Programmation événementielle
TP 4 – Interface graphique (partie 2)

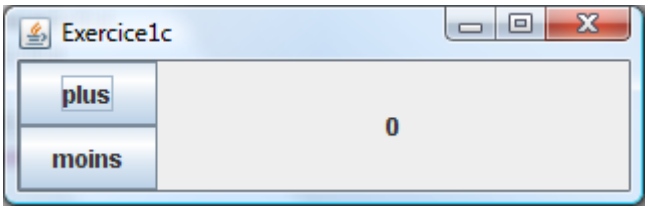
Exercice 1 – Réagir aux actions avec des *listeners*

a) Utiliser les événements

Les événements qui sont générés par les composants permettent de transporter des informations que l'objet listener qui les reçoit peut récupérer. Ces informations dépendent de la nature de l'événement, mais tous les événements peuvent fournir une référence sur l'objet qui l'a généré avec la méthode getSource().

Par exemple on peut brancher deux boutons sur un même listener et vouloir réagir différemment aux deux sources (on pourrait aussi mettre deux listeners différents, un sur chaque bouton, mais ce n'est pas le sujet de cet exercice…)

Ecrivez l'interface Exercice1 permettant de créer l'interface suivante afin qu'un bouton incrémente le compteur et que le deuxième le décrémente. Utiliser impérativement un seul objet listener dans une classe externe ou une classe interne non anonyme pour gérer les deux boutons.



Indication importante :

- On pourra utiliser la méthodes ***Object getSource()*** de la classe ***Event*** pour obtenir l'objet à l'origine de l'événement (c'est-à-dire une référence sur le bouton plus ou sur le bouton moins).
- Il ne faut pas oublier à this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  pour permettre une fermeture propre de la fenêtre si vous ne gérez pas l'événement de fermeture.

b) Question de fenêtres

Il est possible de trouver les listeners associés à certains composants en repérant dans la documentation les méthodes ayant la forme addXXXXListener.

Trouvez l'interface listener associé aux événements de la fenêtre JFrame (fermeture, réduction, etc). Modifiez l'exercice précédent de manière à ce que le programme s'arrête correctement lorsque l'on ferme la fenêtre en utilisant un listener.

Indications :

- Pour arrêter un programme : ***System.exit(0) ;***
- L'interface listener qui nous intéresse concerne des événements sur la fenêtre : Window en anglais…

c) Des « Adapters » pour simplifier

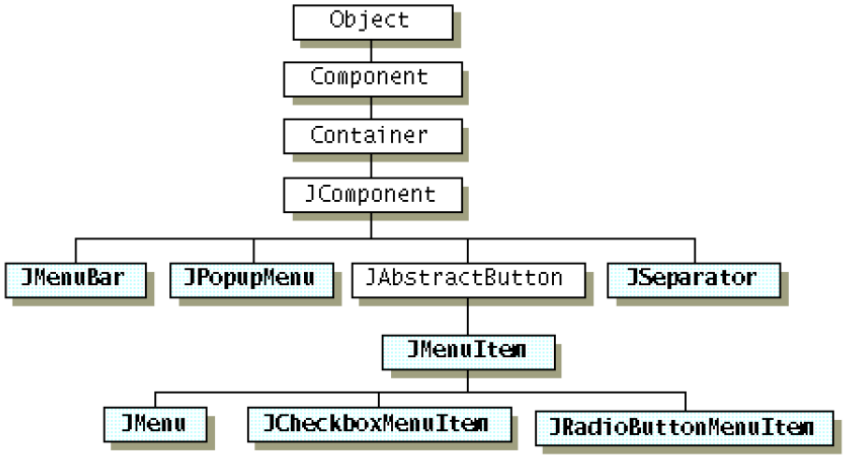
Dans l'exercice précédent, il a fallu définir les 7 méthodes de l'interface alors que seule la méthode ***windowClosing*** nous a été utile. Par facilité il est possible d'utiliser la classe WindowAdapter qui est une classe implémentant l'interface WindowListener (en ne faisant rien). Il suffit donc d'hériter de cette classe et de ne redéfinir que la méthode voulue.

