Concepte de component

Java Beans

Consorci d'Educació de Barcelona
Generalitat de Catalunya
Ajuntament de Barcelona

Introducció

- Component: unitat de composició d'aplicacions programari.
- Nou paradigma: DSBC: Desenvolupament de Software Basat en Components. → tracta d'establir les bases pel disseny i desenvolupament d'aplicacions distribuïdes basades en components de programari reutilitzable.
- Característiques:
  - Independent de la plataforma.
  - Identificable.
  - Autocontingut.
  - Pot ser reemplaçat per un altre component.
  - Amb accés solament a través de la seva interfície.
  - Els seus serveis (funcionalitats) no varien (pot canviar la implementació).
  - Ben documentat.
  - Es distribueix com un paquet.

Introducció

- La manera d'especificar, implementar, o empaquetar un component depèn de la tecnologia utilitzada. Les tecnologies basades en components inclouen dos elements:
  - Model de components: especifica la regles de disseny dels components i les seves interfícies.
  - Plataforma de components: és la infraestructura de programari requerida per a l'execució d'aplicacions basades en components.
- Exemple de tecnologies de components són:
  - La plataforma .NET de Microsoft per a sistemes Windows.
  - JavaBeans i EJB d'Oracle.
- Nosaltres veurem una mica de JavaBeans

JavaBeans

- Un ***JavaBean*** és un component de programari reutilitzable que està escrit en llenguatge Java.
- Característiques:
  - **Introspecció**: mecanisme mitjançant el qual els propis *JavaBeans* proporcionen informació sobre les seves característiques. Els *Beans* suporten la introspecció de dues formes: utilitzant patrons d'anomenat i proporcionant les característiques mitjançant una classe Bean Information relacionada.
  - **Control d'esdeveniments**: els *Beans* es comuniquen amb altres mitjançant esdeveniments.
  - **Propietats**: defineixen les característiques d'aparença i comportament.
  - **Persistència**: permet als *Beans* emmagatzemar el seu estat i restaurar-lo posteriorment. Es basa en la serialització.
  - **Personalització**: els programadors poden alterar l'aparença i conducta del Bean durant el disseny.

JavaBeans

- Una definició més detallada: un JavaBean és una classe Java que es defineix a través de les propietats que exposa, els mètodes que ofereix i els esdeveniments que atén o genera. La seva implementació requereix complir unes certes regles:
  - Ha de tenir un constructor sense arguments, encara que pot tenir més d'un.
  - Ha d'implementar la interfície *Serializable* (per poder implementar persistència).
  - Les seves propietats han de ser accessibles mitjançant mètodes ***get*** i ***set***.
  - Els noms dels mètodes han d'obeir a unes certes convencions d'anomenament.

Propietats i atributs

- Les propietats d'un ***Bean*** són els atributs que determinen la seva aparença i comportament.
- Per a accedir a les propietats han d'existir els corresponents ***getter*** i ***setter.***
- Les propietats poden ser simples, indexades, lligades o restringides.

Propietats simples

- Representen un únic valor.
- Si la propietat és booleana s'escriu "isNomPropietat()" per obternir el seu valor.

Simple Bean Property Example

```
public class MyBean
{
    public MyBean() { }

    private String title;

    public String getTitle()
    {
        return this.title;
    }

    public void setTitle(String title)
    {
        this.title = title;
    }
```

Creates a new instance of MyBean

Holds Value of Property 'Title'

Getter Property for 'title'

Setter Property for 'title'

Propietats indexades

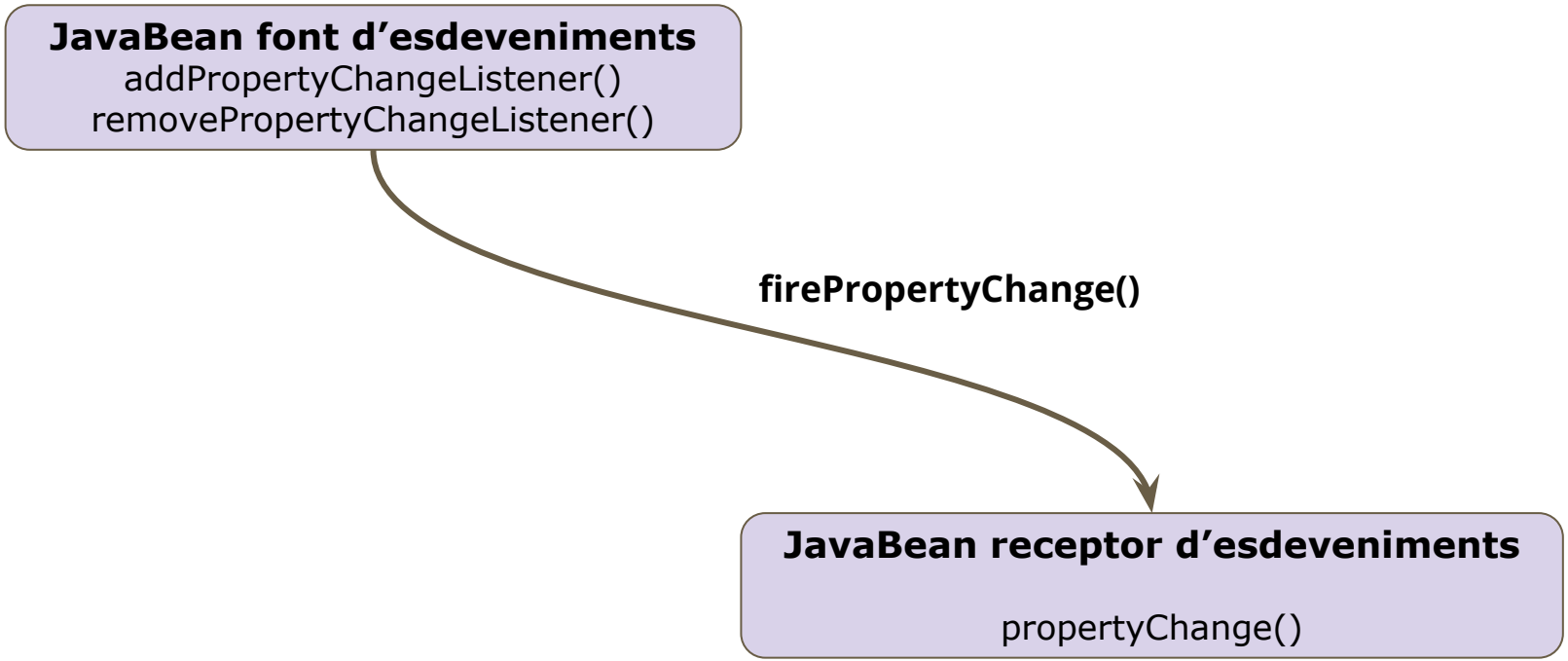
- Representen un *array* de valors als quals s'accedeix mitjançant un índex.
- S'han de definir mètodes ***getter*** i ***setter*** per accedir al *array* complet i als valors individuals.

```
private int[] categories={1,2,3};
public void setCategories(int[] valor){ this.categories=valor;}
public int[] getCategories() { return this.categories;}
public void setCategories(int index, int valor) {
this.categories(index)=valor;}
public int getCategories(int index) {return this.categories(index);}
```

