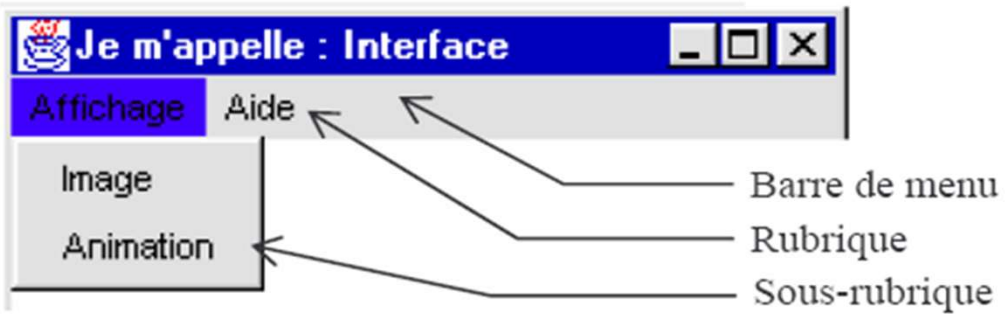
Bases de Swing Composants, Conteneurs

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

La classe JMenuBar

On peut doter une fenêtre d'une barre de menu qui s'affichera en haut et permettra d'accéder à des rubriques et sous rubriques. On utilisera pour cela les classes **JMenuBar**, **JMenu** et **JMenuItem**.



*La classe **JMenuBar***

Elle permet de définir une barre de menu. Ses principales méthodes sont :

void add(JMenu) qui ajoute une rubrique dans la barre

JMenu getMenu(int) qui retourne la rubrique dont le numéro est spécifié

On l'associe à la fenêtre (classe **JFrame**, **JApplet** ou **JInternalFrame**) en utilisant la méthode **setJMenuBar**

La classe JMenu

Elle permet de définir une rubrique dans une barre de menu. Ses principales méthodes sont :

JMenu(String) qui crée et définit le texte de la rubrique

JMenuItem add(JMenuItem) qui ajoute une sous rubrique (elles apparaîtront dans l'ordre où elles sont ajoutées). La valeur de retour est cette sous-rubrique.

JMenuItem add(String) qui crée une sous_rubrique (**JMenuItem**) à partir de la chaîne de caractères et l'ajoute. La valeur de retour est cette sous-rubrique.

void addSeparator() qui place un séparateur entre sous rubriques

void insert(JMenuItem,int) qui ajoute une sous rubrique au rang spécifié

void insertSeparator(int) qui place un séparateur entre sous rubriques au rang spécifié.

void addActionListener(ActionListener) pour associer l'objet qui traitera l'événement de sélection de cette rubrique . En général inutile s'il y a des sous_rubriques puisque chacune traitera ses propres événements (voir **JMenuItem**)

Remarque : la classe **Jmenu** hérite de **JMenuItem** de sorte que l'un des éléments d'une rubrique peut être lui même une rubrique.

La classe JMenuItem

Elle permet de définir une sous-rubrique dans une barre de menu. Ses principales méthodes sont :

JMenuItem(String) construction avec définition du texte de la sous-rubrique.

JMenuItem (ImageIcon) construction avec une icône dans la sous-rubrique.

JMenuItem (String,ImageIcon) construction avec définition du texte et d'une icône dans la sous-rubrique.

void setEnabled(boolean) valide ou invalide la possibilité d'utiliser la sous-rubrique.

String getText() qui retourne le texte contenu dans la sous-rubrique.

void addActionListener(ActionListener) pour associer l'objet qui traitera l'événement de sélection de cette sous-rubrique

Les composants graphiques: Contrôle des couleurs

- Contrôle des couleurs d'un composant
  - Deux méthodes permettent de définir les couleurs d'un composant
    - setForeground (Color c) : la couleur de l'*encre* avec laquelle on écrira sur le composant
    - setBackground (Color c) : la couleur du fond
  - Ces deux méthodes utilisent un argument instance de la classe java.awt.Color.
    - La gamme complète de couleurs prédéfinies est listée dans la page de documentation relative à la classe Color.
  - Il est aussi possible de créer une couleur spécifique (RGB)

```
int r = 255, g = 255, b = 0 ;
Color c = new Color (r, g, b) ;
```

Les composants graphiques: Contrôle des polices de caractères

- Contrôle des polices de caractères
  - La police utilisée pour afficher du texte dans un composant peut être définie avec setFont(...) avec comme argument une instance de java.awt.Font.
    - **Font f = new Font ("TimesRoman", Font.PLAIN, 14) ;**
  - Les constantes de style de police sont en réalité des valeurs entières, parmi celles citées ci-après :
    - **Font.BOLD**
    - **Font.ITALIC**
    - **Font.PLAIN**
    - **Font.BOLD + Font.ITALIC**
  - Les tailles en points doivent être définies avec une valeur entière.

