

Chapitre 7: interfaces graphiques & événements

1^{ÈRE} ANNÉE CYCLE INGÉNIEUR

M. NAJIB

Objectifs

L'objectif de ce chapitre est de vous initier à l'utilisation des interfaces graphiques SWING en JAVA.

Ce chapitre se focalise dans sa première partie sur le codage des interfaces graphiques en JAVA et de la compréhension du code SWING sans utilisation des outils drag&drop pour la composition d'une interface graphique

À la fin de cette première partie, vous êtes sensé acquérir les connaissances suivantes:

- Comprendre le rôle de JFrame en JAVA
- Maîtriser l'utilisation des Layouts
- Utilisation des panels
- Utilisation des composants d'une interface graphique

Introduction

- En JAVA il existe plusieurs API qui permettent la programmation des interfaces graphiques (AWT, SWING, JAVA FX)
- SWING est L'API de programmation des interfaces graphiques la plus utilisée en JAVA.



Classe JFrame

- Pour l'utilisation des composants graphiques de SWING il faut utiliser le package

`javax.swing.[nomComposant]`

- Pour la création de la première interface, nous allons utiliser la classe **Jframe** de la manière suivante:

```
JFrame jf = new JFrame();
```

Classe JFrame

La création de notre première fenêtre doit prendre en considération d'autres paramètres pour aboutir au résultat désiré.

- Les dimensions de la fenêtre : **fenêtre.setSize(largeur, hauteur)**
- Affichage de la fenêtre: **fenêtre.setVisible(True ou False)**
- Redimensionnement de la fenêtre: **fenêtre.setResizable(True ou False)**
- Le titre de la fenêtre: **fenêtre.setTitle(String-titre)**
- Comportement du programme à la fermeture: **fenêtre.setDefaultCloseOperation()**

Classe JFrame

Exemple de l'utilisation des fonctions les plus utilisées pour la création et la personnalisation de la première fenêtre:

```
import javax.swing.JFrame;
public class Fenetre1 {

    // execution
    public static void main(String args[]){
        JFrame f = new JFrame();
        f.setVisible(true);
        f.setSize(400, 200);
        f.setTitle("Ma premiere fenetre");
        f.setResizable(false);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

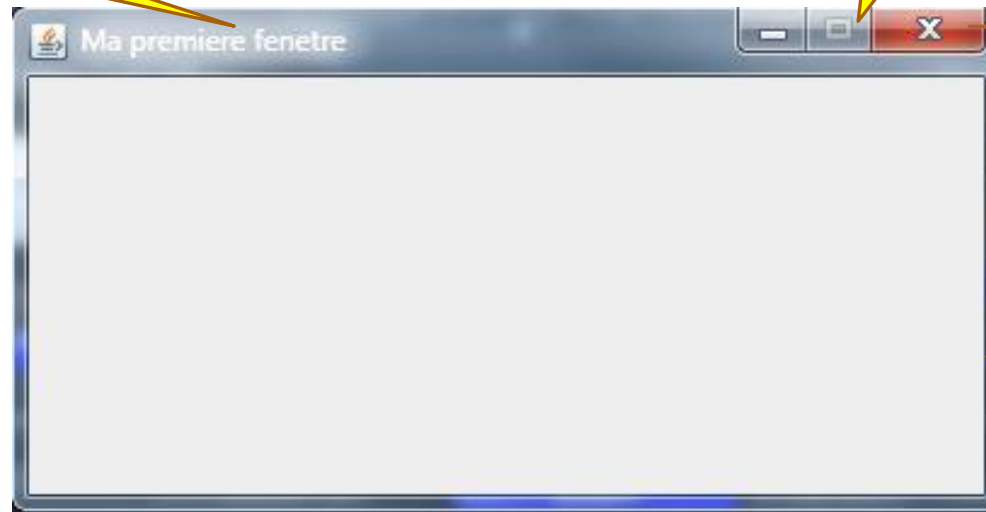
Classe JFrame

Résultat de la première fenêtre:

Titre de la
fenêtre

Bouton agrandir
désactivé

Testez le bouton
fermer ??



Fenêtre qui respecte les dimensions
spécifiées lors de la création

Classe fenêtre personnalisée

Pour créer une fenêtre personnalisée il faut étendre la classe JFrame

