Concepto de componente

Introducción

- Componente: unidad de composición de aplicaciones software.
- Un nuevo paradigma: DSBC: desarrollo de software basado en componentes → trata de sentar las bases para el diseño y desarrollo de aplicaciones distribuidas basadas en componentes de software reutilizables
- **Caracteristicas:**
  - **Independiente de la plataforma**
  - **Identificable**
  - **Autocontenido**
  - **Puede ser reemplazado por otro componente**
  - **Con acceso solamente a través de su interfaz**
  - **Sus servicios (funcionalidades) no varían (puede cambiar la implementación)**
  - **Bien documentado**
  - **Se distribuye como un paquete**

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

- Modelo de componentes: especifica la reglas de diseño de los componentes y sus interfaces
- Plataforma de componentes: es la infraestructura de software requerida para la ejecución de aplicaciones basadas en componentes.

Ejemplo de tecnologías de componentes son:

- La plataforma .NET de Microsoft para sistemas Windows.
- JavaBeans y EJB de Oracle.

Nosotros veremos algo de JavaBeans

JavaBeans

- Un JavaBean es un componente de software reutilizable que está escrito en lenguaje Java.
- Características:
  - **Introspección**: mecanismo mediante el cual los propios JavaBeans proporcionan información sobre sus características. Los Beans soportan la introspección de dos formas: utilizando patrones de nombrado y proporcionando las características mediante una clase Bean Information relacionada.
  - **Manejo de eventos**: los Beans se comunican con otros mediante eventos.
  - **Propiedades**: definen las características de apariencia y comportamiento
  - **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.

- Una definición más detallada: un JavaBean es una clase Java que se define a través de las propiedades que expone, los métodos que ofrece y los eventos que atiende o genera. Su implementación requiere cumplir ciertas reglas:
  - Debe tener un constructor sin argumentos, aunque puede tener más de uno.
  - Debe implementar la interfaz **Serializable** (para poder implementar persistencia)
  - Sus propiedades deben ser accesibles mediante métodos **get** y **set.**
  - Los nombres de los métodos deben obedecer a ciertas convenciones de nombrado

Propiedades y atributos

- Las propiedades de un Bean son los atributos que determinan su apariencia y comportamiento.
- Para acceder a las propiedades deben existir los correspondientes getter y setter
- Las propiedades pueden ser simples, indexadas, ligadas o restringidas

Propiedad simple

- Representan un único valor
- Si la propiedad es booleana se escribe “isNombrePropiedad()” para obterner su valor

Propiedades indexadas

- Representan un array de valores a los que se accede mediante índice.
- Se deben definir métodos getter y setter para acceder al array completo y a valores individuales

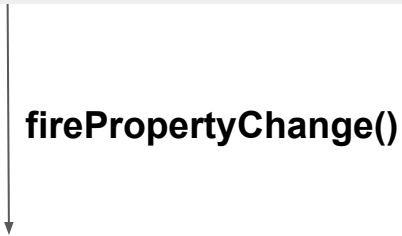
```
private int[] categorias={1,2,3};

public void setCategorias(int[] valor){
        this.categorias=valor;
public int[] getCategorias() { return this.categorias;}

public void setCategorias(int indice, int valor) { this.categorias(indice)=valor;}

public int getCategorias(int indice) {return this.categorias(indice);}
```

Propiedades ligadas

- Son las propiedades asociadas a eventos.
- Cuando la propiedad cambia se notifica a todos los objetos interesados en el cambio, permitiéndoles realizar alguna acción.
- Para que el Bean soporte propiedades ligadas (o compartidas) debe mantener una lista de los receptores de la propiedad y alertar a dichos receptores cuando cambie la propiedad, para ello proporciona una serie de métodos de la clase **PropertyChangeSupport**.
- La lista de receptores se mantiene gracias a los métodos **addPropertyChangeListener()** y **removePropertyChangeListener()**

JavaBean Fuente de Eventos

addPropertyChangeListener()
removePropertyChangeListener()



JavaBean Receptor de Eventos

propertyChange()

Ejemplo

Vamos a crear dos Beans:

- El primer Bean (fuente) de nombre Producto tiene una propiedad ligada denominada *stockactual* de tipo int.
- El segundo Bean (receptor) de nombre Pedido está interesado en los cambios de dicha propiedad
- El problema es que cuando el stock actual de un producto sea inferior al stock mínimo se debe generar un pedido.

(Ver código en el moodle)

Ejercicio práctico JavaBeans

Crea un JavaBean que represente el Empleado de una compañía. El Bean Empleado dispone de:

- Cuatro propiedades: NIF, nombre, cargo que ocupa y sueldo
- Dos constructores:
  - Empleado(): Otorga a los campos cargo y sueldo los valores de “Junior” y 1000€ respectivamente
  - Empleado(NIF, nombre) : llama al constructor anterior y otorga los valores de NIF y nombre pasados como argumentos

Cuando se modifica el valor de “cargo” del Bean se comprueba que la modificación realizada no ponga el atributo a “NULL” o lo deje en blanco. En caso de no ser ni lo uno ni lo otro, se notifica el cambio de la propiedad a un Bean oyente.

De la misma manera, cuando se modifica el valor de “sueldo”, el Bean Empleado comprueba que la modificación realizada sea superior a 0. En caso de serlo, se notifica el cambio de la propiedad al Bean oyente

A continuación, crea el Bean oyente. Lo llamaremos PanelEmpleado y es un Bean que puede leer las propiedades de sueldo y cargo del Bean Empleado.

El Bean PanelEmpleado dispone de:

- Dos atributos:
  - limiteVariacionSueldo: representa un límite de la variación del sueldo de un empleado ya sea un incremento o un decremento. Se trata de un valor entero entre 10 y 50.
  - listaDeCargos: representa una lista de cargos asignables al trabajador. Se trata de un véctor de Strings o similar. Deben existir todos los getters y setters necesarios según hemos visto en teoría. Por defecto, se inicializa con los valores: (“Junior”, “SemiSenior”, “Analista”, “CEO”)

El Bean PanelEmpleado dispone de dos constructores:

- PanelEmpleado(): Otorga al atributo limiteVariacionSueldo el valor de 10
- PanelEmpleado(int valor) : otorga al atributo limiteVariacionSueldo el valor indicado en el argumento.

Si el Bean PanelEmpleado recibe la notificación de una actualización en el valor del sueldo de un empleado:

- Calculará el porcentaje que varía el sueldo actual respecto al anterior.
- Si el porcentaje es superior al valor de limiteVariacionSueldo, PanelEmpleado generará una excepción que pueda ser capturada y mostrada por el Bean Empleado, mostrando un mensaje que detalle el error.
- A su vez, el Bean Empleado no podrá modificar el sueldo

Si el Bean PanelEmpleado recibe una notificación de que el atributo cargo de Empleado ha sido modificado:

- Comprobará que el nuevo cargo se encuentre dentro de la listaDeCargos.
- Si no está en la lista, generará una excepción que será capturada por el Bean Empleado mostrando un mensaje de error
- Si sucede esto, el cargo del empleado no podrá ser modificado

Finalmente, crea una clase llamada PruebasFinales que contendrá el main. Define un objeto Empleado y un objeto PanelEmpleado. Vincula ambos objetos y realiza un pequeño juego de pruebas para comprobar que ambos componentes funcionan correctamente.