Université de Bouira
Département d'Informatique
Module : Les Interactions Homme Machine
Niveau 3 info

Série TP 3

JFrame, Panes, et Layout Managers

Etapes à suivre

I. Top-Level Containers - JFrame

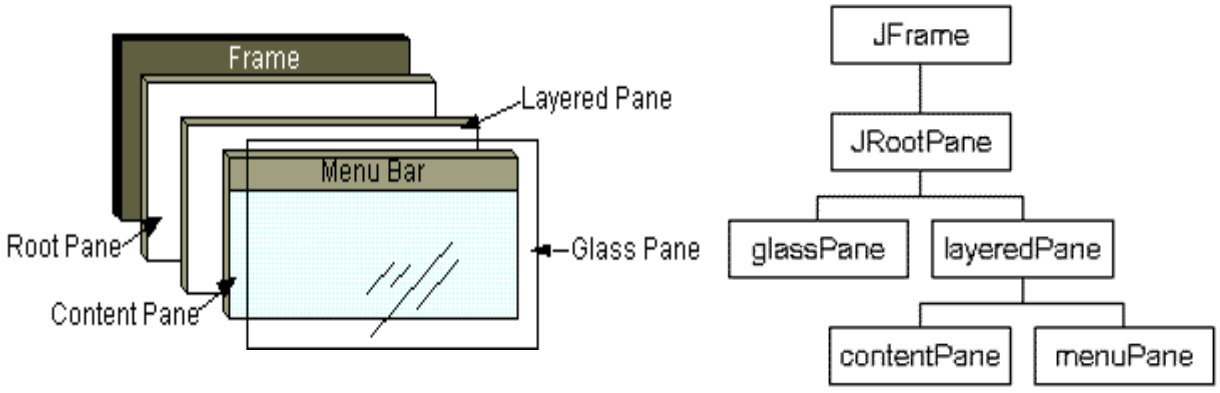
- Créer un nouveau projet Java sous le nom *IHMTP3*.
- Créer dans celui-ci une classe *MyJFrame* héritant de JFrame (extends JFrame).
- Pour configurer l'état initial de la fenêtre créée, ajouter une méthode *initJFrame* et appeler celle-ci depuis un constructeur. Utiliser les méthodes suivantes pour l'initialisation :
  - ✓ setTitle(String title)
  - ✓ setSize(int width, int height)
  - ✓ setLocationRelativeTo(null)
  - ✓ setResizable(booblean resizable)
  - ✓ setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
- Ajouter une classe main nommée *TestJFrame*. Créer dans cette dernière une instance de la classe *MyJFrame* qui devrait s'exécuter dans l'Event Dispatch Thread (en utilisant SwingUtilities).
- Pour afficher cette instance, utiliser la méthode *setVisible*(true).

II. Les containers par défault de JFrame

Tous les composants associés à un objet JFrame sont gérés par un objet de la classe JRootPane. Un objet JRootPane contient plusieurs Panes. Tous les composants ajoutés au JFrame doivent être ajoutés à un des Pane du JRootPane et non au JFrame directement. C'est aussi à un de ces Panes qu'il faut associer un layout manager si nécessaire pour manipuler le positionnement de chaque composant.

Le JRootPane se compose de plusieurs éléments :

- glassPane.
- layeredPane qui se compose du contentPane et du menuBar.



Exemples d'utilisation de RootPane

- Depuis la méthode *initJFrame*, récupérer le *RootPane* de la fenêtre et ajouter à cette dernière une bordure. Utiliser la classe *BorderFactory* pour créer un objet standard de traits de type *MatteBorder* de couleur noire.

```
this.getRootPane().setBorder(BorderFactory.createMatteBorder(3,
3, 3, 3, Color.BLACK));
```

- Depuis la méthode *initJFrame*, récupérer le RootPane de la fenêtre et appliquer un style de décoration de fenêtre en utilisant la méthode *setWindowDecorationStyle* comme suit :

```
this.setUndecorated(true) ;
this.getRootPane( ).setWindowDecorationStyel(JRootPane.FRAME) ;
```

- Afficher votre fenêtre.