```
import javax.swing.JFrame;  
  
public class Fenetre3 extends JFrame {  
  
}
```

Classe fenêtre personnalisée

Exemple d'une classe personnalisée

```
import javax.swing.JFrame;
public class Fenetre3 extends JFrame {
    // constructeur
    public Fenetre3(){
        this.setVisible(true);
        this.setSize(400, 200);
        this.setTitle("Ma premiere fenetre");
        this.setResizable(false);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    // deuxième constructeur
    public Fenetre3(String titre){
        this.setVisible(true);
        this.setSize(400, 200);
        this.setTitle(titre);
        this.setResizable(false);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Constructeur de
base

Paramètres
seront appliqués
à l'instance de la
classe

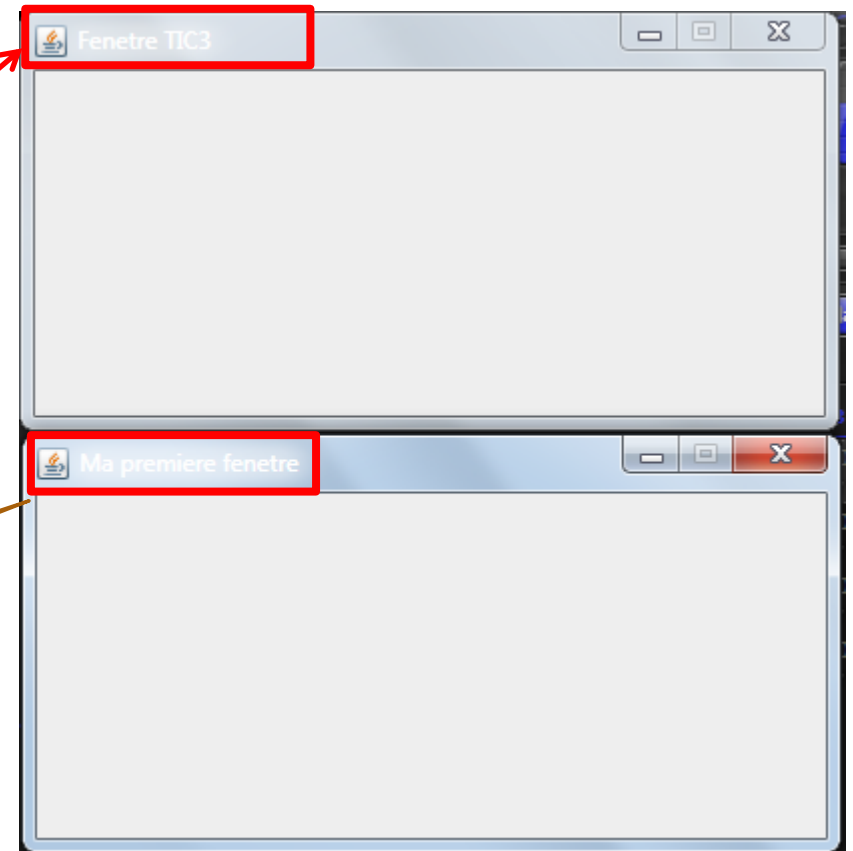
Constructeur
avec titre

Classe fenêtre personnalisée

Résultat de la création de l'utilisation des constructeurs

```
public class Main {  
    public static void main(String argv[]){  
        Fenetre3 f3 = new Fenetre3();  
        Fenetre3 f33 = new Fenetre3("Fenetre TIC3");  
    }  
}
```

Constructeur de
base



Classe JPanel

La classe JPanel permet de créer un panel qui sera utilisé pour contenir les composants de l'interface graphique

La création d'un panel se faire de la manière suivante:

```
import javax.swing.JPanel;  
  
JPanel nomPanel= new JPanel();
```

Classe JPanel

Exemple de la création d'un JPanel

```
public fenetre2(){  
    this.setTitle("Titre de la fenêtre");  
    this.setSize(600, 300);  
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    // les panels  
    JPanel jp = new JPanel();  
    jp.setBackground(Color.yellow);  
    this.setContentPane(jp);  
}
```

