

## TD6 : JavaBean – Un Bean Complet

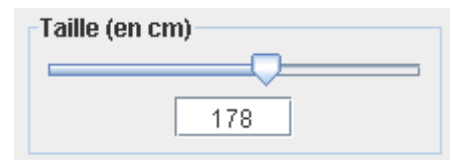
*Pour parfaire notre formation, nous allons construire un bean complet qui prévient de ses modifications via les événements. Gardez votre Guide du Développeur à portée de main, vous en aurez probablement besoin.*

### PRÉSENTATION DU BEAN

---

Notre Bean, que nous appellerons InfoPanel, permet de rentrer une valeur entière comprise entre une valeur minimale et une valeur maximale. Il ressemblera à ceci. On y trouve :

- Un bord avec label décrivant l'information
- Une barre de défilement (par exemple un JSlider) permettant de modifier la valeur (avec un minimum, un maximum)
- Un champ d'édition affichant la valeur et permettant de la modifier directement. Bien entendu, la barre de défilement et le champ d'édition doivent rester en concordance



### CRÉATION DE LA CLASSE SOURCE : INFOPANEL

---

#### Théorie

Le principe d'un composant est, comme vous l'avez vu, d'être « un morceau de code réutilisable ». Afin que ce code puisse communiquer avec l'extérieur (et ce, en faisant le moins d'hypothèse possible sur cet extérieur), tout composant doit pouvoir générer des événements en fonction de son état. Nous avons vu dans le TD précédent le processus simplifié. Si celui-ci est fort pratique, il ne concerne que les événements concernant les propriétés liées. Nous allons voir ici le processus complet reprenant une classe source, une classe d'événement et une classe d'écouteur.

#### Mise en pratique

Créez une classe de panneau d'information (InfoPanel) reprenant les spécifications données ci-dessus. Faites attention à bien respecter les principes de la programmation de composants tels que vu jusqu'à présent. Quelles propriétés devrait posséder ce bean ? Créez les assesseurs nécessaires. Cette classe va servir de classe source pour l'envoi d'événements.

## ÉVÉNEMENT ET ÉCOUTEUR ASSOCIÉS

---

### Théorie

La communication entre un composant et l'extérieur passe par la création d'un objet d'une classe événement [GDD]. Toute classe souhaitant pouvoir écouter les événements d'un type précis (une même source pouvant générer divers types d'événements) doit implémenter une interface d'écouteur correspondant. Celle-ci contiendra la ou les méthodes pouvant être appelées par la source sur chaque écouteur. Le principe du contrat représenté par l'interface permet d'être sûr que chaque classe inscrite possède effectivement la méthode demandée.

### Mise en pratique

Créez une classe d'événement liée à `InfoPanel` sous le nom `InfoEvent`. Veillez à nouveau à respecter les principes de la programmation de JavaBeans [GDD]. Dans un premier temps, votre classe se contentera d'implémenter un constructeur (voir l'API pour les méthodes de base fournies par `EventObject`).

Ceci fait, créez une interface `InfoListener` qui implémentera une seule méthode nommée `infoChanged`, prenant comme paramètre un objet de type `InfoEvent`. N'hésitez pas à consulter l'API pour y prendre des exemples d'écouteurs existants (`actionListener`, `windowListener`...).

## MODIFICATION DE LA CLASSE SOURCE

---

### Théorie

Il reste maintenant à intégrer le processus de gestion d'événement au niveau de la classe source. Celle-ci doit en effet implémenter deux fonctionnalités : d'une part, tenir à jour un registre d'objets écouteurs (soit permettre à des objets implémentant l'interface concernée de s'inscrire ou de se désinscrire) et, d'autre part, avertir chacun de ces objets lorsque certaines conditions sont remplies.

### Mise en pratique

La classe `InfoPanel` va être complétée pour tenir à jour un registre de classes implémentant `InfoListener`, afin de les avertir (via la méthode définie dans l'interface) lorsque la valeur de l'information est modifiée. A cette fin, il va vous falloir ajouter :

- Un attribut privé reprenant la liste des `InfoListeners` inscrits (la classe `Vector` est à préférer à ce niveau).
- Deux méthodes permettant à un `InfoListener` de s'inscrire/désinscrire de la dite liste.
- Une méthode privée créant un objet de type `InfoEvent` et l'envoyant à tous les écouteurs via la méthode définie dans `InfoListener`.

Ne reste plus alors qu'à appeler la méthode d'envoi à chaque changement de valeur de l'information.

## UTILISATION DU BEAN

---

Ajouter votre javabean à la palette de Netbeans et recodez l'application BMI.

