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

Série TP 8 (Suite Série 7)

Autres Composants Swing - JRadioButton, JSlider, JComboBox, JProgressBar, JList, JScrollPane, JTabbedPane, JTextArea, et JTextPane.

Etapes à suivre

I. Top-Level Containers - JFrame

- Créer un nouveau projet Java sous le nom *IHMTP4*.
- 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. JSlider

Ce composant permet d'utiliser un système de mesure pour une application : redimensionner une image, choisir le volume d'un morceau de musique, l'opacité d'une couleur, etc. En déplaçant son curseur, la méthode *stateChanged( )* de l'écouteur **ChangeListener** est appelée.

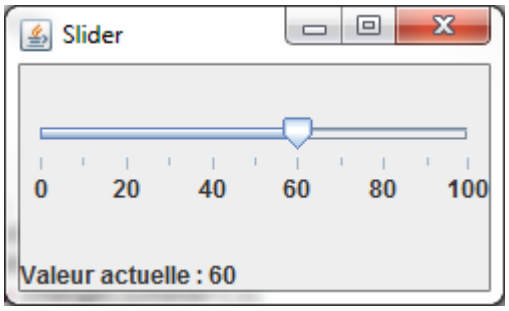
- Créer un **JPanel**, *panel*, et l'appliquer comme ContentPane de la fenêtre.
- Créer un objet-composant de type **JSlider** appelé *slider* en indiquant sa valeur minimum, sa valeur maximum et sa valeur courante comme paramètres respectivement.
- Un JSlider peut éventuellement afficher et personnaliser des marques (Ticks) pour la plage de ses valeurs grâce aux méthodes setMinorTickSpacing ( ), setMajorTickSpacing (), setPaintTicks ( ), et setPaintLabels( ) comme suit :

```
JSlider slider = new JSlider(0, 100, 30);

slider.setMinorTickSpacing(10);

slider.setMajorTickSpacing(20);

slider.setPaintTicks(true);

slider.setPaintLabels(true);
```

- Ajouter un **JLabel** nommé *valeurSlider* permettant d'afficher la valeur actuelle de *slider* et de le mettre à jour à chaque nouveau déplacement du curseur. Utiliser l'écouter ChangeListener du JSlider.

```
JLabel valeurSlider=new JLabel("Valeur Actuelle : ");
this.add(label);

slider.addChangeListener(new ChangeListener(){
     @Override
     public void stateChanged(ChangeEvent e) {
          int value = slider.getValue();
          valeurSlider.setText(Integer.toString(value));
     }
});
```

- Ajouter *slider* à *panel* puis afficher votre fenêtre.



III. JProgressBar

Ce composant permet de réaliser une barre de progression pour des traitements longs. Il peut être horizontal ou vertical.

- Ajouter à *panel* un composant **JProgressBar** nommé *progBar*.
- Indiquer sa valeur minimum et sa valeur maximum en utilisant les méthodes setMinimun(0) et setMaximum(100) respectivement.
- Utiliser la méthode setStringPainted( ) pour déterminer si la barre de progression affiche le pourcentage de la tâche accomplie.

```
JProgressBar progBar = new JProgressBar();
progBar.setMinimum(0);
progBar.setMaximum(100);
progBar.setStringPainted(true);
panel.add(progBar);
```

- Ajouter à *panel* un composant **JButton** appelé *startBtn*. L'initialiser à "Start".
- Ajouter à *startBtn* un écouteur ActionListener. Utiliser une classe interne *ClickAction* pour implémenter la méthode *actionPerformed*.

```
JButton startBtn = new JButton("Start");
startBtn.addActionListener(new ClickAction());
this.add(startBtn);
```

- Ajouter à *panel* un composant **Timer** comme suit :

```
Timer timer = new Timer(50, new UpdateBarListener());
```

Un **Timer** est une sorte d'horloge générant des tics de manière régulière, auquel un programme réagit pour effectuer une tâche répétitive.
Pour éviter d'utiliser directement un thread et simplifier la programmation d'un timer, Swing propose la classe javax.swing.Timer.

Cette classe rappelle la méthode *actionPerformed* ( ) d'un (ou plusieurs) listener à chaque fois qu'un laps de temps donné s'est écoulé.

Cet exemple utilise la classe Timer pour faire incrémenter la valeur de la barre de progression jusqu'à sa valeur maximale en lançant la classe interne *UpdateBarListener* chaque 50ms. Si la valeur maximum est atteinte, le timer est arrêté et le texte de startBtn devient "End". Cette dernière implémente ActionListener comme suit :

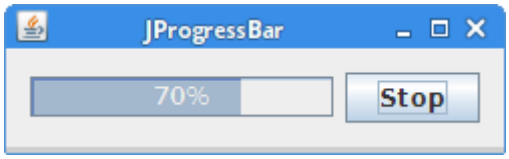
```
private class UpdateBarListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            int val = progBar.getValue();

            if (val >= 100) {
                timer.stop();
                startBtn.setText("End");
            }
            else
                progBar.setValue(++val);
        }
    }
```

- Modifier la méthode ActionPerformed de la classe interne *ClickAction* pour que le bouton startBtn démarre ou arrête le timer comme suit :

```
private class ClickAction extends AbstractAction {

    @Override
    public void actionPerformed(ActionEvent e) {

        if (timer.isRunning()) {

            timer.stop();
            startBtn.setText("Start");

        } else if (!"End".equals(startBtn.getText())) {

            timer.start();
            startBtn.setText("Stop");
        }
    }
}
```

Le texte du bouton *startBtn* est mis à jour dynamiquement; Il peut avoir "Start", "Stop", ou "End" valeurs de chaîne.