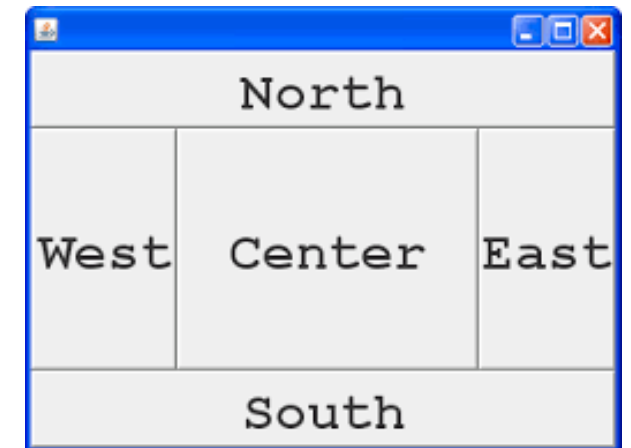
Création d'un
JPanel()

Spécifier le JPanel par
défaut à intégrer dans la

Layout en JAVA

Les layouts en JAVA


1. **Absolut**
2. BorderLayout
3. FlowLayout
4. **GrideLayout**



Voir la démonstration: implémentation et utilisation des Layouts

JTextField

La création des JTEXTFIELD en JAVA se fait de la manière suivante:



```
jtf1 = new JTextField();  
jtf1.setBounds(141, 38, 111, 20);  
panel.add(jtf1);  
jtf1.setColumns(10);
```

JLabel

Les JLabel en JAVA sont utilisés pour la création des labels pour afficher du texte au niveau des interfaces graphiques.

Le code suivant montre la création des

```
JLabel lblNewLabel = new JLabel("Login");  
lblNewLabel.setBounds(39, 41, 46, 14);  
panel.add(lblNewLabel);  
  
JLabel lblPrenom = new JLabel("Password");  
lblPrenom.setBounds(39, 75, 46, 14);  
panel.add(lblPrenom);
```