DISPOSITION DES COMPOSANTS

1. FlowLayout
2. BorderLayout
3. GridLayout
4. GridBagLayout

Gestion du placement des composants

- Comment positionner les composants les uns par rapport aux autres?
- Comment un container agence-t-il visuellement les composants qu'il contient?
- Que se passe-t-il quand on agrandit la fenêtre? Les composants doivent-ils s'agrandir? Ajoute-t-on de l'espace? Où?
- Utilisation d'un gestionnaire de placement (layout)
- Chaque layout définit une stratégie de positionnement

Gestionnaires de placement

- L'objet qui gère la disposition des composants est une instance implémentant l'interface **java.awt.LayoutMangager**. Cet objet doit être ajouté au conteneur, soit lors de l'appel à son constructeur, soit par la méthode **setLayout**. Un seul LayoutManager est autorisé par container.

```
fenetre.getContentPane().setLayout(new BorderLayout());
```

- Les différentes stratégies de positionnement doivent prendre en compte la position et la taille des composants qui lui sont confiés
- Chaque composant a deux tailles:
  - La taille effective (size)
  - La taille idéale (preferredSize)
- Un layout va essayer de rendre la taille réelle la plus proche possible de la taille préférée compte tenu des contraintes dues à sa stratégie et dues à la taille réelle du conteneur

Les objets de placement

- A chaque conteneur est associé un gestionnaire de présentation (*layout manager*)
- Le gestionnaire de présentation gère le positionnement et le (re)dimensionnement des composants d'un conteneur.
- Le ré-agencement des composants dans un conteneur a lieu lors de :
  - ✓ la modification de sa taille,
  - ✓ le changement de la taille ou le déplacement d'un des composants.
  - ✓ l'ajout, l'affichage, la suppression ou le masquage d'un composant.
- Les principaux gestionnaires de présentation de l'AWT sont: **FlowLayout**, **BorderLayout, GridLayout, GridBagLayout**

Les objets de placement

- Tout conteneur possède un gestionnaire de présentation par défaut.
- Les gestionnaires de présentation par défaut sont:
  - ✓ Le **BorderLayout** pour **Window** et ses descendants (J**Frame**, J**Dialog**, …)
  - ✓ Le **FlowLayout** pour J**Panel** et ses descendants

- Une fois installé, un gestionnaire de présentation fonctionne "tout seul" en interagissant avec le conteneur.

FlowLayout

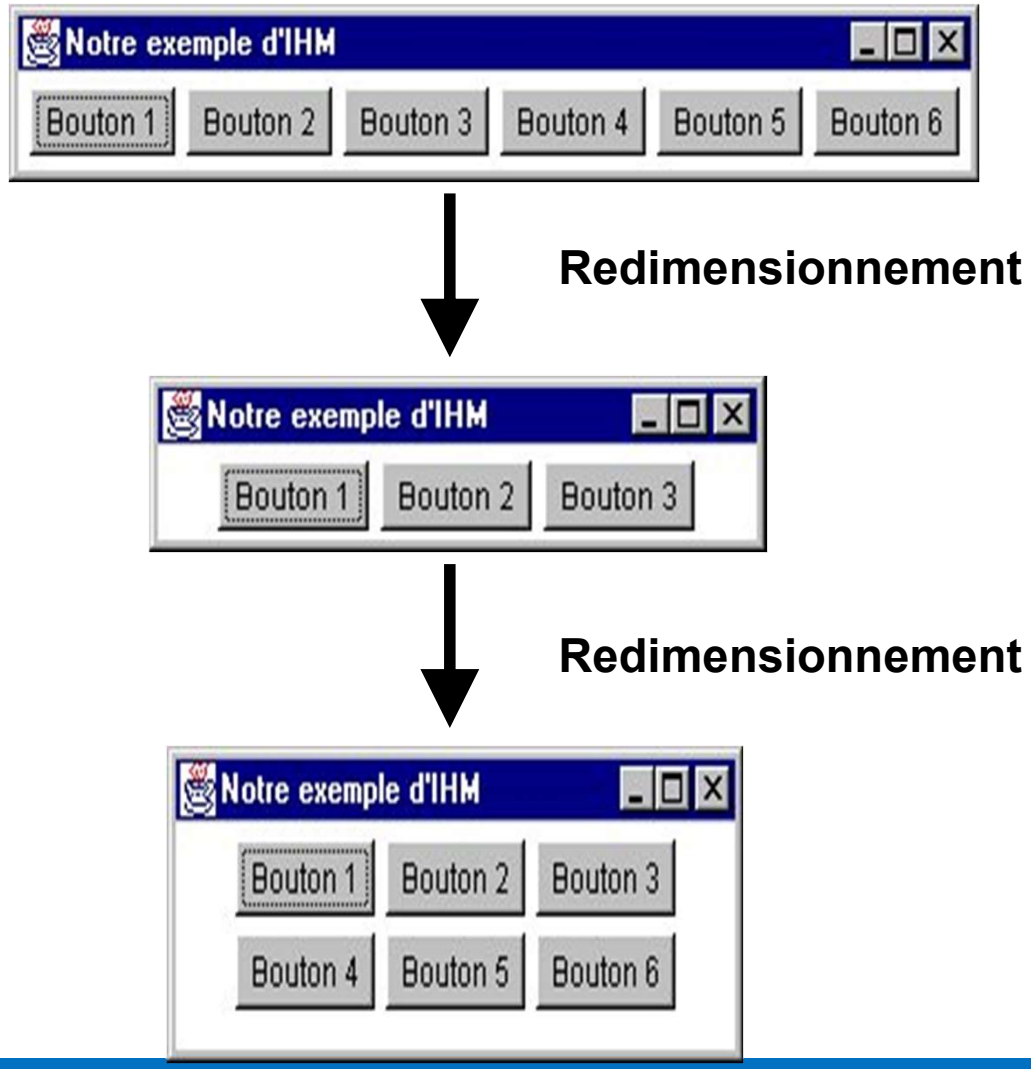
Cette classe permet un placement ligne par ligne. Elle possède un constructeur qui définit la façon dont les composants seront ajoutés par la suite à l'aide de la méthode **add(Component)**. Ce constructeur accepte trois paramètres selon le modèle suivant **FlowLayout(int,int,int)**. Le premier paramètre indique comment seront alignés les composants, il peut prendre les valeur suivantes : FlowLayout.CENTER, FlowLayout.LEFT ou FlowLayout.RIGHT. Le deuxième paramètre donne l'espacement horizontal entre composants (en pixel). Le dernier paramètre donne l'espacement vertical entre composants (en pixel).

