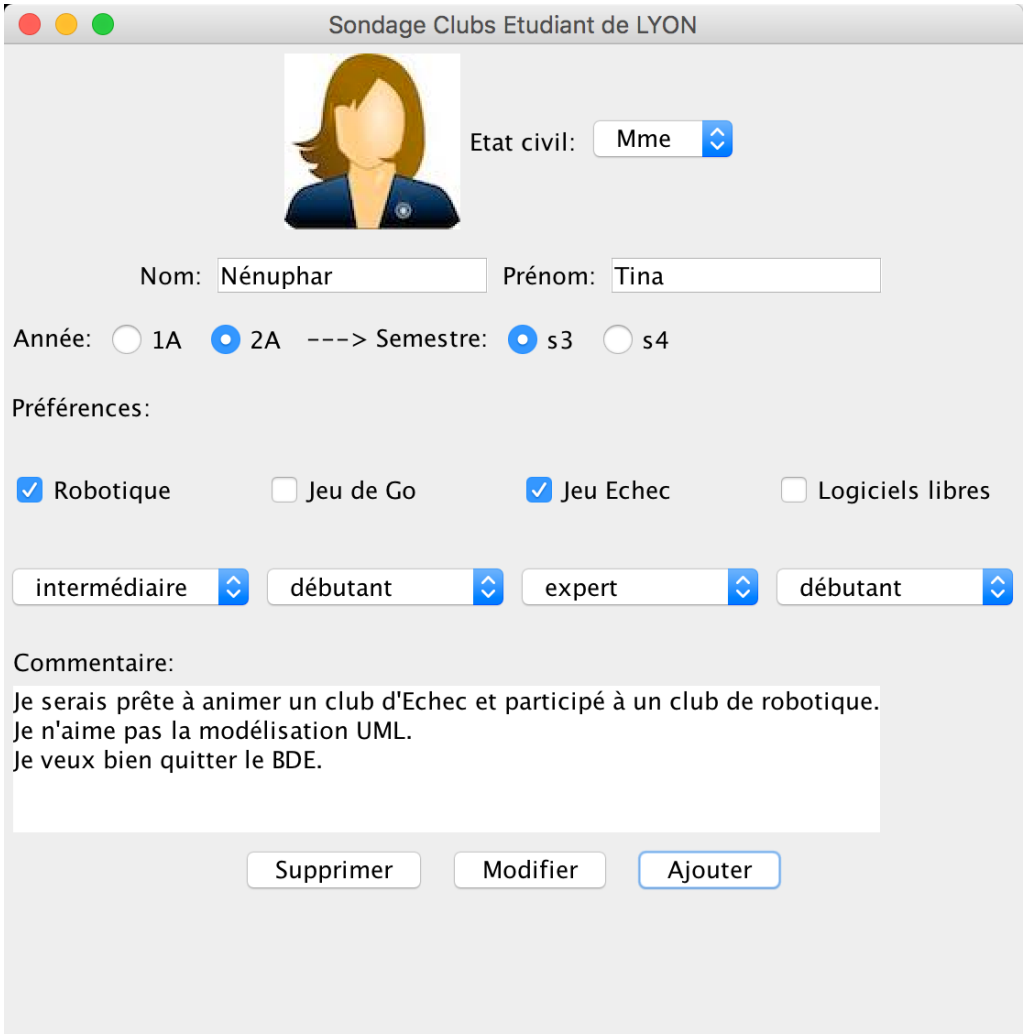
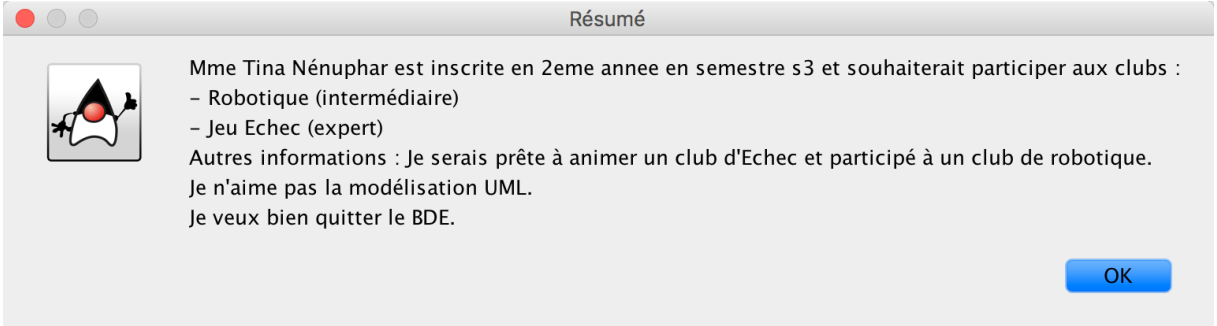
DévIHM – Mise en Pratique#2 : Gestion des évènements

