Comprendre les principes de Swing

L'objectif de ces exercices est de comprendre les bases de Swing sur un exemple fil rouge. Ils sont à faire sur machine ! Le point de départ est la classe `ComprendreSwing`.

Exercice 1 : Les constituants

Commençons par comprendre cette application.

1.1. Lire le texte de la classe `ComprendreSwing` et expliquer les éléments qui la composent, en particulier (par ordre d'apparition dans le texte) :

1. `JPanel`
2. `JLabel`
3. `JTextField`
4. `JButton`
5. `setLayout`
6. `FlowLayout`
7. `addActionListener`
8. `addMouseListener`
9. `ActionListener`
10. `actionPerformed`
11. `ActionEvent`
12. `MouseAdapter`
13. `mouseCliked`, `mouseEntered` et `mouseExited`
14. `MouseEvent`
15. `JFrame`
16. `getContentPane`
17. `add`
18. `pack`
19. `setDefaultCloseOperation`
20. `JFrame.EXIT_ON_CLOSE`
21. `setLocation`
22. `setVisible`
23. `EventQueue`
24. `invokeLater`

1.2. Cette application crée deux fenêtres identiques (au titre près). Dessiner l'apparence que devrait avoir ces fenêtres.

Exercice 2 : Exécuter l'application
2.1. Exécuter `ComprendreSwing` pour vérifier si l'apparence des fenêtres est celle attendue.
2.2. Jouer avec cette application. Ne pas hésiter à déplacer et redimensionner les fenêtres.
2.3. Le programme principal s'est terminé. Pourquoi l'application s'exécute-t-elle toujours ?

Exercice 3 : Le placement des composantes graphiques
On envisage plusieurs autres aspects pour la présentation de cette application. Ici, on s'intéresse à l'agencement des composants et pas à la barre haute de la fenêtre. Celle-ci dépend du système de gestion de fenêtres du système d'exploitation (Ubuntu/GNOME, Windows, MacOS, etc.).
3.1. Modifier l'apparence de l'application pour qu'elle corresponde à celle de la figure 1.

FIGURE 1 – Nouvelle apparence souhaitée pour l'application

3.2. Modifier l'apparence de l'application pour qu'elle corresponde à celle de la figure 2.

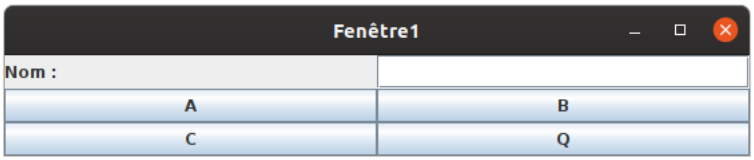

FIGURE 2 – Nouvelle apparence souhaitée pour l'application

3.3. Modifier l'apparence de l'application pour qu'elle corresponde à celle de la figure 3.

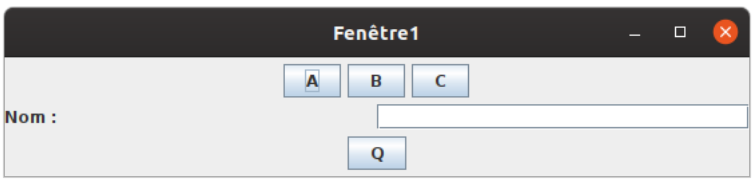

FIGURE 3 – Nouvelle apparence souhaitée pour l'application

3.4. Modifier l'apparence de l'application pour qu'elle corresponde à celle de la figure 4.

Exercice 4 : Aligner le texte des JLabel
Le texte « Nom : » est aligné à gauche. Ce n'est pas très esthétique dans les dernières apparences envisagées. Modifier l'application pour qu'il soit aligné à droite. On pourra utiliser la méthode `setHorizontalAlignment` de `JLabel`.

Exercice 5 : Fermer une fenêtre
L'application crée plusieurs fenêtres. Regardons ce qui se passe quand on ferme une fenêtre.

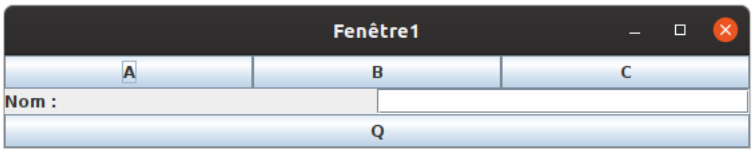

FIGURE 4 – Nouvelle apparence souhaitée pour l'application

5.1. Que se passe-t-il si on ferme une fenêtre (par exemple en cliquant sur la croix en haut à droite sous Ubuntu/GNOME) ?
5.2. Qu'est ce qui provoque ce comportement ?
5.3. Terminer l'application quand on ferme l'une de ses fenêtre n'est généralement pas souhaitable. Comment ne fermer que la fenêtre concernée et laisser l'application s'exécuter ? On pourra utiliser `DISPOSE_ON_CLOSE` plutôt que `EXIT_ON_CLOSE`.

Exercice 6 : Le bouton « Q »
Considérons le bouton Q et l'effet qu'il a.
6.1. Que se passe-t-il quand on clique sur le bouton « Q » ?
6.2. Expliquer les mécanismes de Swing qui sont mis en œuvre.
6.3. Faire que l'application s'arrête quand on clique sur « Q ». On utilisera `System.exit()`.

Exercice 7 : Le bouton « C »
Faire que le texte de la zone de saisie (`JTextField`) soit effacé quand on clique sur le bouton « C ». On pourra utiliser la méthode `setText` de la classe `JTextField`.

Exercice 8 : Listener et Adapter
Les gestionnaires d'événements de Swing (Listener) proposent souvent des « adaptateurs ». C'est par exemple le cas de `MouseListener` avec `MouseAdapter`.
8.1. Commençons par répondre à quelques questions rapides.

1. Combien y a-t-il de méthodes dans l'interface `MouseListener` ?
2. Combien y a-t-il de méthodes abstraites dans la classe `MouseAdapter` ?
3. Combien de méthodes sont définies dans la classe `ActionTrace` ?

8.2. Pourquoi les « adaptateurs » sont-ils des classes abstraites ?
8.3. Quel est l'intérêt des « adaptateurs » ?
8.4. Quel est le risque des « adaptateurs » ?
8.5. Qu'est ce qui devrait se passer quand on clique sur un bouton A ou B ? Que se passe-t-il en réalité ? Pourquoi ? Comment éviter ce type de problème ?
8.6. Ajouter les @Override qu'il est conseillé de mettre systématiquement quand on définit ou redéfinit une méthode. Conclusion ?

Exercice 9 : Le bouton Q (suite)
Faire en sorte que le bouton Q ferme la fenêtre mais pas l'application. Ceci signifie qu'il ne faut plus utiliser `System.exit` mais appeler la méthode `dispose()` de la fenêtre (`JFrame`).