- FlowLayout(int align, int hgap, int vgap)
  - hgap est l'espacement horizontal (d'une colonne à une autre)
  - vgap est l'espacement vertical (d'une ligne à une autre)
- Le FlowLayout est le plus simple des managers de l'AWT
- Gestionnaire de présentation utilisé par défaut dans les Panel si aucun LayoutManager n'est spécifié.
- Un FlowLayout peut spécifier :
  - ✓ une justification à gauche, à droite ou centrée,
  - ✓ un espacement horizontal ou vertical entre deux composants.
  - ✓ Par défaut, les composants sont centrés à l'intérieur de la zone qui leur est allouée.

FlowLayout

- La stratégie de disposition du FlowLayout est la suivante:
  - Respecter la **taille préférée** de tous les composants contenus.
  - Disposer autant de composants que l'on peut en faire tenir horizontalement à l'intérieur de l'objet **Container**.
  - Commencer une nouvelle rangée de composants si on ne peut pas les faire tenir sur une seule rangée.
  - Si tous les composants ne peuvent pas tenir dans l'objet **Container**, ce n'est pas géré (c'est-à-dire que les composants peuvent ne pas apparaître).

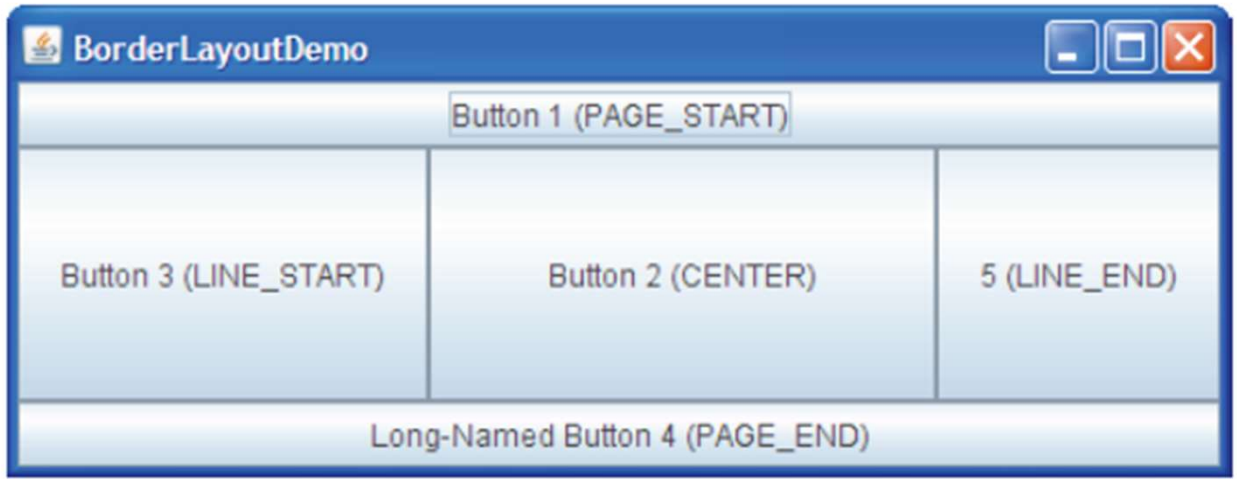
FlowLayout


Notre exemple d'IHM
Bouton 1
Bouton 2
Bouton 3
Bouton 4
Bouton 5
Bouton 6
Redimensionnement
Notre exemple d'IHM
Bouton 1
Bouton 2
Bouton 3
Redimensionnement
Notre exemple d'IHM
Bouton 1
Bouton 2
Bouton 3
Bouton 4
Bouton 5
Bouton 6

BorderLayout

- BorderLayout divise son espace de travail en cinq zones géographiques : North, South, East, West et Center.
- Les composants sont ajoutés par nom à ces zones (un seul composant par zone).
- Exemple
  add("North", new Button("Le bouton nord !"));
- Si une des zones de bordure ne contient rien, sa taille est 0.

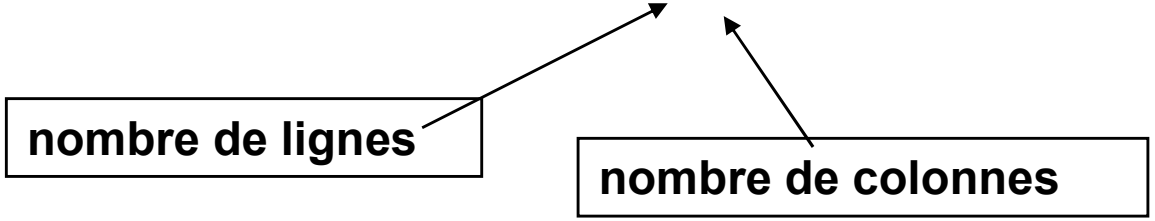
• Layout par défaut des conteneurs de haut-niveau
• Le composant au centre essaie d'occuper le maximum d'espace disponible