Propietats lligades

- Són les propietats associades a esdeveniments.
- Quan la propietat canvia es notifica a tots els objectes interessats en el canvi, permetent-los realitzar alguna acció.
- Perquè el ***Bean*** suporti propietats lligades (o compartides) ha de mantenir una llista dels receptors de la propietat i alertar a aquests receptors quan canviï la propietat, per això proporciona una sèrie de mètodes de la classe. **PropertyChangeSupport**.
- La llista de receptors es manté gràcies als mètodes **addPropertyChangeListener()** i **removePropertyChangeListener()**.

Propietats lligades



Exemple

Crearem dos **Beans**:

- El primer **Bean (font)** de nom *Producte* té una propietat lligada denominada ***getStockactual*** de tipus ***int***.
- El segon **Bean (receptor)** de nom *Comanda* està interessat en els canvis d'aquesta propietat.
- El problema és que quan l'estoc actual d'un producte sigui inferior a l'estoc mínim s'ha de generar una comanda.

(Veure codi al moodle)

Activitat Java Beans

A1.- Crea un ***JavaBean*** que representi l'Empleat d'una companyia. El **Bean** Empleat disposa de:

- Quatre propietats: NIF, nom, càrrec que ocupa i sou.
- Dos constructors:
  - Empleat(): Atorga als camps càrrec i sou els valors de "Júnior" i 1000€ respectivament.
  - Empleat(NIF, nom): crida al constructor anterior i atorga els valors de NIF i nom passats com a arguments.

Quan es modifica el valor de "càrrec" del **Bean** es comprova que la modificació realitzada no posi l'atribut a "NULL" o el deixi en blanc. En cas de no ser ni l'un ni l'un altre, es notifica el canvi de la propietat a un **Bean** receptor.

De la mateixa manera, quan es modifica el valor de "sou", el Bean Empleat comprova que la modificació realitzada sigui superior a 0. En cas de ser-ho, es notifica el canvi de la propietat al **Bean** receptor.

Activitat Java Beans

A continuació, crear el **Bean** receptor. L'anomenarem **PanellEmpleat** i és un **Bean** que pot llegir les propietats de sou i càrrec del Bean Empleat.

El **Bean PanellEmpleat** disposa de:

- Dos atributs:
  - **limitVariacioSou**: representa un límit de la variació del sou d'un empleat ja sigui un increment o un decrement. Es tracta d'un valor sencer entre 10 i 50.
  - **llistaDeCarrecs**: representa una llista de càrrecs assignables al treballador. Es tracta d'un ***arrai de Strings*** o similar. Han d'existir tots els **getters** i **setters** necessaris segons hem vist en teoria. Per defecte, s'inicialitza amb els valors: ("Júnior", "semisènior", "Analista", "CEO").

El **Bean PanellEmpleat** disposa de dos constructors:

- **PanellEmpleat()**: Atorga a l'atribut **limitVariacioSou** el valor de 10.
- **PanellEmpleat(int valor)**: atorga a l'atribut **limitVariacioSou** el valor indicat en l'argument.

Activitat Java Beans

Si el **Bean PanellEmpleat** rep la notificació d'una actualització en el valor del sou d'un empleat:

- Calcularà el percentatge en que varia el sou actual respecte a l'anterior.
- Si el percentatge és superior al valor de **limitVariacioSou**, **PanellEmpleat** generarà una excepció que pugui ser capturada i mostrada pel **Bean Empleat**, mostrant un missatge que detalli l'error.
- Al seu torn, el **Bean Empleat** no podrà modificar el sou.

Si el **Bean PanellEmpleat** rep una notificació de que l'atribut càrrec d'Empleat ha estat modificat:

- Comprovarà que el nou càrrec es trobi dins de la **listaDeCarrecs**.
- Si no està en la llista, generarà una excepció que serà capturada pel **Bean Empleat** mostrant un missatge d'error.
- Si succeeix això, el càrrec de l'empleat no podrà ser modificat.

Activitat Java Beans

Finalment, crea una classe anomenada **ProvesFinals** que contindrà el ***main***. Defineix un objecte **Empleat** i un objecte **PanellEmpleat**. Vincula tots dos objectes i realitza un petit joc de proves per a comprovar que tots dos components funcionen correctament.