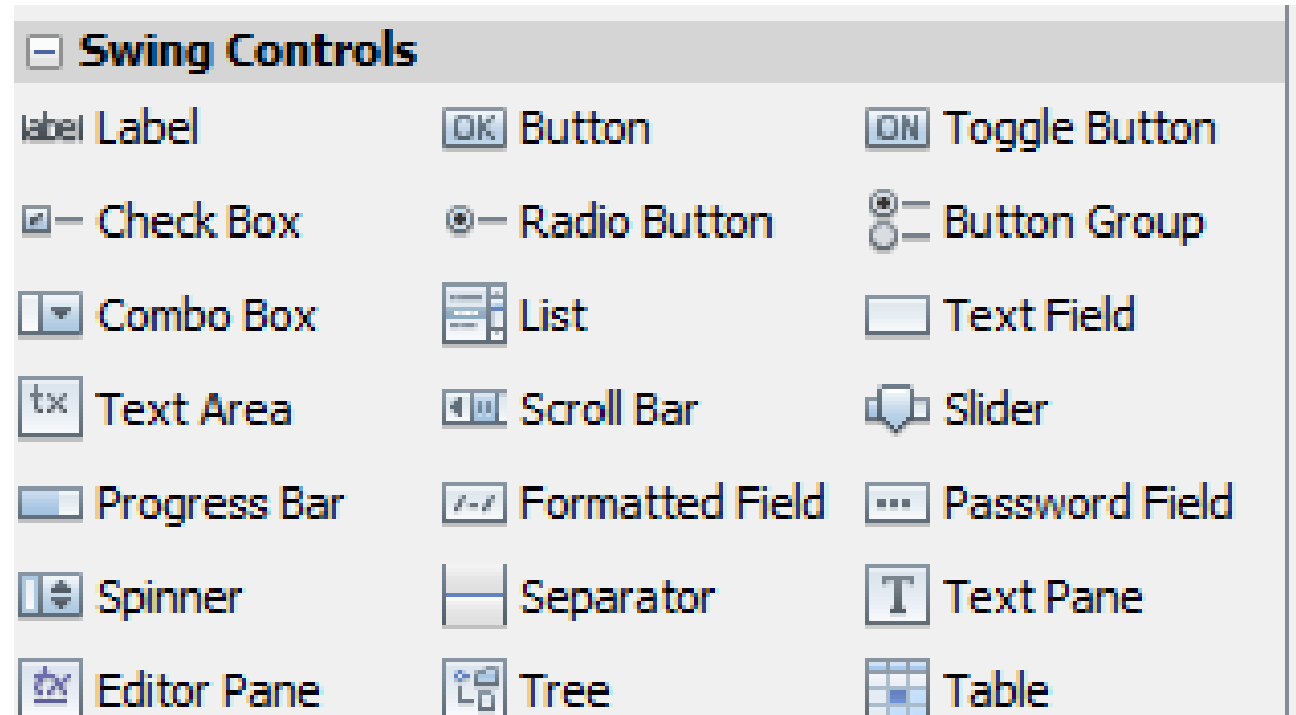
JButton

La création des boutons en JAVA se fait de la manière suivante:

```
JButton btnConnector = new JButton("Connector");  
btnConnector.setBounds(141, 113, 89, 23);  
panel.add(btnConnector);
```

Les composants de base

- JTextField
- JLabel
- JButton
- Spinner
- ButtonGroup
- Button
- TextArea
- Jtable
- etc



Partie 2: événements

Les événements

Les événements sont basé sur l'utilisation des listener pour détecter les événements générés par un utilisateur

On peut distinguer les événements

- Clique
- Survole
- Chargement d'une fenêtre
- Fermeture d'une fenêtre

Les événements

- Évènement clique
- Utilisation d'un **listener**
- Appel d'un traitement externe

```
jButton1.addActionListener(new java.awt.event.ActionListener() {  
  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton1ActionPerformed(evt);  
    }  
  
});
```

Exemple

Exemple d'un traitement simple
calculatrice

Caclulatrice

Nombre 1

Nombre 2

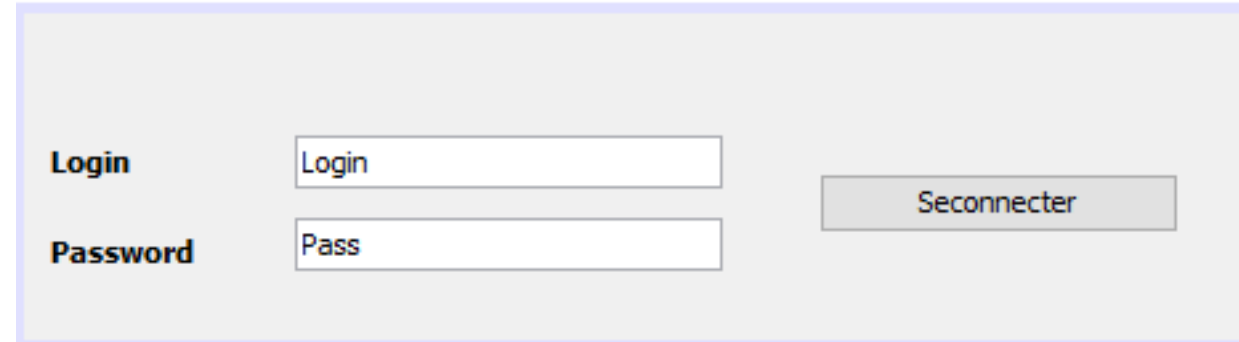
Résultat

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String nbr1, nbr2;  
  
    nbr1 = jTextField1.getText();  
    nbr2 = jTextField2.getText();  
  
