

## Java Swing

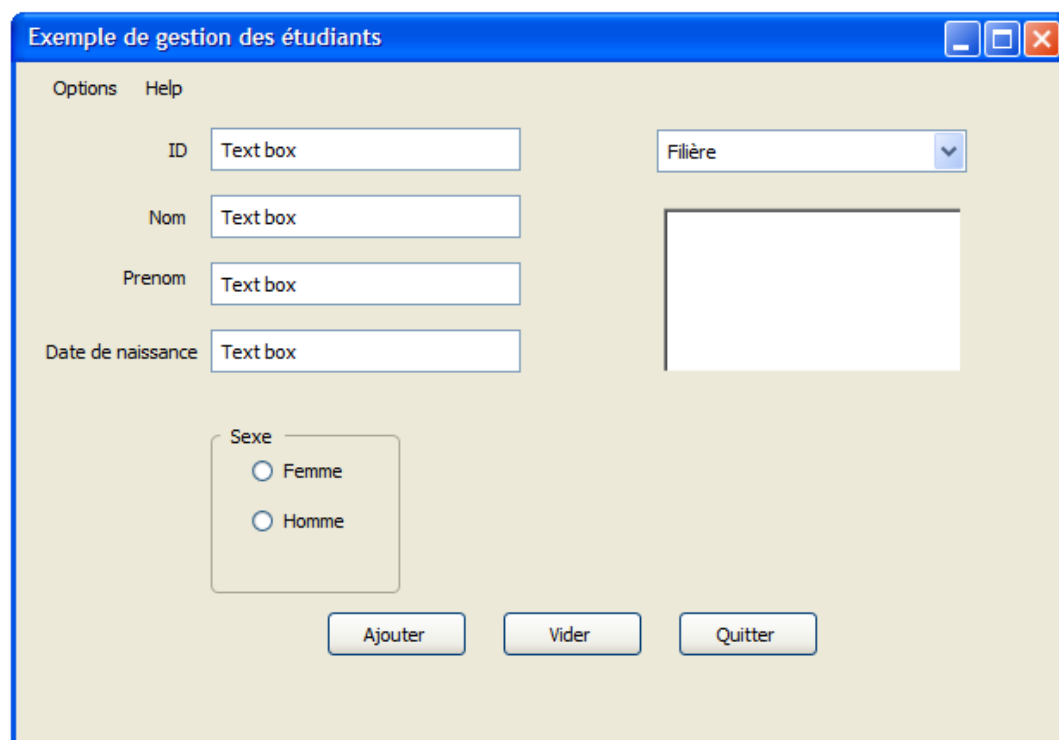
### Atelier II: Création des interfaces événementielle à partir des maquettes

#### Objectif

- Vers la fin de cet atelier vous devrez être capable de mieux gérer l'interface grace aux layouts proposés par Swing ;
- Personnaliser les composants JList et les JCombo ;
- Création d'une barre de menu
- Création de l'interactivité grâce aux écouteurs (Listners).

#### Etape 1 : création de la fenêtres

L'application d'authentification de l'atelier 1 sera mise à côté, et entamant une nouvelle interface qui prendra en charge la gestion des étudiants à travers plusieurs formulaire. Le premier formulaire est réservé pour l'ajout d'un nouveau étudiant. La figure suivante représente la maquette du formulaire d'ajout :



Exemple de gestion des étudiants

Options Help

ID Text box Filière

Nom Text box

Prenom Text box

Date de naissance Text box

Sexe

☐ Femme

☐ Homme

Ajouter Vider Quitter

Figure1. maquette 1

Pour réaliser cette maquette il faut la décomposer sous un ensemble de panneaux qui intègrent des éventuels panneaux ou des objets atomiques. La décomposition doit se faire tout en gardant l'ordre d'apparition des objets. La démarche repose principalement sur les gestionnaires des panneaux proposés par Swing. La figure suivante représente une proposition parmi plusieurs qui peuvent simplifier l'anatomie de cette interface :

