Module: Java Avancé
Organisation du cours

- Module de 21h : **7 semaines** x3h (1h30 cours & 1h30 TP)
- Introduit les concepts de base de la programmation IHM, en Java, avec l'API SWING ainsi que le lien avec les BD (JDBC)

Contenu des chapitres:

1. Bases d'IHM, Interface utilisateur: Présentation des principaux composants de base de l'API Swing (bouton, label, image, champ de saisie, etc…); Gestion et programmation événementielle;
2. Accès aux sources de données et lien avec les BD : JDBC

Modalités d'évaluation :

-Note de CC: note de TP (moyenne des TP) (35%)- Bonus/malus sur l'implication (présence / pertinence des interactions)

-Examen final en fin de la p1 sur les concepts vus en TPs (65%)

ECOLE SUPÉRIEURE DE LA STATISTIQUE
ET DE L'ANALYSE DE L'INFORMATION

Chapitre 1
Développement interfaces utilisateurs en Java (SWING)

AÏCHA EL GOLLI
aicha.elgolli@essai.ucar.tn

Septembre 2024

Objectifs du cours

- Comprendre le fonctionnement d'une interface graphique (Interface Homme Machine):
  - positionnement et rendu des composants
  - gestion des événements
- Maîtriser les composants les plus courants (fenêtres, boutons, listes, menus, arbres, etc)
- Être capable de concevoir et d'implémenter proprement une interface

Sommaire de ce chapitre

1. Introduction : les interfaces utilisateur (IHM, GUI)
2. Bases de Swing : **Composants, Conteneurs**
    1. Principe
    2. Les conteneurs de haut niveau
    3. Les conteneurs de niveau intermédiaire
    4. Les composants
    5. Les composants de menu
3. Placer les composants (layout) Gestionnaire de **dispositions**, mise en page :
    1. FlowLayout
    2. BorderLayout
    3. GridLayout
    4. GridBagLayout
4. Exemples
5. La Gestion Des **Événements**

Introduction :
les interfaces utilisateur (IHM, GUI)

Principes de base

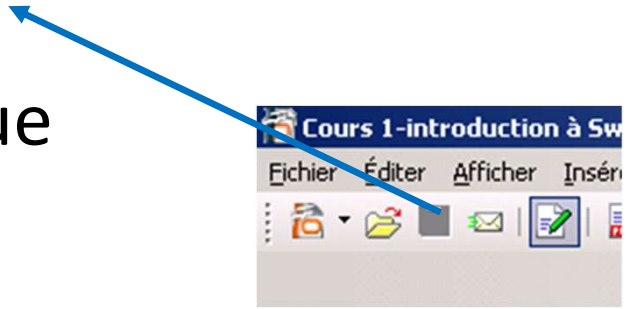
- Interface Console
  - C'est le programme qui pilote l'utilisateur, en le sollicitant quand nécessaire pour qu'il fournisse des données

    ==> Dialogue en mode **texte et séquentiel** dans une fenêtre appelée « Console »

- Interface graphique (GUI – Graphical User Interface)
  - L'utilisateur pilote le programme, qui réagit à ses demandes (sélection d'articles, d'item de menu, clic bouton,…)
  - Chaque action de l'user = événement
  - Programmation « événementielle »

Principe d'une GUI

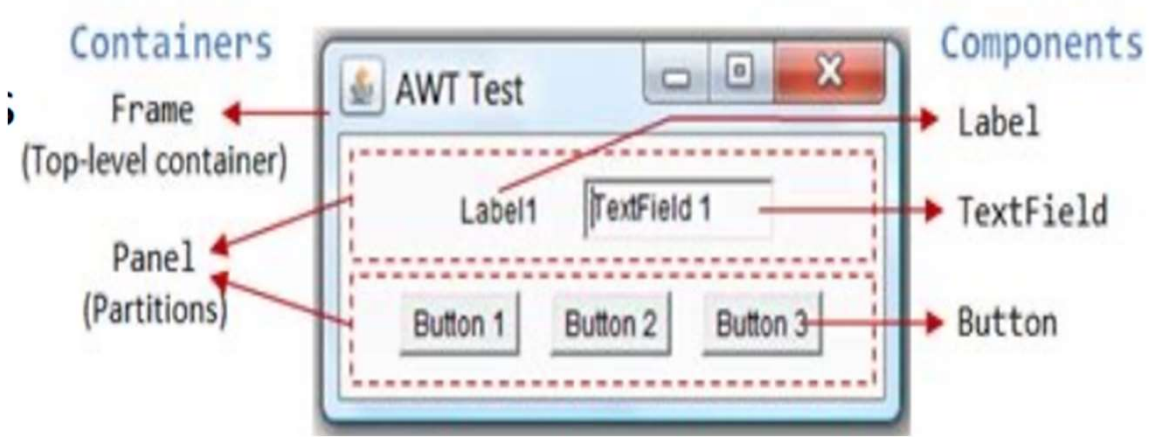
- intermédiaire entre l'utilisateur et la partie "métier" d'un logiciel
- ce qui relève de l'IG:
  - gérer le fait que le bouton soit actif ou non (idem pour la commande "Enregistrer")
  - lancer la sauvegarde quand on clique
- ce qui ne relève pas de l'IG:
  - la sauvegarde elle-même!



