

Java Avancé – TP2 Gestion des événements

Durée : 1 séance, pgm + CR à envoyer pour début de séance 3

L'objectif du TP est de **programmer la gestion des événements liés à l'interface Etudiant réalisée au TP1**, toujours sans l'aide de l'IDE. Nous allons mettre en place les 3 façons de gérer les écouteurs, vues en cours.



Sondage Clubs Etudiant de LYON

 Etat civil: Mme

Nom: Nénuphar Prénom: Tina

Année: 1A 2A ---> Semestre: s3 s4

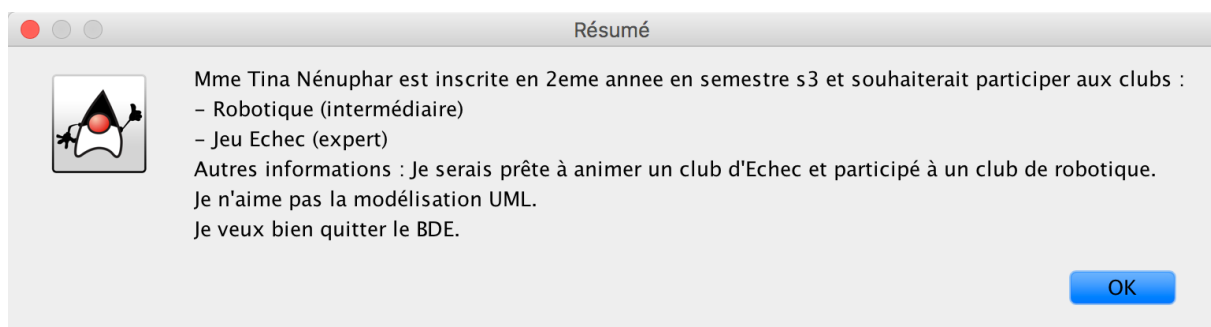
Préférences:

Robotique Jeu de Go Jeu Echec Logiciels libres


intermédiaire débutant expert débutant

Commentaire:
Je serais prête à animer un club d'Echec et participé à un club de robotique.
Je n'aime pas la modélisation UML.
Je veux bien quitter le BDE.

Supprimer Modifier Ajouter



Résumé

 Mme Tina Nénuphar est inscrite en 2eme annee en semestre s3 et souhaiterait participer aux clubs :

- Robotique (intermédiaire)
- Jeu Echec (expert)

Autres informations : Je serais prête à animer un club d'Echec et participé à un club de robotique.
Je n'aime pas la modélisation UML.
Je veux bien quitter le BDE.

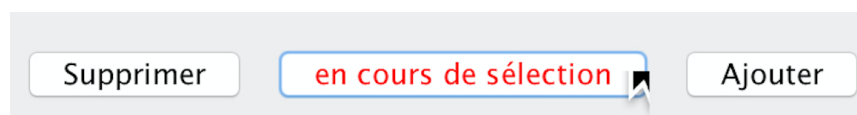
OK

Vous allez ajouter les évènements suivants à votre interface :

1. Ajouter des évènements de types `WindowEvent` pour l'ouverture et la fermeture de la fenêtre. Ici on utilise 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 :
 - `windowClosing()` affichera le message « Application Fermée » au moment de la fermeture de la fenêtre, dans la fenêtre *output*.
 - `windowOpened()` ajoutera « de Lyon » au titre de votre fenêtre au moment de l'ouverture de la fenêtre avec la méthode `setTitle()` de `JFrame`. Pour récupérer le titre initial, on utilisera `getTitle()` ;
2. *Aurait-on pu utiliser `WindowAdapter` ici ? Quel est le type d'événement que prennent toutes ces méthodes en argument ? Comment sont gérés les cas considérés (ouverture, iconification, etc.) ?*
3. Faire en sorte que l'image utilisée **change** selon l'état civil sélectionné. Pour cela, il faut charger une nouvelle image. Pour l'écouteur d'événement, la `comboBox` fonctionne avec un écouteur d'item. On définira cette fois l'écouteur `ItemListener` **en classe anonyme**. *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.

4. 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)`. Dans 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.



*Comment définir un bord normal ? Une solution peut-être de récupérer celui d'un bouton dont on n'a pas modifié le bord. Une autre (**moins bien**) est de réinitialiser le bouton en créant une instance `JButton()` ayant l'apparence par défaut.*

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

5. Gérer l'affichage des semestres.

- Pour cela, il faut d'abord **ajouter une ligne dans le formulaire** qui contient : un label pour « semestre » et 4 boutons radios qui sont (S1, S2, S3 et S4). Faire en sorte qu'initialement les boutons des semestres soient cachés. 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 comme le montre la figure suivante :

Année: 1A 2A ---> Semestre: s1 s2

Année: 1A 2A ---> Semestre: s3 s4

- Gérer ensuite les actions sur les radio-boutons pour synchroniser l'affichage en créant une 2^{ème} **classe interne dédiée**. Utiliser l'écouteur `ActionListener`.

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

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

6. **Réinitialiser tous les champs de la fenêtre** quand on clique sur le bouton « **Supprimer** ». Il faut que tous les champs texte soient vides, l'état civil soit « M. » avec l'icône image associé, il faut que tous les boutons soient décochés, les boutons des semestres cachés, et la zone commentaire remise à vide.

7. **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 1^{ère}/2^{ème} année en semestre S1/S2/S3/S4 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 ». Voici le code à utiliser que nous détaillerons au Cours 3 (fenêtres de dialogue) :

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

TRAVAIL A RENDRE

Déposer votre CR de TP2 + code source **sur TOMUSS**.

Pour ceux qui ont le temps, faites les **tests unitaires** pour tous les cas possibles de saisies, afin de valider votre méthode de résumé des informations.

Remarque : commentez *pertinemment* les différentes parties de votre code.

FACULTATIF

8. Avant d'ajouter le texte récapitulatif de la Q7, **vérifier que les champs** sont remplis et ne sont pas nuls.
9. 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*).
10. 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.
11. Ajouter du code à la **fermeture de la fenêtre** : afficher une boîte 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