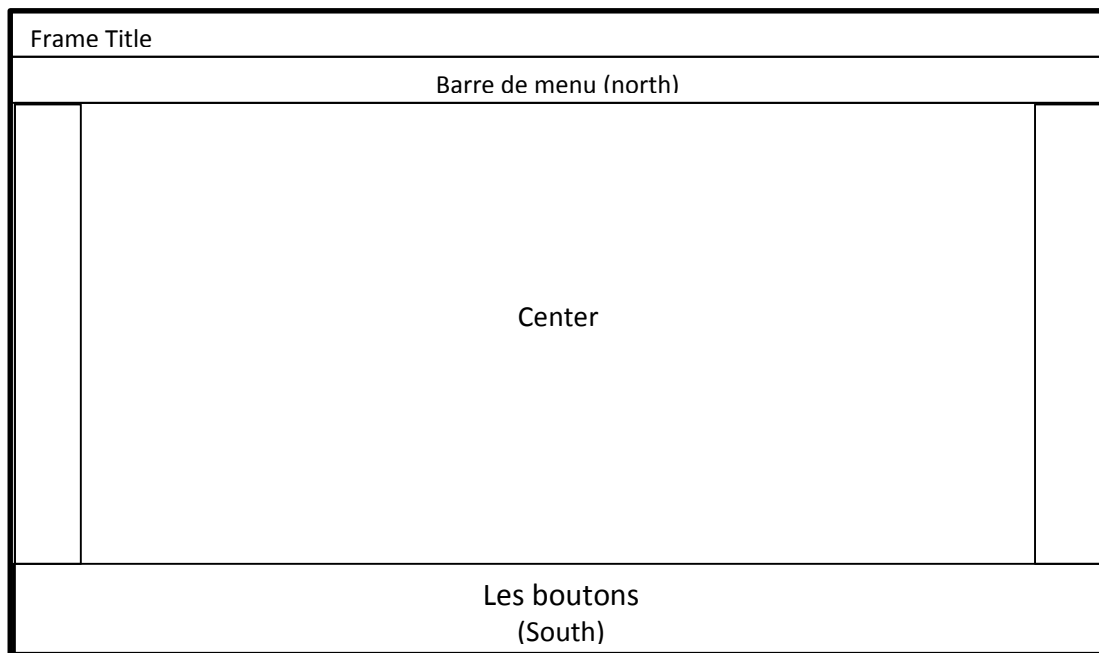


Figure2. La maquette selon le BorderLayout

## Etape 2 : Création des panneaux

### a. Région sud

Le layout (gestionnaire du panneau) principal utilisé est le *BorderLayout*. Ce dernier se caractérise par cinq régions. A chaque que vous ajoutez un composant au Layout il faut indiquer dans quelle région exactement. Exemple :

```
Panel p = new Panel();
p.setLayout(new BorderLayout());
p.add(new Button("Vider"), BorderLayout.SOUTH);
```

Cependant, dans notre cas on a besoin d'ajouter trois boutons. Il est préférable de créer un panneau de type *FlowLayout* réservé aux boutons, et par la suite l'ajouter dans la région Sud.

### b. Région Nord

On réserve la région nord pour la barre de menu. La création de la barre de menu est une instantiation de la classe *JMenuBar* avec un titre. Cette dernière contient des objets de *JMenu*. Les *JMenu* sont les menus proposés par la fenêtre afin de déclencher une action

particulière. Un JMenu également peut contenir des sous éléments (sous menu) qui représentent une instantiation des objets JMenuItem. L'exemple suivant illustre clairement la démarche de création d'une barre de menu avec un deux menus, dont chacun incorpore un sous menu.

```
private JMenuBar menuBar = new JMenuBar();
private JMenu test1 = new JMenu("Fichier");
private JMenu test2 = new JMenu("Help");
private JMenuItem item1 = new JMenuItem("ouvrir");
private JMenuItem item2= new JMenuItem("online help");
// .....
JPanel monpan = new JPanel(new BorderLayout());

test1.add(item1);
test2.add(item2);
menuBar.add(test1);
menuBar.add(test2);
// this.setJMenuBar(menuBar);
monpan.add(menuBar, BorderLayout.NORTH);
this.getContentPane().add(monpan);
```

### C. Region du centre

C'est la région la plus importante et relativement la plus compliquée. En effet, le centre va contenir plusieurs éléments dont l'apparition doit être bien organisée. On a 13 éléments qui doivent avoir un emplacement bien précis dans le centre, et à travers lesquels l'utilisateur peut saisir des informations. Une proposition parmi plusieurs consiste à diviser la partie du centre en deux zones. Les deux zones ont des gestions de panneaux de type *GridLayout*.