GridLayout

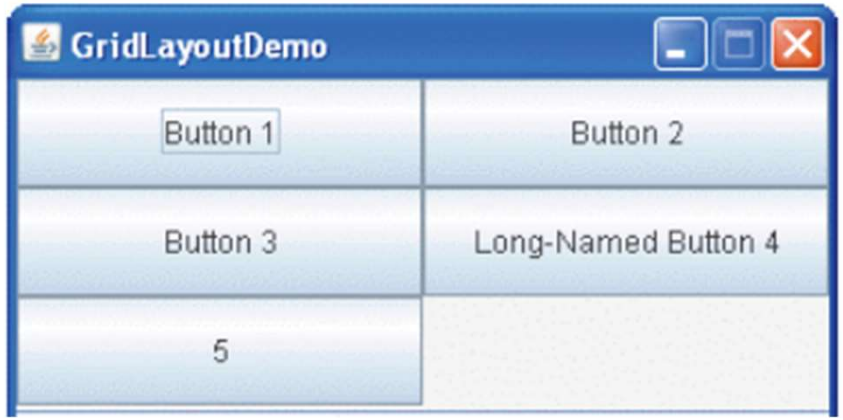
- Le GridLayout dispose les composants dans une grille.
  - Découpage de la zone d'affichage en lignes et en colonnes qui définissent des cellules de dimensions égales.
  - Chaque composant à la même taille
    - quand ils sont ajoutés dans les cellules le remplissage s ’effectue de gauche à droite et de haut en bas.
  - Les 2 paramètres sont les rangées et les colonnes.
  - Construction d'un GridLayout :

new GridLayout(3,2);



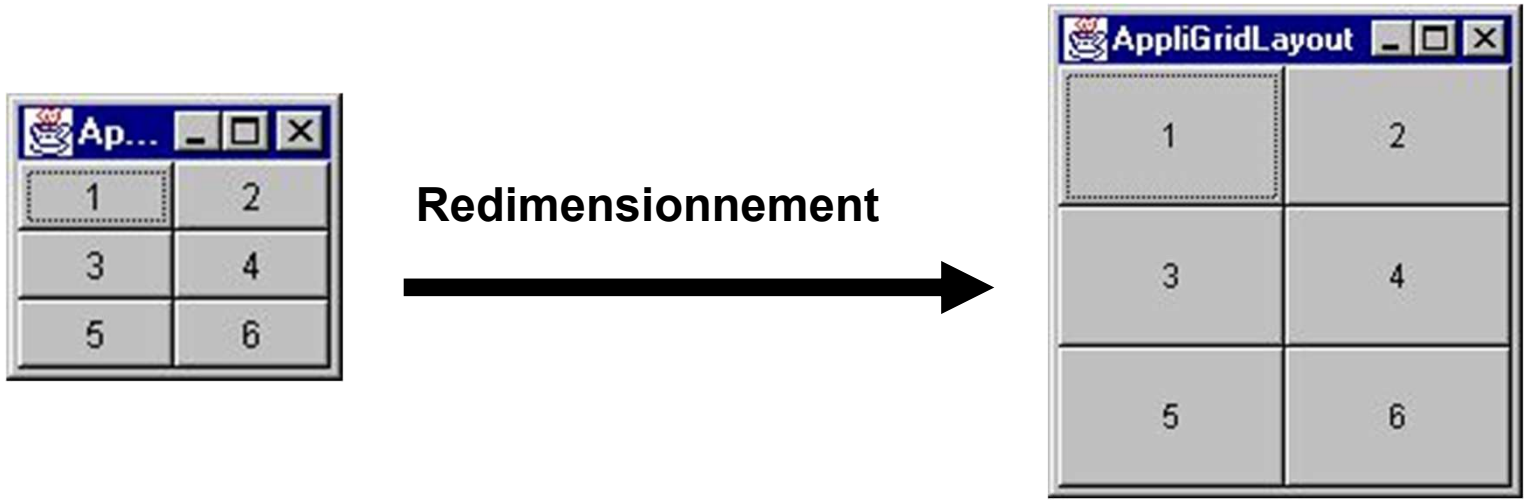
GridLayout

Cette classe permet un placement sur un tableau. Elle possède un constructeur qui définit les dimensions de ce tableau et les espacements entre les composants. Ce constructeur accepte quatre paramètres selon le modèle suivant **GridLayout(int,int,int,int)**. Le premier paramètre est le nombre de lignes du tableau tandis que le deuxième est le nombre de colonnes du tableau.. Le troisième paramètre donne l'espacement horizontal entre composants (en pixel). Le dernier paramètre donne l'espacement vertical entre composants (en pixel).
Les composants seront ajoutés par la suite à l'aide de la méthode **add(Component)** qui remplira le tableau ligne par ligne.



GridLayout

- Lors d'un redimensionnement les composants changent tous de taille mais leurs positions relatives ne changent pas.



GridBagLayout

Cette classe permet un contrôle plus précis du placement des composants grâce à l'utilisation d'un objet de classe **GridBagConstraints** qui permet de définir les règles de placement pour chaque composant. **GridBagLayout** permet de placer les composants dans une grille dont les cases sont numérotées de gauche à droite et de haut en bas à partir de 0. Chaque composant peut occuper une ou plusieurs cases et la taille des cases est ajustée en fonction des composants qu'elles contiennent. Toutes les cases d'une même ligne ont la même hauteur qui correspond à la hauteur nécessaire pour placer le composant le plus haut de cette ligne. De même toutes les cases d'une même colonne ont la même largeur qui correspond à la largeur nécessaire pour placer le composant le plus large de cette ligne.