Exemples d'utilisation de ContentPane

- Dans la méthode *initFrame,* créer un composant **JLabel (**JComponent d'affichage de texte). L'appeler *label1*.
- Récupérer le *ContentPane* de la JFrame et ajouter *label1* avec la méthode *add*. Afficher votre fenêtre et remarquer la position du JLabel dans la JFrame.

```
JLabel label1 = new JLabel("Label 1");
this.getContentPane().add(label1);
```

- Créer un deuxième composant JLabel, label2, et l'ajouter au ContentPane de la fenêtre.
- Afficher de nouveau votre fenêtre. Remarquer que label1 n'est plus présent dans la fenêtre. Ou plus exactement, est caché par label2 car ils partagent la même position dans le ContentPane.

```
JLabel label2 = new JLabel("Label 2");
this.getContentPane().add(label2);
```

Afin de régler ce problème, Java Swing permet de positionner les composants à l'intérieur d'un container de deux manières différentes : soit par le positionnement absolu, soit par l'utilisation d'un gestionnaire de positionnement = **Layout Manager**. La deuxième méthode est la plus souple et pratique.

Il existe de nombreux gestionnaires de layout dans Swing, en voici quelques uns [1]:

- FlowLayout : Il place les composants sur une ligne et recommence une nouvelle ligne à chaque fois qu'il atteint la fin de la ligne.
- BorderLayout : C'est le layout par défaut des ContentPane. Ce layout place les composants dans 5 zones du container : La zone du haut, la zone du bas, la zone de gauche, celle de droite et la zone du centre.

[1] http://baptiste-wicht.developpez.com/tutoriels/java/swing/debutant/?page=afficher

- CardLayout : Ce layout place les composants sur des couches disposées les unes sur les autres. On ne peut voir qu'une couche à la fois. On utilise surtout ce layout quand on a une série de composants qui s'affichent en fonction de quelques choses (liste déroulante, boutons, ...).
- GridLayout : Ce composant place les composants dans une grille. Il va redimensionner les composants pour les rendre tous de la même taille. C'est surtout utile quand on a plusieurs boutons ou champs texte en colonne et ligne qui doivent avoir la même taille, par exemple, les boutons d'une calculatrice.

- Pour appliquer un layout manager au ContentPane précédent, utiliser la méthode *setLayout* et créer un objet ***FlowLayout***. Choisir la position FlowLayout.LEFT.
- Afficher la fenêtre. Les deux JLabel devraient être alignés à gauche, l'un à coté de l'autre.

```
this.getContentPane().setLayout(new FlowLayout(FlowLayout.LEFT));
```

- Pour changer l'escape entre les deux JLabel, ajouter les valeurs hgap et vgap en paramètres de l'objet FlowLayout créé.

```
this.getContentPane().setLayout(new FlowLayout(FlowLayout.LEFT,
20, 20));
```

- Changer le gestionnaire de positionnement FlowLayout en ***BorderLayout***.
- Indiquer comme deuxième paramètre de la méthode *add* la position BorderLayout.NORTH pour *label1* et BorderLayout.SOUTH pour *label2*.

```
this.getContentPane().setLayout(new BorderLayout());
JLabel label1 = new JLabel("Label 1");
this.getContentPane().add(label1, BorderLayout.NORTH);
JLabel label2 = new JLabel("Label 2");
this.getContentPane().add(label2, BorderLayout.SOUTH);
```

Quelques actions sur les composants JLabel

- Ré appliquer le gestionnaire de positionnement FlowLayout pour le ContentPane de la fenêtre.
- Créer un troisième JLabel label3, puis utiliser les méthodes suivantes pour centrer le texte du JLabel, changer sa dimension, changer la couleur du background, et ajouter une bordure, respectivement. Afficher votre fenêtre.

```
this.getContentPane().setLayout(new FlowLayout(FlowLayout.LEFT));
JLabel label3 = new JLabel("Label 3");
label3.setHorizontalAlignment(SwingConstants.CENTER);
label3.setPreferredSize(new Dimension(90, 40));
label3.setBackground(Color.cyan);
label3.setOpaque(true);
label3.setBorder(BorderFactory.createMatteBorder(2, 2, 2, 2,
Color.black));
this.getContentPane().add(label3);
```

- Modifier le gestionnaire de positionnement en **GridLayout.** Indiquer comme paramètres, 2 pour le nombre de lignes, 3 pour le nombre de colonnes, et 5 pour le hgap et vgap.

```
this.getContentPane( ).setLayout(new GridLayout(2, 3, 5, 5));
```

- Créer 3 autres JLabel avec la même configuration que label3 et les ajouter au ContentPane.
- Afficher la nouvelle fenêtre et remarquer le changement de positionnement en grille des JLabel.

