asociados devuelven vectores (arrays).
  No hay ninguna propiedad indexada; ni en Productos, ni en Pedidos.

- ¿Qué propiedades de los Beans creados son ligadas?

  Las propiedades ligadas son atributos de los Beans que tienen métodos setters que pueden lanzar eventos (*firePropertyChange*).
  La única propiedad ligada es *stockactual* que lanza un evento cuando es menor que *stockminimo*. El evento es imprimir un mensaje por pantalla que avise de que el *stockactual* es escaso y se está por debajo de lo que se considera mínimo para poder hacer frente a los pedidos.

- ¿Qué propiedades de los Beans creados son restringidas?

  Las propiedades restringidas tienen setters que lanzan eventos (*firePropertyChange*) cuando los valores que se intentan asignar al atributo no son válidos.
  No hay propiedades restringidas ni en Productos, ni en Pedidos.

Define nuevos productos en la clase *PruebaLibreriaJava1* y cambia el stock actual en alguno de ellos. Visualiza por cada producto su stock mínimo.

Valía cualquier objeto de tipo Producto que se cree,

```
Producto producto2 = new Producto(2, "Libros de informática", 30, 5, 40);

Producto producto3 = new Producto(3, "Monitores de ordenador", 20, 5, 150);
```

Y luego se cambia el stock actual

```
Pedido pedido2 = new Pedido();
producto2.addPropertyChangeListener(pedido2);
producto2.setStockactual(2);

Pedido pedido3 = new Pedido();
producto2.addPropertyChangeListener(pedido2);
producto2.setStockactual(10);
```

observando los mensajes de pantalla si salta algún mensaje al reducir el *stockactual* por debajo de *stockminimo*.

¡Buen trabajo!