Remarque : il existe des Adapters pour toutes les interfaces Listeners complexes

Modifiez et simplifiez la classe précédente en utilisant la classe WindowAdapter

Exercice 2 – Menus

Dans un JFrame, on peut définir une barre des menus, grâce aux classes JMenuBar, JMenu et JMenuItem.



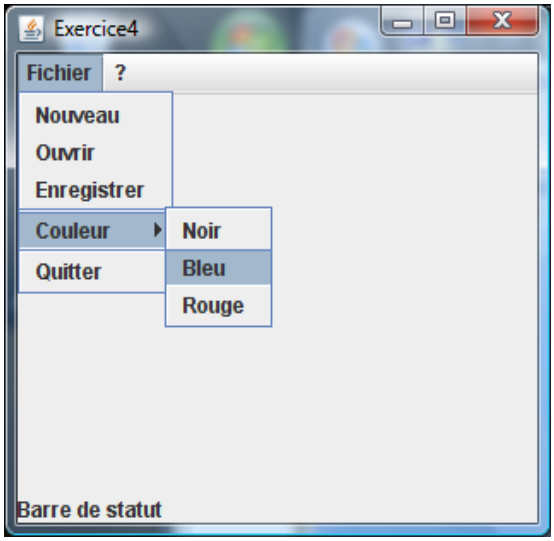
Il est important de noter que ce schéma exprime qu'un JMenu est un JMenuItem : lorsqu'un menu est employé en tant qu'item, il a le comportement d'un sous-menu.

Exemple :

```
private void creerMenu() {
    JMenuBar menuBar = new JMenuBar() ;
    JMenu menu1 = new JMenu("Fichier");
    JMenu menu2 = new JMenu("?");
    JMenu submenu = new JMenu("Couleur");

    menu1.add(new JMenuItem("Nouveau"));
    menu1.add(new JMenuItem("Ouvrir"));
    menu1.add(new JMenuItem("Enregistrer"));
    menu1.add(new JSeparator());
    menu1.add(submenu);
    menu1.add(new JSeparator());
    menu1.add(new JMenuItem("Quitter"));
    menu2.add(new JMenuItem("A propos"));

    submenu.add(new JMenuItem("Noir"));
    submenu.add(new JMenuItem("Bleu"));
    submenu.add(new JMenuItem("Rouge"));
    menuBar.add(menu1);
    menuBar.add(menu2);
    this.setJMenuBar(menuBar);
  }
```



Trouvez comment réagir à la sélection d'un item du menu, puis modifiez l'exemple précédent pour qu'un label en bas de la fenêtre affiche le dernier item sélectionné. De plus, l'item « quitter » doit provoquer la fin du programme. Vous écrirez une unique classe externe

Exercice 3 – Les Boites de dialogues

Un certain nombre de *boîtes de dialogue simples* sont utilisées très fréquemment dans toutes sortes d'applications : les boîtes d'information (figure 2), les « questions » à deux et trois boutons (figure 3), la saisie d'une chaîne, soit libre (figure 4), soit en la choisissant dans une liste de possibilités (figure 5).

Bien entendu, pour des boîtes de dialogue plus complexes il n'y a pas de solution toute prête, et le programmeur doit construire sa propre boîte dialogue. Cela se fait en définissant une sous-classe de **JDialog**.

En Java les boîtes de dialogue simples s'obtiennent très facilement à l'aide des méthodes **statiques show*XXX*Dialog** de la classe **JOptionPane**.

Dans cet exercice, nous allons écrire une application (voyez la figure 1) qui est une sorte de « présentoir » des diverses sortes de boîtes de dialogue qu'on peut obtenir de cette manière.