Pour ranger les composants du ContentPane à la suite soit sur ligne, soit sur une colonne, le layout **BoxLayout** peut être utilisé. Son constructeur prend deux paramètres : un container et un axis (ex : X_AXIS, Y_AXIS, PAGE_AXIS).

- Définir BoxLayout comme gestionnaire de positionnement du ContentPane comme suit :

```
BoxLayout boxLayout = new BoxLayout(getContentPane(),
                                                BoxLayout.Y_AXIS);
this.getContentPane( ).setLayout(boxLayout);
```

Pour ajouter de l'espace entre les différents JLabel, plusieurs méthodes sont possibles. Parmi-elles, la création d'une composant invisible RigidArea (Box class) entre chaque composant :

```
this.getContentPane( ).add(label1);
this.getContentPane( ).add(Box.createRigidArea(new Dimension(0,
5));
this.getContentPane( ).add(label2);
```

Demander du texte à l'utilisateur

Java Swing offre plusieurs types de composants permettant à l'utilisateur d'entrer un texte :

- Le **JTextField** : C'est le plus simple des composants textes. Il permet d'entrer un texte sur une seule ligne.
- Le **JTextArea** : Il permet d'entrer un texte complet sur plusieurs lignes.
- Le **JEditorPane** : Très complet, vous pouvez modifier la police, la taille, la couleur de votre texte. Il permet même d'afficher des pages html. Vous pouvez y insérer du texte et des images.
- Le **JTextPane** : C'est le composant texte le plus évolué de Swing. Il permet la même chose que le JEditorPane mais il permet également d'insérer des composant personnalisés, c'est-àire que vous pouvez y insérer des composants Swing.

Exemple d'utilisation d'un JTextField

- Remettre le positionnement de ContentPane à FlowLayout(FlowLayout.LEFT).
- Supprimer tous les JLabel précédemment créés.
- Créer un objet de type JTextField et l'appeler *textField*.
- En utilisant la méthode *setColumns,* donner le nombre de colonne à afficher pour ce JTextField.

- Ajouter *textField* au *ContentPane*, puis afficher la fenêtre.
- Pour mettre un texte par défaut (Ex : Enter your email…) dans le JTextField, passer une chaine de caractères au constructeur ou utiliser la méthode *setText*.
- En utilisant la méthode *setBorder* et la classe *BorderFactory*, ajouter une bordure de type TitleBorder au textField via la méthode *createTitleBorder*("Email").
- Utiliser la méthode setCaretColor pour changer la couleur du curseur dans le textField.
- Utiliser la méthode setHorizontalAlignment(JTextField.CENTER) pour centrer le texte.
- Afficher votre nouvelle fenêtre.

```
JTextField textField = new JTextField("Enter your email ...");
textField.setColumns(20);
textField.setBorder(BorderFactory.createTitledBorder("Email"));
textField.setCaretColor(Color.red);
textField.setHorizontalAlignment(JTextField.CENTER);
this.getContentPane().add(textField);
```

*Autre variante du JTextField est le **JPasswordField**.*

- Refaire les mêmes étapes ci-dessus pour créer et configurer un JPasswordField. L'appeler *passwordField*.
- Ajouter un Tooltip Text au *passwordField* en utilisant la méthode *setTooltipText*(String text). Ex text: "Please, enter your password here."
- Afficher votre nouvelle fenêtre.

```
JPasswordField passwordField = new JPasswordField("Enter your
password ...");
passwordField.setColumns(20);
passwordField.setToolTipText("Enter your Password");
passwordField.setBorder(BorderFactory.createTitledBorder("Passwo
rd"));
this.getContentPane().add(passwordField);
```

- Le code source complet sur **GitHub** :

https://github.com/GitTeaching/MesTPIHM