    int somme = 0;  
    try{  
        somme = Integer.parseInt(nbr1) + Integer.parseInt(nbr2);  
        jTextField3.setText(""+somme);  
    }catch(Exception e){  
        System.err.println("probleme de saisie " );  
        jTextField3.setText("attention!!!");  
    }  
    jTextField3.setText(""+somme);  
}
```

Navigation entre fenêtres

La navigation entre les fenêtres:

Navigation simple par séparation de JFrame



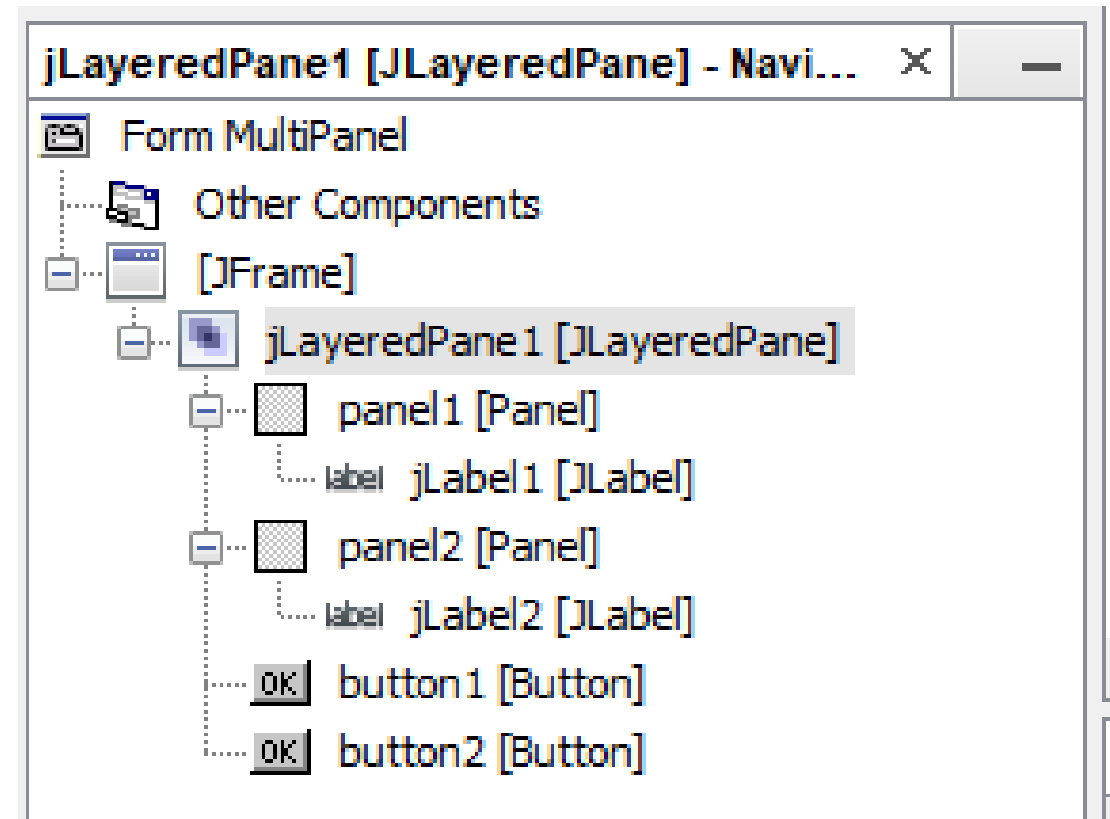
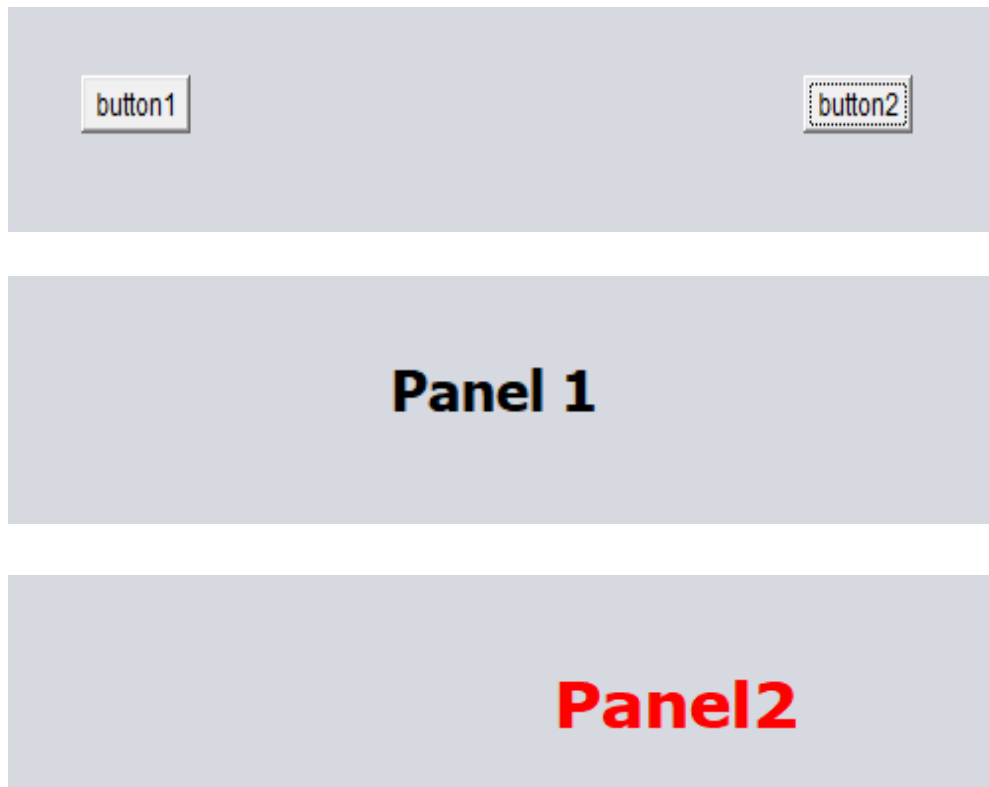
A login form with a light gray background. It contains two labels, 'Login' and 'Password', each followed by a text input field. The 'Login' field contains the text 'Login' and the 'Password' field contains the text 'Pass'. To the right of these fields is a button labeled 'Seconnecter'.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String login = jtfLogin.getText();  
    String password = jtfPass.getText();  
  
    if(login.equals("Admin") && password.equals("1234")){  
        Fen2 welcomeJF = new Fen2();  
        welcomeJF.setVisible(true);  
        this.setVisible(false);  
    }  
}
```