Durée : 2h

L'objectif du TP est de **programmer la gestion des événements liés à l'interface Etudiant fournie**, sans l'aide de l'IDE. Nous allons mettre en place les 3 façons de gérer les écouteurs, vues en cours.





La gestion des événements s'effectue dans une méthode à part, souvent appelée `handleEvent()`, qui est lancée après `initComponents()` dans le constructeur. Cela permet de bien séparer les choses. Vous allez ainsi :

1. Ajouter des évènements de types `WindowEvent` pour l'ouverture et la fermeture de la fenêtre. Ici on utiliser l'écouteur `WindowListener` en **définissant la classe principale comme étant son propre écouteur**. Vous allez redéfinir toutes les méthodes de l'interface pour fournir un message approprié, notamment :

- `windowOpened()` ajoutera « de Lyon » au titre de votre fenêtre à son ouverture, avec la méthode `setTitle()` de `JFrame`. Pour récupérer le titre initial, on utilisera `getTitle();`

  Les autres méthodes de l'écouteur afficheront un **message** dans la fenêtre *output*.

  Q1- Aurait-on pu utiliser `WindowAdapter` ici ? Quel est le *type* d'*événement* que prennent toutes ces méthodes en argument ?

2. Faire en sorte que l'image utilisée **change** selon l'état civil sélectionné. On va utiliser un écouteur d'item pour la `comboBox` : définir l'écouteur `ItemListener` **en classe anonyme**.

   Q2- Quelle *unique* méthode faut-il donc définir ? Quel *événement* prend-elle en argument ? Notez la différence avec `WindowListener`.

   On utilisera la méthode `getItem`() de l'argument pour connaître l'état civil sélectionné et agir en conséquence (changer la photo).

3. On souhaite **changer le texte ainsi que la couleur du texte des boutons Supprimer et Modifier**, quand la souris passe sur le bouton. Utiliser l'écouteur `MouseListener` que vous codez **en classe interne**. Le texte du bouton doit être « Bouton en train d'être cliqué » et de couleur rouge. Pour changer la couleur du texte, vous pouvez utiliser la méthode `setForeground(Color.red)`. Vous constaterez ici que `e.getSource()` ne fonctionne pas ici pour modifier la couleur, son retour doit être converti en `JButton`.

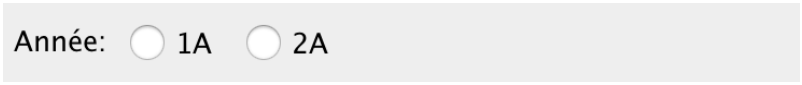
   Q3- Il existe aussi la méthode `getComponent()` d'Event : quelle *différence* existe-t-il entre `getSource()` et `getComponent ()` ?

   Pour cette question, modifier les deux méthodes suivantes :

   - `mouseEntered()` pour changer le texte du bouton quand la souris entre dans la « zone graphique » du bouton ;
   - `mouseExited()` pour rendre le bouton à son état initial quand la souris quitte la « zone graphique » du bouton.