Fondamentaux de l'IHM

L'interface visible d'une application est constituée de 2 éléments :

(1) **Les conteneurs** : Contiennent des objets graphiques qui peuvent être des composants ou d'autres conteneurs (ici, panel)



(2) Les composants **atomiques** : boutons, label, cases à cocher, zone de texte, slider, etc.

Une interface utilisateur se compose de :

- Une **fenêtre** de travail
- Une **zone** où afficher les composants graphiques dans cette fenêtre de travail :
    - Un panneau (**Panel**)
- Des **composants** à insérer dans cette fenêtre
    - Boutons, cases à cocher, menus, barre de tâches,…
- Une **mise en page** des composants
    - Le **Layout**: mis bout à bout, centrés, en tableau,…
- La **représentation graphique** des composants
    - Couleur, forme, image,…
- Des gestionnaires **d'évènements**
    - Répondre aux actions de l'utilisateur

Java propose

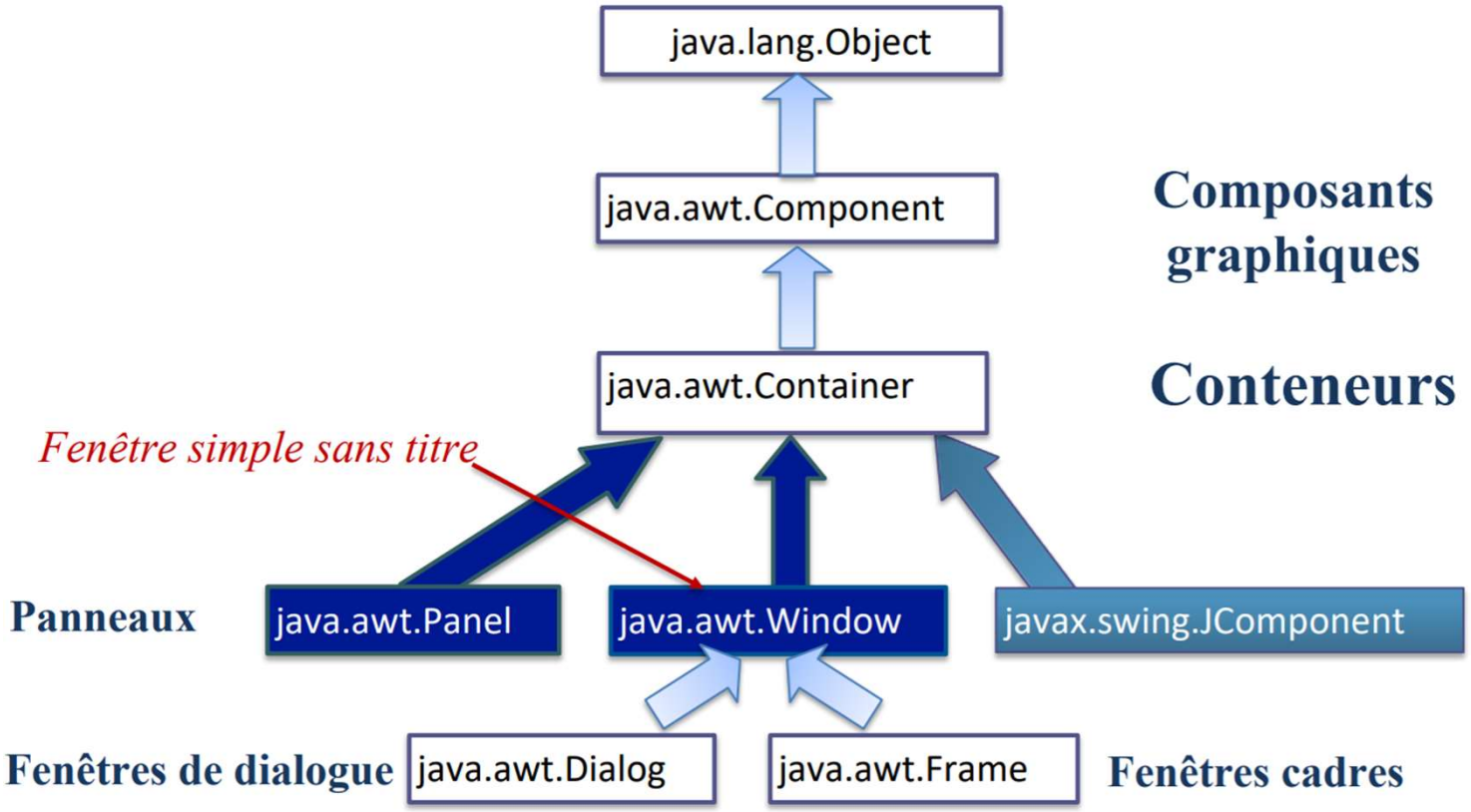
- Des **composants graphiques**
  - `Widgets`
- Des classes de **gestion de la position** des composants sur la fenêtre
  - `LayoutManager`
- Des éléments de **représentation graphique**
  - `Color, Font, Graphics, Point, Rectangle, Image, Icon...`
- Des mécanismes de **gestion d'événements**
  - `java.awt.events`