Pour placer un composant il faut suivre la procédure suivante :

- appeler la méthode **setLayout** avec en paramètre le nom de l'objet de placement de classe **GridBagLayout**
- préparer un objet de classe **GridBagConstraints** permettant de définir la position, la taille et le comportement de l'objet à placer
- utiliser la méthode **setConstraints** de l'objet de placement en lui donnant en paramètre le composant d'interface graphique à placer et l'objet de classe **GridBagConstraints** précédemment préparé.
- ajouter ce composant à l'interface graphique par la méthode **add**

GridBagLayout

- **gridx, gridy**: Spécifie la ligne et la colonne le composant à placer la première colonne est gridx=0 et la première ligne est gridy=0

- **gridwidth, gridheight:** Spécifie le nombre de colonnes (pour gridwidth) ou de lignes (pour gridheight) que le composant va utiliser dans la grille. Il s'agit du nombre de cases qu'il va prendre et non du nombre de pixels. Ces propriétés doivent être une valeur entière supérieure ou égale à 1. Cependant, il est également possible de spécifier deux constantes particulières : GridBagConstraints.REMAINDER et GridBagConstraints.RELATIVE. REMAINDER permet de préciser que ce composant est le dernier de sa ligne ou de sa colonne. Nous vous recommandons de toujours l'utiliser. RELATIVE permet de préciser l'avant dernier composant. Il occupera toutes les cellules disponibles depuis sa position jusqu'à l'avant dernière cellule.
- **fill:** Cette propriété permet de définir la manière dont un composant sera (ou ne sera pas) redimensionné lorsque l'espace qui lui est alloué est supérieur à celui de ses désidératas. Les valeurs possibles sont :
  - GridBagConstraints.NONE par défaut,
  - GridBagConstraints.HORIZONTAL remplissage horizontal du composant parent,
  - GridBagConstraints.VERTICAL remplissage vertical du composant parent,
  - GridBagConstraints.BOTH remplissage de toute la zone dans lequel le composant est ajouté

GridBagLayout

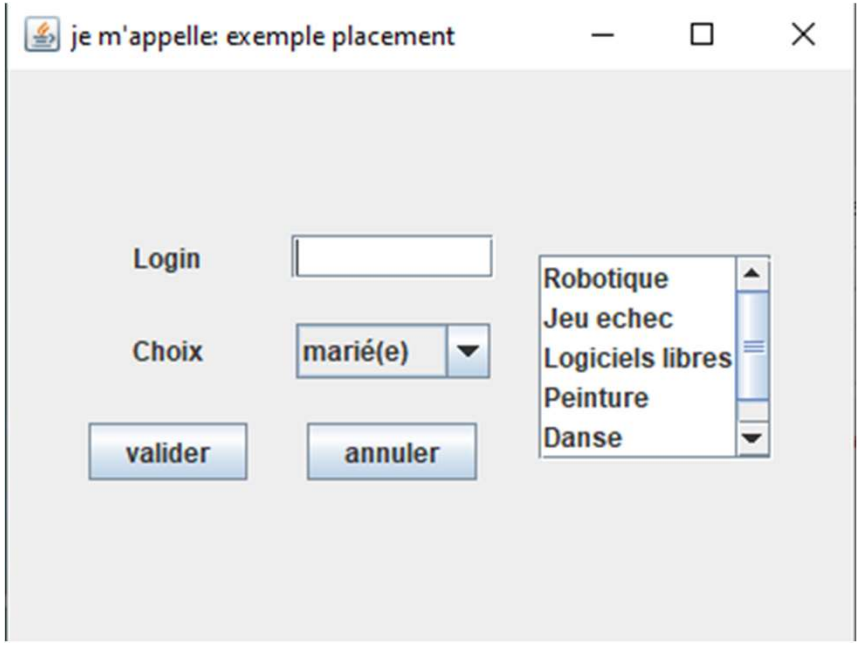
- **ipadx, ipady:** Ces attributs permettent de définir des marges internes au composant
- **insets**: Permet de préciser des marges en pixels autour du composant. Ce Champ est positionné en faisant new Insets(h,g,b,d) où h définit la marge haute, g la marge gauche, b la marge basse et d la marge droite.
- **Anchor:** La propriété anchor permet de spécifier un point d'ancrage à un composant à l'intérieur de sa (ou ses) cellule(s). Chaque champ peut prendre les valeurs suivantes:

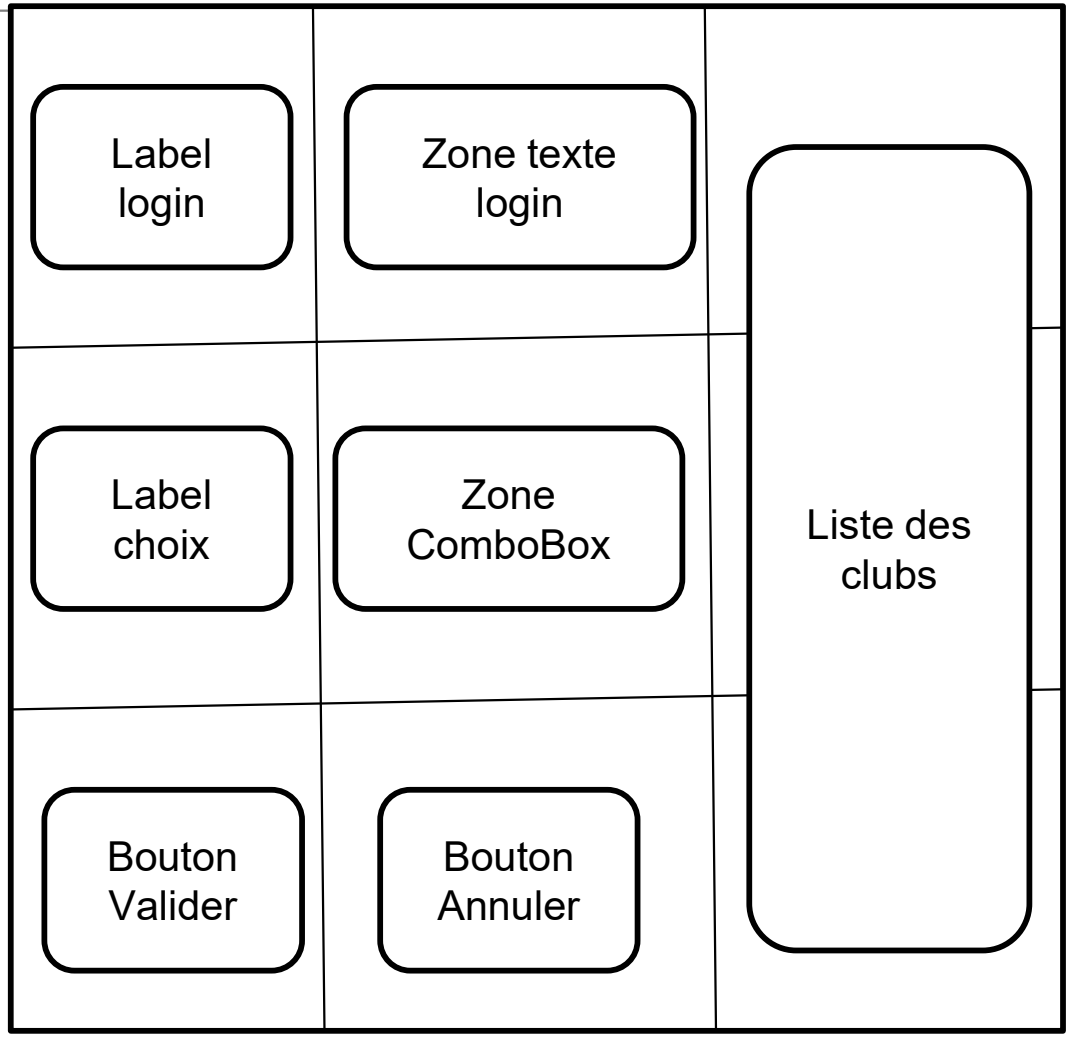
FIRST_LINE_START	PAGE_START	FIRST_LINE_END
LINE_START	CENTER	LINE_END
LAST_LINE_START	PAGE_END	LAST_LINE_END

- **Weightx, weighty**: Ces deux propriétés permettent de définir comment l'espace supplémentaire sera distribué parmi les composants horizontalement (weightx) et verticalement (weighty). Ils expriment un pourcentage de répartition de largeur (respectivement de hauteur) pour ce composant.