Hello Ing3

Navigation entre fenêtres (LayeredJPanel)

Utilisation du composant **LayeredJPanel**



Navigation entre fenêtres (LayeredJPanel)

Action pour afficher et masquer un panel/Jpanel est comme suite

```
private void button1ActionPerformed(java.awt.event.ActionEvent evt) {  
    panel1.setVisible(true);  
    panel2.setVisible(false);  
}
```

```
private void button2ActionPerformed(java.awt.event.ActionEvent evt) {  
    panel2.setVisible(true);  
    panel1.setVisible(false);  
}
```


Jtable

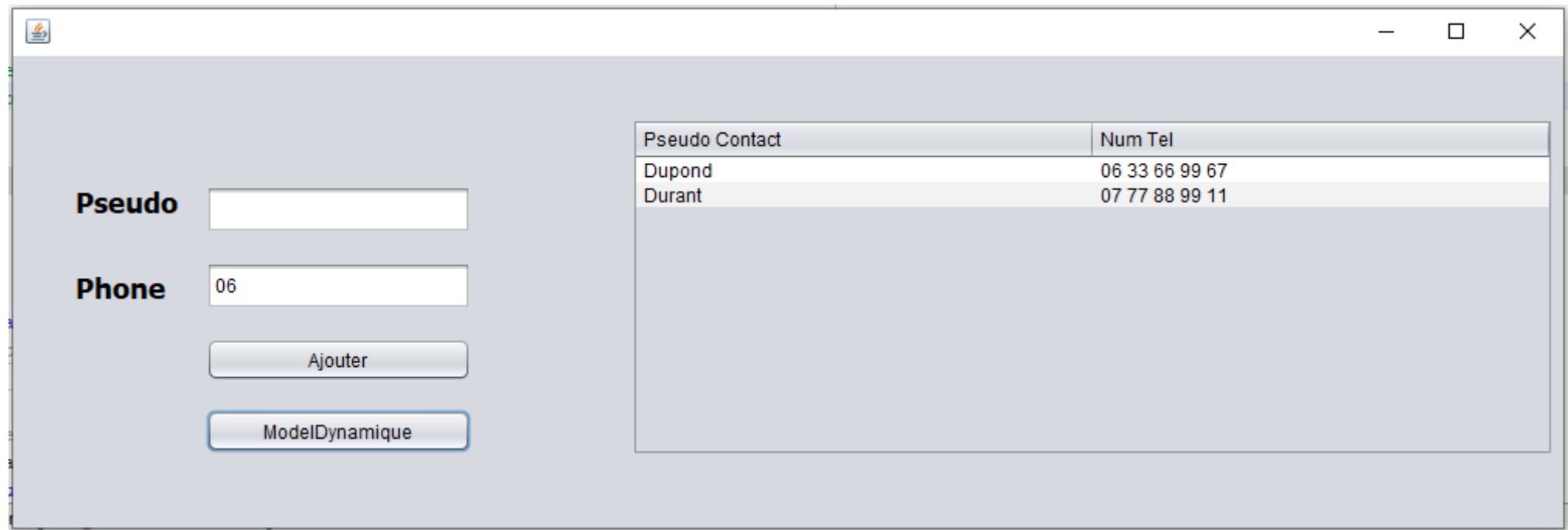
L'utilisation des Jtable nécessite la création d'un modèle de données pour formater définir le nombre et le type de colonne qui seront utilisées pour l'affichage des données :