AWT, SWING

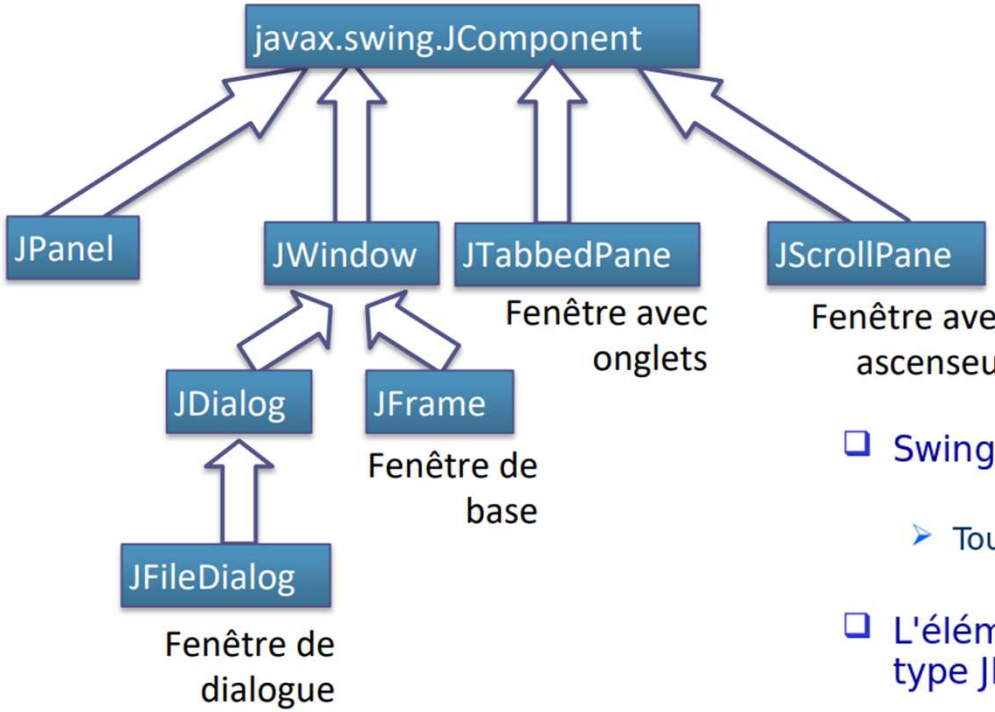
- Première bibliothèque graphique JAVA: **AWT**
  - Package: `java.awt`
  - Utilisation du code qui dépend du système d'exploitation
  - Une gestion des événements
  - Composants limités
- Nouvelle bibliothèque graphique JAVA: **SWING**
  - Package: `javax.swing`
  - Plus riche et plus personnalisable
  - Construite sur AWT, fournit des composants plus performants