Exemple d'application

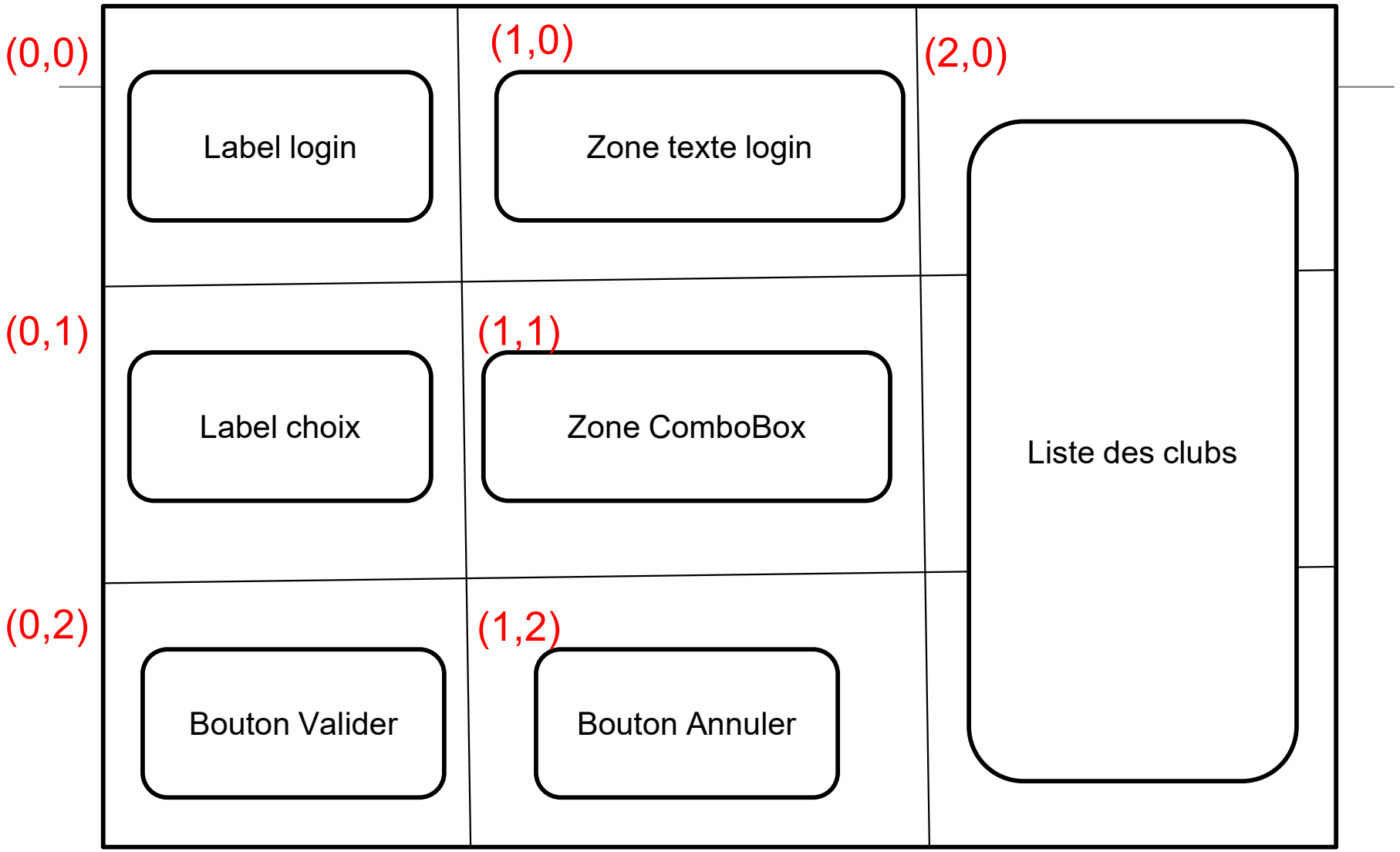
Placement des composants





Les propriétés gridx et gridy

Ces propriétés permettent de positionner un composant dans la grille

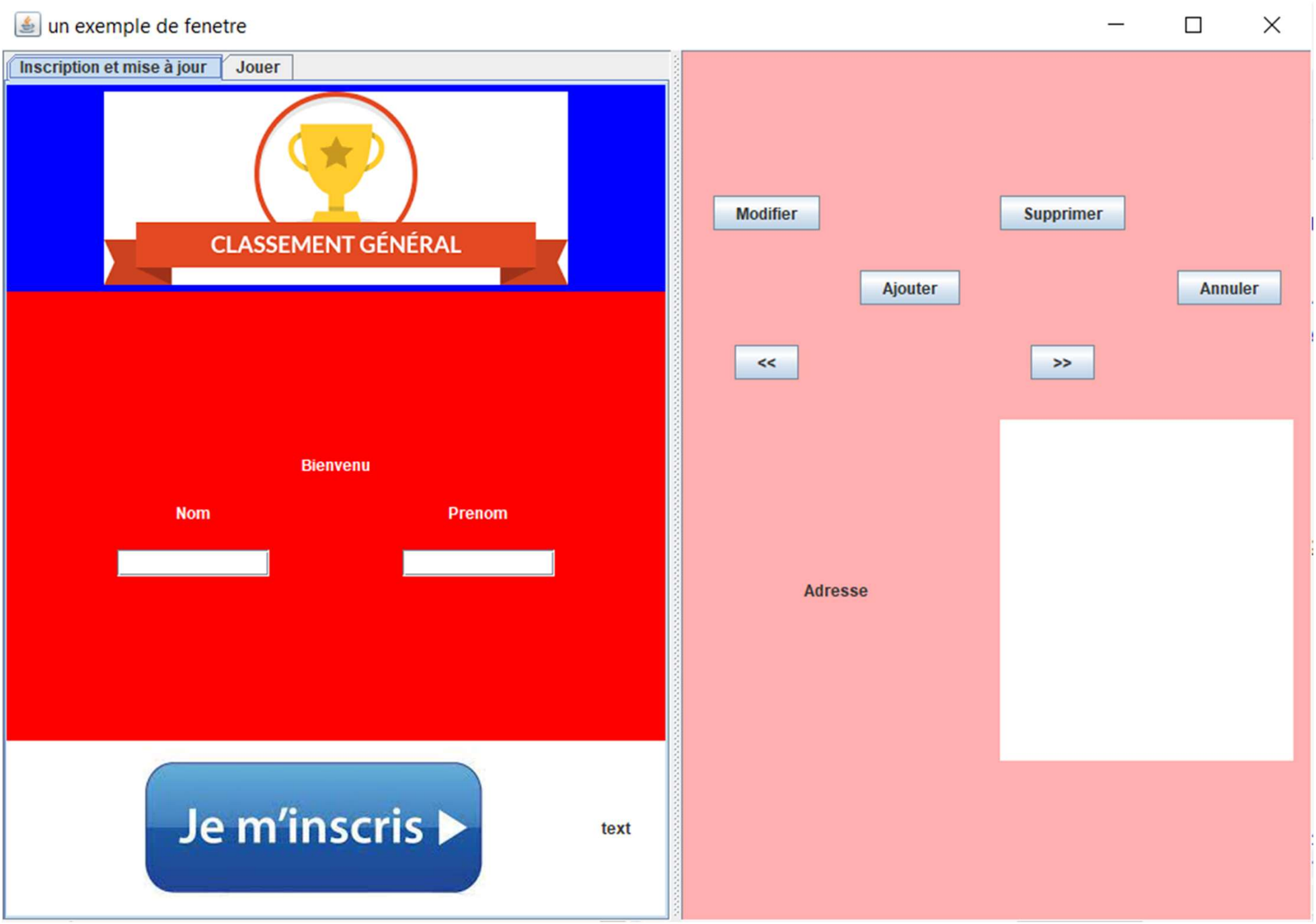


```java
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JScrollPane;
import javax.swing.JTextField;

public class FenetrePlacement extends JFrame {

//definition des noms de composants

private JButton valider, annuler;//les boutons valider et annuler
private JTextField txtlogin;//la zone ou on saisie le login
private JLabel login, choix;
private JComboBox<String> listchoix;
/**
 * Depuis Java 7, l'objet JComboBox peut être paramétré avec
 * un type générique, comme ceci :
 * JComboBox<String> combo = new JComboBox<String>();
 * ce qui permet de mieux gérer le contenu de nos listes et
 * ainsi mieux récupérer les valeurs de ces dernières.
 * */
private JList<String> clubs;
```

```
public FenetrePlacement(String nomfen){
super("je m'appelle: "+nomfen);
initComponents();
setSize(400,300);
}

private void initComponents() {
    //initialisation des composants
valider= new JButton("valider");
annuler= new JButton("annuler");
txtlogin= new JTextField("",8);
String[]donnees={"Robotique", "Jeu echec","Logiciels libres","Peinture","Danse", "Musique"};
clubs= new JList<String>(donnees);
login= new JLabel ("Login");
choix= new JLabel ("Choix");
String[] etats={"marié(e)", "célibataire", "veuf(ve)"};
listchoix= new JComboBox<String>(etats);
```

```
//définition des objets utilisés pour placer les composants
GridBagLayout placeur= new GridBagLayout();//objet pour placement des composants
GridBagConstraints contraintes =new GridBagConstraints();//regles de placement
contraintes.insets=new Insets(10,10,10,10);
setLayout(placeur);//utiliser cet objet de placement pour la fenetre
//placement du label login
contraintes.gridx=0; contraintes.gridy=0;

placeur.setConstraints(login, contraintes);
add(login); //ou  add(login,contraintes);

//placement de la zone de saisie
contraintes.gridx=1;
add(txtlogin, contraintes);

//placement de la combox
clubs.setVisibleRowCount(5);
JScrollPane defile= new JScrollPane(clubs);
contraintes.gridx=2;

contraintes.gridwidth=1;
contraintes.gridheight=GridBagConstraints.REMAINDER;
add(defile, contraintes);

//placement du label choix
contraintes.gridx=0; contraintes.gridy=1;
contraintes.gridwidth=1; contraintes.gridheight=1;
add(choix, contraintes);

//placement de la liste de choix
contraintes.gridx=1; contraintes.gridy=1;
add(listchoix, contraintes);

//placement bouton valider
contraintes.gridx=0; contraintes.gridy=2;
add(valider, contraintes);

//placement bouton annuler
contraintes.gridx=1;
add(annuler, contraintes);
}
```

```
public static void main(String[] args) {
JFrame fen = new FenetrePlacement("exemple placement") ;
fen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
fen.setVisible(true) ;
}}
```