Cela se présente comme un cadre contenant une colonne de boutons. La pression sur un bouton affiche la boîte de dialogue en question. Lorsque la chose est pertinente (quand vous posez une question à l'utilisateur), l'information obtenue à travers la boîte de dialogue peut être réalisée en réalisant la création de la boîte de dialogue directement dans un System.out.println. (System.out.println(code d'appel))

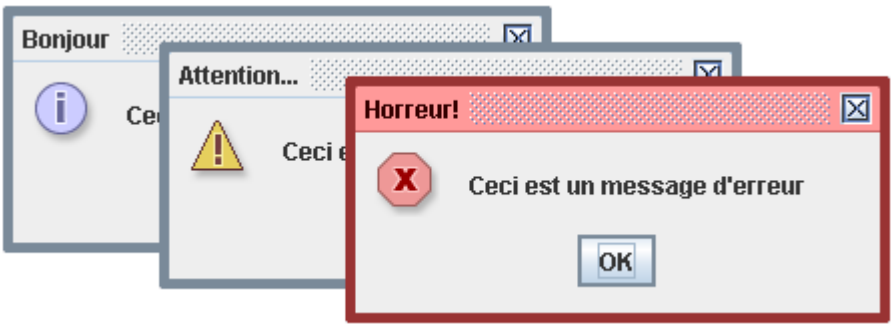


*Fig. 2 Ces boîtes sont obtenues grâce à la méthode **showMessageDialog** avec respectivement les types de messages : JOptionPane.INFORMATION_MESSAGE, JOptionPane.WARNING_MESSAGE et JOptionPane.ERROR_MESSAGE*

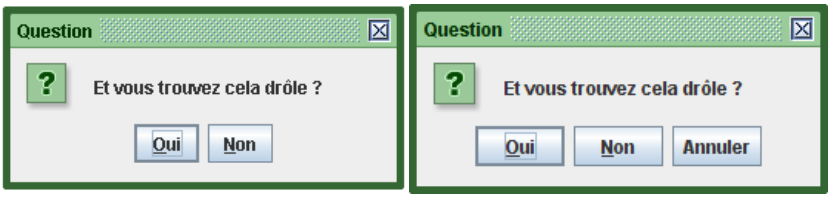


Fig. 3

*Cette boîte est obtenue grâce à la méthode **showConfirmDialog** avec l'option : JOptionPane.YES_NO_OPTION ou JOptionPane.YES_NO_CANCEL_OPTION.*

Ces boites de dialogues retournent la valeur 0 sir oui est cliqué, 1 si non est cliqué et 2 si annuler est cliqué

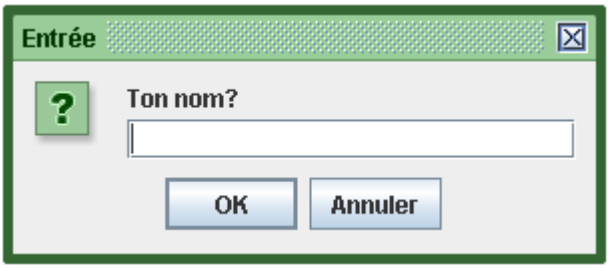


*Fig. 4 Cette boîte est obtenue grâce à la méthode **showInputDialog** avec l'option : JOptionPane.QUESTION_MESSAGE*

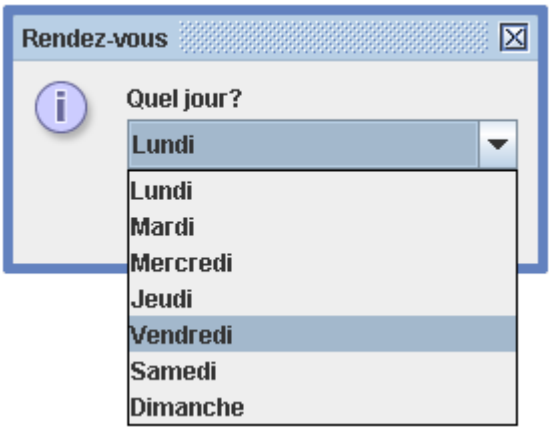


*Fig. 5 Cette boîte est obtenue grâce à la méthode **showInputDialog** avec l'option : JOptionPane.INFORMATION_MESSAGE et avec un tableau Object[] contenant une liste de valeur en 6^{ième} paramètre.*

Exercice 4 – Un jeu de questions

Réalisez l'interface graphique suivante qui permet de répondre à des questions. Les boutons permettant de répondre sont des boutons radio et ne doivent permettre de ne sélectionner qu'une réponse par question.

Le clic sur le bouton envoyer doit vérifier les réponses de l'utilisateur et afficher dans un JTextField blanc le résultat.