Arborescence des packages AWT/**SWING**

Hiérarchie d'héritage des principaux éléments des interfaces graphiques en Java



Arborescence des packages SWING

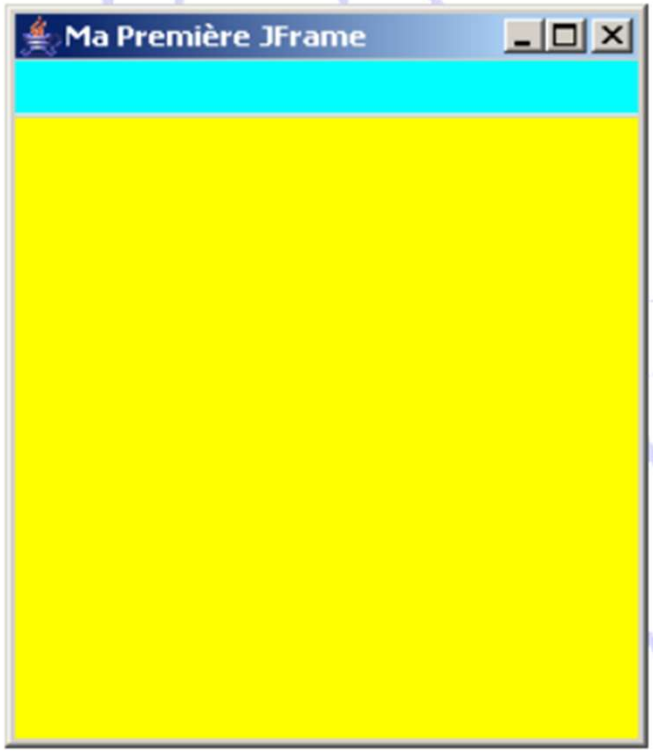


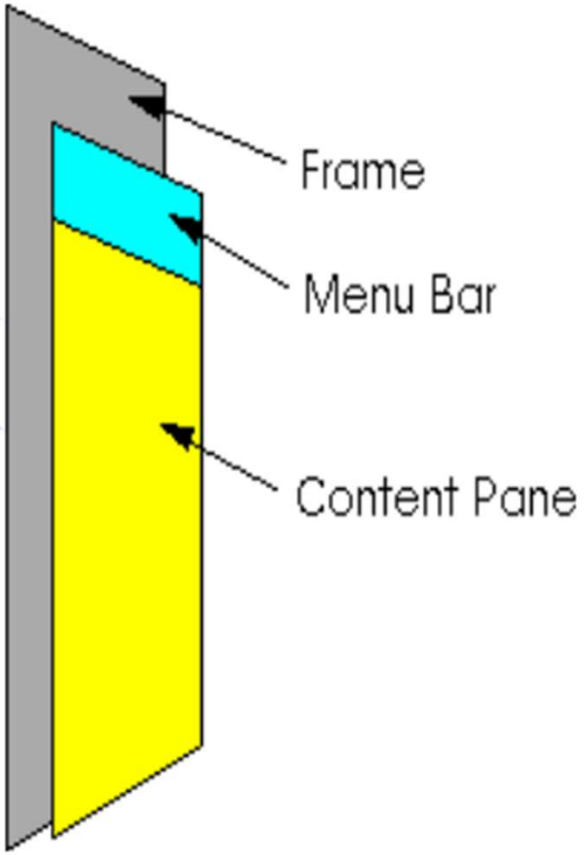
- Swing est une API très fortement orientée objet.
  - Tout est objet : couleur, taille, position, ...
- L'élément racine le plus souvent utilisé est une instance de type JFrame.
  - Une JFrame représente une fenêtre manipulable via le "window manager" de votre système d'exploitation.
  - Il est conseillé de ne pas directement ajouter de composants dans la JFrame : elle contient un "contentPane". C'est dans cet élément qu'il vous faut ajouter vos composants

Arborescence des packages SWING

- Swing est une API très fortement orientée objet.
  - Tout est objet : couleur, taille, position, ...
- L'élément racine le plus souvent utilisé est une instance de type JFrame.
  - Une JFrame représente une fenêtre manipulable via le "window manager" de votre système d'exploitation.
  - Il est conseillé de ne pas directement ajouter de composants dans la JFrame : elle contient un "contentPane". C'est dans cet élément qu'il vous faut ajouter vos composants

Structure générale d'une fenêtre de type JFrame





Rappel: Ecriture d'une application

Une application est un pg qui peut fonctionner en autonome. Il doit comporter une classe contenant une méthode **main** déclarée comme suit:

```
public static void main(String args[]){
//corps du pg principal
}
```

IG avec Swing

SWING offre une panoplie de classes pour la création d'interfaces graphiques. Chacun des objets de ces classes est susceptible à réagir à des événements provenant de la souris ou du clavier.