un exemple de fenetre
Inscription et mise à jour
Jouer
CLASSEMENT GÉNÉRAL
Bienvenu
Nom
Prenom
Je m'inscris
text
Modifier
Supprimer
Ajouter
Annuler
<<
>>
Adresse

```
public class Fenetre extends JFrame {
private JPanel inscription , jouer, pNord,pCentre,pSud, pane;
private JTextField txtNom , txtPrenom ;
private JTextArea txtAdresse;
private JButton modifier ,supprimer,ajouter,annuler , suivant , precedant;
private JLabel nom , prenom, bonjour, img, copy, adresse;
private JTabbedPane onglets;
private JSplitPane splite;

public Fenetre() {
super("un exemple de fenetre");
setSize(1000,700);
initComponents();
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);}

private void initComponents(){
bonjour= new JLabel("Bienvenu");
bonjour.setForeground(Color.white);
nom = new JLabel("Nom");
nom.setForeground(Color.white);
prenom = new JLabel("Prenom");
prenom.setForeground(Color.white);
adresse= new JLabel("Adresse");
img= new JLabel(new ImageIcon("img/classement.png"));
copy= new JLabel(" text ",new ImageIcon("img/inscri.jpg"), JLabel.CENTER);
txtNom= new JTextField("",10);
txtPrenom= new JTextField("",10);
```

```
onglets = new JTabbedPane(JTabbedPane.TOP) ;
inscription = new JPanel() ;
pane=new JPanel() ;
pane.setBackground(Color.pink);
onglets.add("Inscription et mise à jour",inscription) ;
jouer = new JPanel() ;
onglets.add("Jouer",jouer) ;
splite = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, onglets, pane);
splite.setDividerLocation(500);
add(splite);
inscription.setLayout(new BorderLayout());
pNord = new JPanel();
pNord.setBackground(Color.blue);
pNord.add(img);
pCentre = new JPanel ();
pCentre.setBackground(Color.red);

pSud = new JPanel() ;
pSud.setBackground(Color.white);
pSud.add(copy);
inscription.add(pNord,"North");
inscription.add(pCentre,"Center");
inscription.add(pSud,"South");
```