- Afficher la nouvelle fenêtre et tester.



IV. JTabbedPane

Le composant JTabbedPane vous permet d'obtenir une interface composée d'autant d'onglets que vous le désirez et gérable de façon dynamique.

- Réinitialiser la JFrame à l'étape 1.
- Créer un composant **JTabbedPane** nommé *jtp*. Utiliser la méthode *setPreferredSize*( ) pour initialiser sa dimension à (350, 250).
- Ajouter à jtp **trois onglets** en utilisant la méthode *addTab* comme suit :

```
JTabbedPane jtp = new JTabbedPane();
jtp.setPreferredSize(new Dimension(350, 250));
jtp.addTab("JLabel", new LabelPanel());
jtp.addTab("JTextArea", new TextAreaPanel());
jtp.addTab("JTextPane", new TextPanePanel());
this.getContentPane().add(jtp);
```

Chaque onglet de *jtp* a son propre **JPanel** représenté par les classes *LabelPanel*, *TextAreaPanel*, et *TextPanePanel*. Ces trois classes sont des classes internes qui étendent JPanel.

- Déclarer les trois classes à l'intérieur de *MyJFrame*. Exemple LabelPanel :

```
private class LabelPanel extends JPanel {
  public LabelPanel() {
     //TODO
  }
}
```

- Créer et ajouter un **JLabel** *label* dans le constructeur *LabelPanel (à la place de //TODO)*. Changer son Font à ("Serif", Font.ITALIC, 18).

- Créer et ajouter un composant **JTextArea** *textArea* dans un constructeur *TextAreaPanel*. Ajouter comme paramètres le nombre de lignes et de colonnes (10,30).
- Utiliser la méthode *setLineWrap(true)* pour envelopper les lignes lorsqu'elles sont trop longues pour s'adapter à la largeur de zone de texte.
- Utiliser la méthode *setWrapStyleWord(true)* pour spécifier comment est la ligne va être enveloppé. Dans ce cas, les lignes seront enveloppées aux limites des mots - espaces blancs.
- Ajouter une bordure de type EmptyBorder(8,8,8,8) à *textArea.*

- Pour rendre le texte de *textArea* scrollable, mettre ce dernier dans un composant **JScrollPane**.

```
JTextArea textArea = new JTextArea("Type some text ..", 10, 30);
textArea.setLineWrap(true);
textArea.setWrapStyleWord(true);
textArea.setBorder(BorderFactory.createEmptyBorder(8, 8, 8, 8));
this.add(textArea);

JScrollPane scroll = new JScrollPane(textArea);
this.add(scroll);
```

V. JTextPane

Le composant **JTextPane** est un composant plus avancé pour travailler avec du texte. Le composant peut effectuer des opérations de mise en forme complexes sur le texte. Il peut également afficher des documents HTML.

- Appliquer BorderLayout comme le layout manager du JPanel (i.e. this.setLayout( )).
- Créer et ajouter (en position NORTH) un composant **JTextPane** *textPane* dans un constructeur *TextPanePanel*.
- Utiliser la méthode *setContentType("text/html")* pour définir le contenu du composant *textPane* en tant que document HTML.
- Utiliser la méthode *setEditable(false)* désactiver la modification.
- Ajouter une bordure de type EmptyBorder à *textPane.*
- Pour rendre le texte de *textPane* scrollable, mettre ce dernier dans un composant **JScrollPane**.

```
this.setLayout(new BorderLayout());

JTextPane textPane = new JTextPane();
textPane.setContentType("text/html");
textPane.setEditable(false);
textPane.setBorder(BorderFactory.createEmptyBorder(8, 8, 8, 8));
this.add(textPane, BorderLayout.NORTH);

JScrollPane scroll = new JScrollPane(textPane);
this.add(scroll);
```

- Créer un **document HTML** *test.html* dans le package du projet avec le contenu suivant :

```
<!DOCTYPE html>
   <header></header>
   <body>
      <h2>Hello World !</h2>
      <h3 style=”color:blue”>It’s HTML!</h3>
       <br><br><br>
       <ul>
          <li>List Item 1</li>
          <li>List Item 2</li>
          <li>List Item 3</li>
      </ul>
   </body>
</html>
```

- Télécharger le document HTML créé dans le *textPane* comme suit :

```
try {
         String cd = System.getProperty("user.dir") + "\\src\\";
         textPane.setPage("File:///" + cd + "test.html");
    } catch (IOException e) {
         e.printStackTrace();
    }
```

<u>Le résultat</u> :