- On peut définir ce modèle d'une manière statique (utilitaire)
- On peut utiliser un modèle dynamique créer à partir des données à afficher

JTable

Exemple d'application

➤ Remplissage dynamique d'une JTable à partir d'un formulaire



The screenshot shows a Java Swing application window with a light gray background. On the left side, there is a form with two labels: "Pseudo" and "Phone". The "Pseudo" label is next to an empty text input field. The "Phone" label is next to a text input field containing the value "06". Below these input fields are two buttons: "Ajouter" and "ModelDynamique". On the right side of the window, there is a JTable with two columns: "Pseudo Contact" and "Num Tel". The table contains two rows of data: "Dupond" with phone number "06 33 66 99 67" and "Durant" with phone number "07 77 88 99 11".

Pseudo Contact	Num Tel
Dupond	06 33 66 99 67
Durant	07 77 88 99 11

JTable

Exemple ajout d'une ligne dans le model de la Jtable (Model déjà créé)

```
private void jbtnAddActionPerformed(java.awt.event.ActionEvent evt) {  
    String pseudo = jtfPseudo.getText();  
    String phone = jtfPhone.getText();  
  
    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();  
    model.addRow(new Object[]{pseudo, phone});  
}
```

Exemple Jtable (ModelDynamique)

Exemple d'utilisation Les noms de colonnes

```
private void jbtnDynamiqueActionPerformed(java.awt.event.ActionEvent evt) {  
    LinkedList<Contact> l = new LinkedList<Contact>();  
    l.add(new Contact("Dupond", "06 33 66 99 67"));  
    l.add(new Contact("Durant", "07 77 88 99 11"));  
  
    // vecteur pour les noms de cols  
    Vector<String> VCols = new Vector<String> ();  
    VCols.add("Pseudo Contact");  
    VCols.add("Num Tel ");  
}
```

Exemple Jtable (ModelDynamique)

Création des lignes avec un vecteur de vecteur

```
private void jbtnDynamiqueActionPerformed(java.awt.event.ActionEvent evt) {  
  
    // vecteur à utiliser pour les ligne  
    Vector<Object> VRows = new Vector<Object> ();  
    // vecteur des vecteur ou Vecteur de ligne  
    Vector<Vector<Object>> VData = new Vector<Vector<Object>>();  
  
    for(Contact c : l){  
        VRows = new Vector<Object> ();  
        VRows.add(c.pseudo);  
        VRows.add(c.phone);  
        VData.add(VRows);  
    }  
}
```

Exemple Jtable (ModelDynamique)

Création du model de la Jtable avec les deux vecteurs

```
DefaultTableModel modelDyn = new DefaultTableModel(VData, VCols);  
jTable1.setModel(modelDyn);
```

Jtable événements (MouseClicked)

L'événement on **MouseClicked** permet de déclencher un traitement suite à la sélection d'une ligne de la Jtable;

```
private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {  
  
}
```

Jtable – événements

Récupération du contenu d'une Jtable

- `jTable1.getSelectedRow()` : récupération du numéro de la ligne sélectionnée
- `jTable1.getValueAt(ligne, 0)`: récupération du contenu d'une cellule

Modification d'une cellule de la JTable

- `jTable1.getModel().setValueAt(value, ligne, colonne);`

JTable (Form-Loading)

Exemple de mise à jour d'un formulaire suite à la sélection d'une ligne de la JTable

```
private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {  
    int ligne = jTable1.getSelectedRow();  
  
    String pseudo = jTable1.getValueAt(ligne, 0).toString();  
    String Phone = jTable1.getValueAt(ligne, 1).toString();  
  
    jtfPseudo.setText(pseudo);  
    jtfPhone.setText(Phone);  
}
```

Pseudo Contact	Num Tel
Dupond	06 33 66 99 67
Durant	07 77 88 99 11

Pseudo	<input type="text" value="Dupond"/>
Phone	<input type="text" value="06 33 66 99 67"/>

Merci de votre attention