Bases de Swing Composants, Conteneurs

1. **PRINCIPE**
2. LES CONTENEURS DE HAUT NIVEAU
3. LES CONTENEURS DE NIVEAU INTERMÉDIAIRE
4. LES COMPOSANTS
5. LES COMPOSANTS DE MENU

Composants graphiques (SWING)

- Les **containers**
  - JWindow
    - JFrame
    - JDialog
      - JFileDialog
  - JPanel
  - JTabbedPane
  - JScrollPane

- Les **widgets**
  - JLabel
  - JButton
  - JToggleButton
  - JCheckbox
  - JRadioButton
  - ButtonGroup
  - JComboBox
  - JList
  - JTextField
  - JTextArea
  - JScrollBar
  - JMenuBar
  - JPopupMenu

Différentes catégories de composants (1/2)

Composants de menus	JMenuBar, JMenu, JPopupMenu, JMenuItem, JCheckboxMenuItem, JRadioButtonMenuItem
Composants Conteneurs	JPanel, JScrollPane, JSplitPane, JTabbedPane, JDesktopPane, JToolBar
Composants de haut niveau	JFrame, JDialog, JWindow, JInternalFrame, JApplet

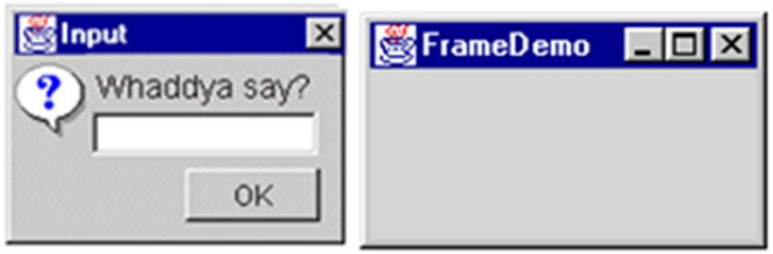
Différentes catégories de composants (2/2)

Composants atomiques	JLabel, JButton, JCheckBox, JRadioButton, JToggleButton, JComboBox, JScrollBar, JSeparator, JSlider, JSpinner, JProgressBar
Composants complexes	JTable, JTree, JList, JFileChooser, JColorChooser, JOptionPane
Composants textuels	JTextField, JFormattedTextField, JPasswordField, JTextArea, JTextPane, JEditorPane

Swing

- D’abord définir un conteneur de haut niveau (**Container**) de cette interface graphique

généralement on choisit de personnaliser par héritage la classe **JFrame** (fenêtre windows)



Définition de l'interface

- Le principe de définition d'une interface est de doter une fenêtre d'un **conteneur de haut niveau** dans lequel viendront se placer les composants de l'interface (boutons, cases à cocher…) ou de **nouveaux conteneurs**

- Cet emboîtement de conteneurs les uns dans les autres permet de définir des interfaces aussi complexes que nécessaire
- Le **conteneur intermédiaire** définit la façon dont seront présentés les éléments (avec ascenseurs, par onglets…)
- Le placement des éléments eux-mêmes dans le conteneur est régi par un objet de placement que l'on associe au conteneur (**layout**)
- Cet objet de placement permet de choisir la façon dont les éléments se situent dans le conteneur et les uns par rapport aux autres (en tableau, en ligne…)
- Les composants de l'IG appartiennent à des classes héritant du composant graphique **JComponent**
- **JComponent** hérite du composant graphique AWT **Container** qui permet à un composant d'en contenir d'autres

Composants et conteneurs

- Les composants doivent être placés dans des conteneurs
- Les conteneurs servent à créer une hiérarchie de composants
- Noeuds = conteneurs, feuilles = composants

La démarche suivie:

1. Faire un dessin de l'interface en plaçant tous les composant
2. Créer une classe héritée de **JFrame** ou de **JDialog**
3. Définir si nécessaire, les composants de la barre de menu, des rubriques et sous rubriques
4. Ajouter cette barre de menu à la fenêtre par la méthode **setMenuBar (JmenuBar)** de cette dernière
5. Récupérer le conteneur associé à la fenêtre par la méthode **getContentPane** de cette dernière. Ce conteneur est de classe **JPanel**, (on peut le modifier en utilisant la méthode **setContentPane** de la fenêtre).
6. Choisir l'objet de placement le plus approprié: **BorderLayout, FlowLayout, GridLayout, GridBagLayout, BoxLayout** ou **OverlayLayout** et l'associer au conteneur par la méthode setLayout de ce dernier
7. à l'aide de la méthode **add** du conteneur, placer les composants d'interface dans les zones définies par l'objet de placement choisi à l'étape 6. Ceci uniquement pour les composants qui sont seuls dans une zone.