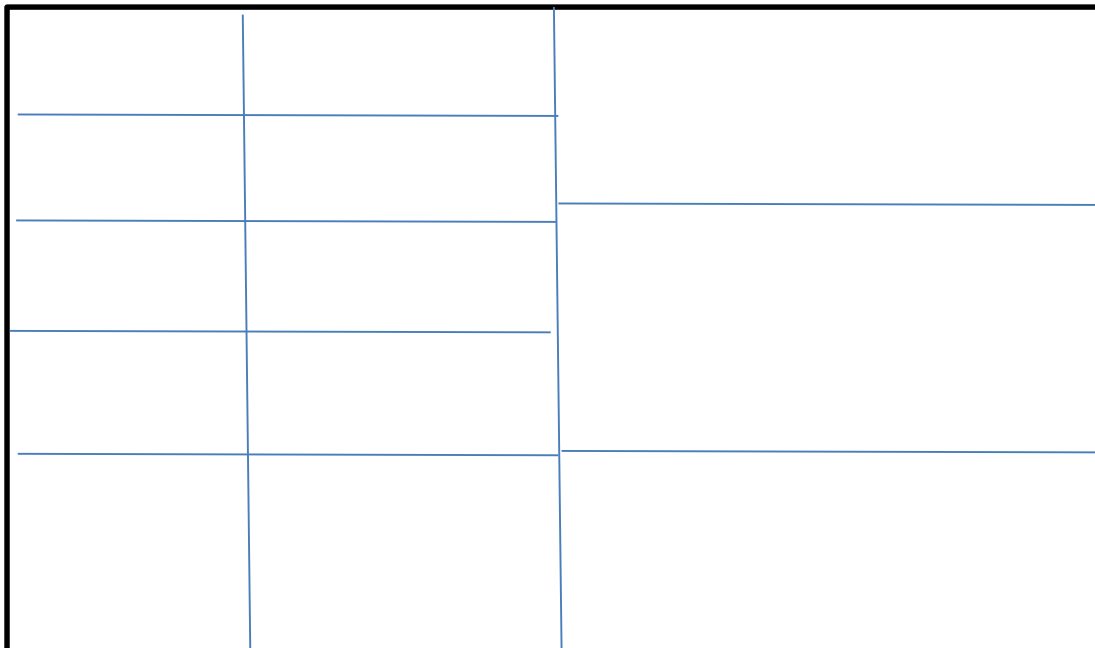


Figure3. Organisation de la region du centre selon deux GridLayout

La création d'un *GridLayout* est similaire aux autres, sauf ici on précise le nombre des colonnes et des lignes. Il est aussi utile parfois de paramétrer l'espace entre les cellules. L'exemple ci-dessous donne une idée sur l'instanciation d'un *GridLayout* :

- `GridLayout(int rows, int cols)`
- `GridLayout(int rows, int cols, int hgap, int vgap)`

Ajouter chaque composant dans une cellule, et en cas de besoin ajouter des espaces entre les cellules.

#### Etape 4 : Gestion des évènements

Les évènements qui vont servir se sont les cliques sur les boutons et les menus. A partir de l'interface d'authentification on appelle (le bouton Ok) l'interface *GestionEtudiant* par une instantiation (`new GestionEtudiant()`), et pour le bouton *Quitter* la fermeture (`System.Exit(0)`).

Pareil pour les boutons de la deuxième interface des *ActionListener* sont mis en œuvres pour l'interactivité de l'application. Il est aussi possible d'appliquer un événement *ActionListener* sur un *MenuItem* :

```
menuFileItem.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            System.out.println(" coucou ");
        }
    }
);
```

Remarque importantes :

Pour les menus l'événement *ActionListener* n'est pas capturer, il faut mettre en œuvre un autre, précisément l'évènement *MenuListener* :

```
JMenu menu = new JMenu("MyMenu");
menu.addMenuListener(new MenuListener() {

    @Override
    public void menuSelected(MenuEvent e) {
        System.out.println("Seletion");
    }

    public void menuDeselected(MenuEvent e) {
        System.out.println("déSelection");
    }

    public void menuCanceled(MenuEvent e) {
        System.out.println(" annulation");
    }

});
```