Les autres méthodes de `MouseListener` **afficheront simplement** ce qui se passe pour le bouton **dans la fenêtre texte *output* :** « bouton XXX : cliqué, pressé, relâché ».

4. **Gérer l'affichage des semestres**
   - Initialement les boutons des semestres sont cachés :



   - Gérer les actions sur les radio-boutons pour synchroniser l'affichage en créant une 2ème **classe interne dédiée** avec l'écouteur `ActionListener`. Si l'utilisateur sélectionne le bouton radio 1A, les 2 boutons S1 et S2 seront affichés. S'il sélectionne le bouton 2A, les 2 boutons S3 et S4 seront affichés :

   Indice : utilisez if (`rb_xA.isSelected()`*)… et jouez sur la visibilité des composants associés aux semestres.*

   *Q4- Aurait-on pu utiliser la **même** classe interne que celle de la Q3 pour gérer le format des boutons ? Justifiez.*

5. **Réinitialiser tous les champs de la fenêtre** quand on clique sur le bouton « **Supprimer »** : champs texte vides, état civil à « Mme » avec l'image associée, boutons décochés, boutons des semestres cachés, zone commentaire réinitialisée.

6. **Afficher les informations de l'étudiant** quand on clique sur le bouton « **Ajouter** » sous la forme suivante :

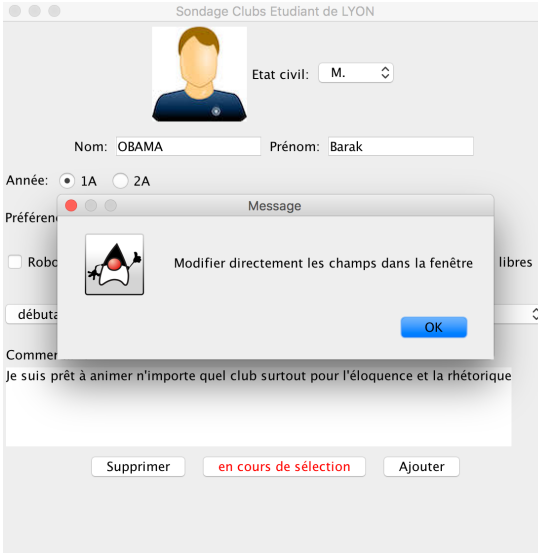
   « M/Mme *prénom nom* est inscrit(e) en *1ère/2ème* année en semestre *S1/S2/S3/S4* et et souhaiterait participer aux clubs : xxx. Autres informations : *(texte de la zone commentaire)* ». L'affichage reprend les informations saisies et des boutons sélectionnés.

   - Dans un premier temps, faites **l'affichage dans la console** avec la méthode `System.out.println()`.
   - Dans un deuxième temps, faites-en sorte que toutes ces informations soient affichées **dans une fenêtre `MessageDialog`**, qui s'ouvre quand on clique sur le bouton « Ajouter ». L'affichage de la fenêtre de Dialogue est fourni, qui utilise :

   JOptionPane.showMessageDialog(la-fenetre-courante, la-chaine-a-afficher, "titre", JOptionPane.INFORMATION_MESSAGE);

EN OPTION

7. Avant d'ajouter le texte récapitulatif du point 6, **vérifier que les champs** sont remplis et ne sont pas nuls.
8. Ajouter du code pour que **l'effet sur les boutons soit facultatif**, par exemple choisi au moment de la création de la fenêtre (*indice : nouveau constructeur*).
9. (code fourni) Ajouter du code pour le **bouton modifier** : quand on clique dessus, une boîte de dialogue (*MessageDialog, cf Cours 3)* s'ouvre mentionnant qu'il faut modifier directement les champs de la fenêtre.



10. (code fourni) Ajouter du code à la **fermeture de la fenêtre** : afficher une boite de dialogue pour confirmer qu'on veut réellement quitter (*ConfirmDialog, cf Cours 3*). Même chose pour le **bouton supprimer** : confirmer qu'on veut réellement supprimer