On mettra un conteneur de niveau intermédiaire dans les zones dans lesquelles doivent se trouver plusieurs composants. Ce conteneur est un objet de classe **Jpanel, JScrollPane, JLayerredPane, JSplitPane, JTabbedPane** ou **JInternalFrame** selon le comportement souhaité.

8. Pour chacun de ces conteneurs, refaire les étapes 6 et 7 de façon à y positionner des composants ou de nouveaux conteneurs pour lesquels on suivra la même démarche. Lorsque c'est terminé ajouter ce conteneur à son propre conteneur par la méthode **add** de ce dernier.
9. Continuer jusqu'à ce que tous les composants aient trouvé leurs places.

Premier exemple

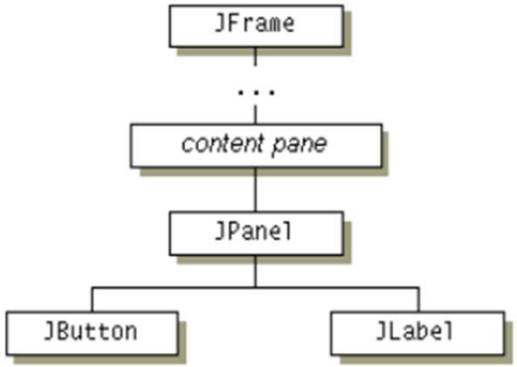
- Nous utilisons ici les composants les plus communs d'une interface.

Leur traduction en Swing :

- Une fenêtre principale (avec boutons d'agrandissement, réduction, fermeture…) : **JFrame**
- Un plan rectangulaire : **JPanel**
- Un bouton : **JButton**
- Une zone texte : **JLabel**

Premier exemple

- La JFrame est un composant qui contient tous les autres.
- Le JPanel est un container intermédiaire qui va se charger de contenir le label et le bouton.
- Le JButton et le JLabel sont des composants atomiques. Autres exemples : zone de texte éditable (JTextField), table (JTable), etc… Diagramme de hiérarchie (contenance) – arborescence :



- Remarque : tous les conteneurs de haut niveau dans la hiérarchie (JFrame, JWindow…) contiennent un container intermédiaire qui se charge de contenir les fils (contentPane)

Le code de l'exemple

MaFenetre.java

```
1  import javax.swing.JButton;
2  import javax.swing.JFrame;
3  import javax.swing.JLabel;
4  import javax.swing.JPanel;
5
6  public class MaFenetre extends JFrame {
7  JButton monbouton;
8  JLabel label;
9  JPanel monpanel;
10
11 public MaFenetre() {
12     initComponents();
13     setTitle("Exemple");
14     setSize(500,100);
15     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16 }// fin constructeur
17 private void initComponents() {
18     monpanel= new JPanel();
19     label =new JLabel("un petit texte");
20     monbouton= new JButton("clic clic");
21     monpanel.add(monbouton);
22     monpanel.add(label);
23     add(monpanel);
24 }
```

```
26 public static void main(String[] args) {
27     JFrame mf=new MaFenetre();
28     mf.setVisible(true);
29 }
30 }
```

`setDefaultCloseOperation()` de JFrame

JFrame.EXIT_ON_CLOSE
— **quitte** l'application

JFrame.HIDE_ON_CLOSE (choix par défaut)
— Cache la fenêtre **sans quitter l'application**
(quand on a plusieurs fenêtres, on peut souhaiter fermer une fenêtre et continuer l'application)

JFrame.DISPOSE_ON_CLOSE
— Quand on a plusieurs fenêtres, **rend la main à la fenêtre parent** tout en fermant la fenêtre courante

JFrame.DO_NOTHING_ON_CLOSE
— **Ignore** la demande de fermeture
(quand par ex on gère « Quitter » dans un menu de fenêtre)