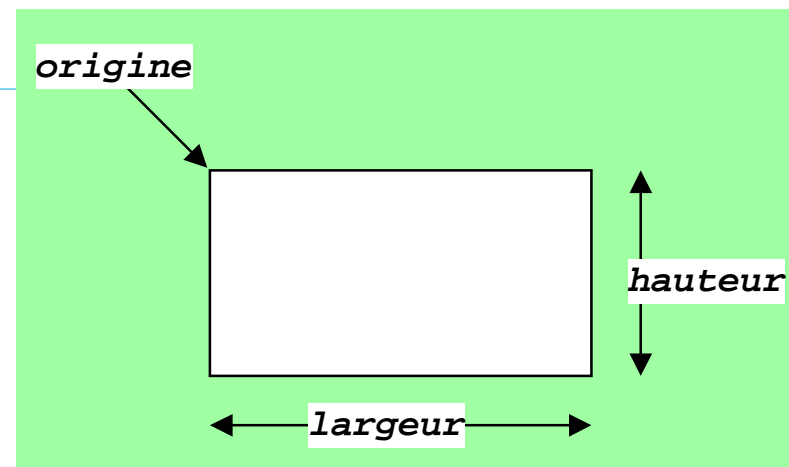


Gestionnaires de géométrie

- ❑ La "géométrie"
- ❑ FlowLayout
- ❑ BorderLayout
- ❑ GridLayout
- ❑ BoxLayout
- ❑ GridBagLayout
- ❑ CardLayout

La géométrie

- Tout composant a un **placement** dans sa mère, donné par son **origine** et ses **dimensions**
- Le **rectangle** est géré par



Rectangle `getBounds()` resp. `setBounds(...)`

- La **position** (origine) est gérée par

Point `getLocation()` resp. `setLocation(...)`

- La **taille** (une **Dimension**) gérée par `getSize()` resp. `setSize(...)`

- Les tailles "idéale", maximale et minimale sont retournées par

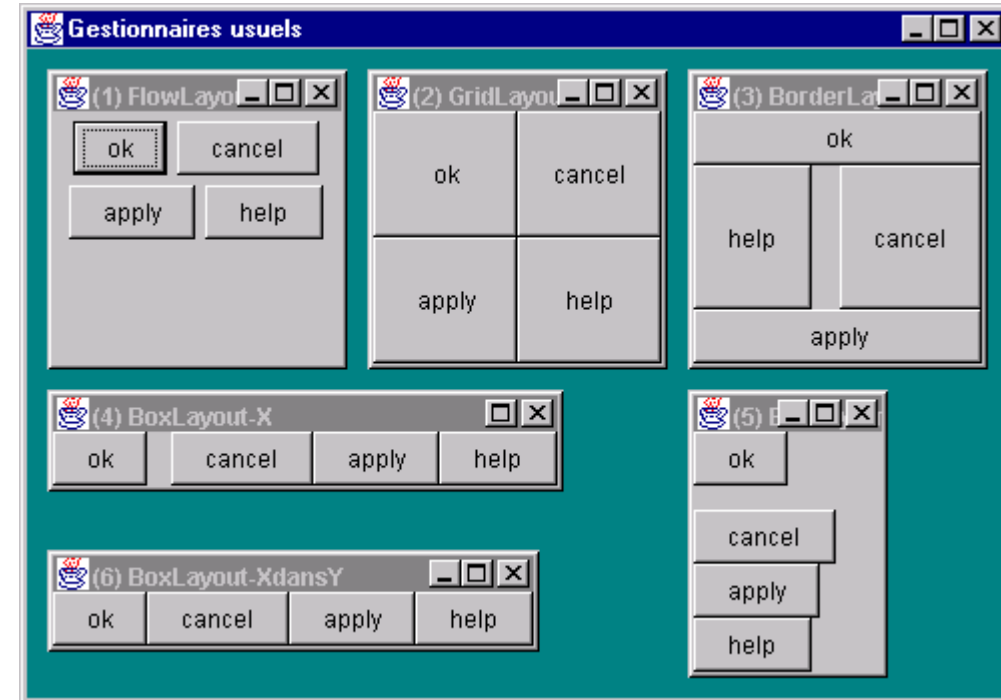
`getPreferredSize()` `getMaximalSize()` `getMinimalSize()`

- Les **gestionnaires de géométrie** utilisent ces informations pour placer les filles dans un conteneur.

Vue d'ensemble

■ Gestionnaires les plus courants:

- **FlowLayout**
- **BorderLayout**
- **GridLayout**
- **BoxLayout**



■ Choix par

`conteneur.setLayout(new XYZLayout())`

■ Plus subtil

GridBagLayout

■ Dépassé

CardLayout

FlowLayout

- **FlowLayout** est le gestionnaire par défaut des **applettes** et des **Panel**. Mais **JApplet** a le gestionnaire **BorderLayout** !
- affiche les composants **de la gauche vers la droite**, et passe à la ligne s'il n'y a plus de place.
- après retaille, la disposition est recalculée.

```
FlowLayout(int align, int hgap, int vgap)  
FlowLayout(int align)  
FlowLayout()
```

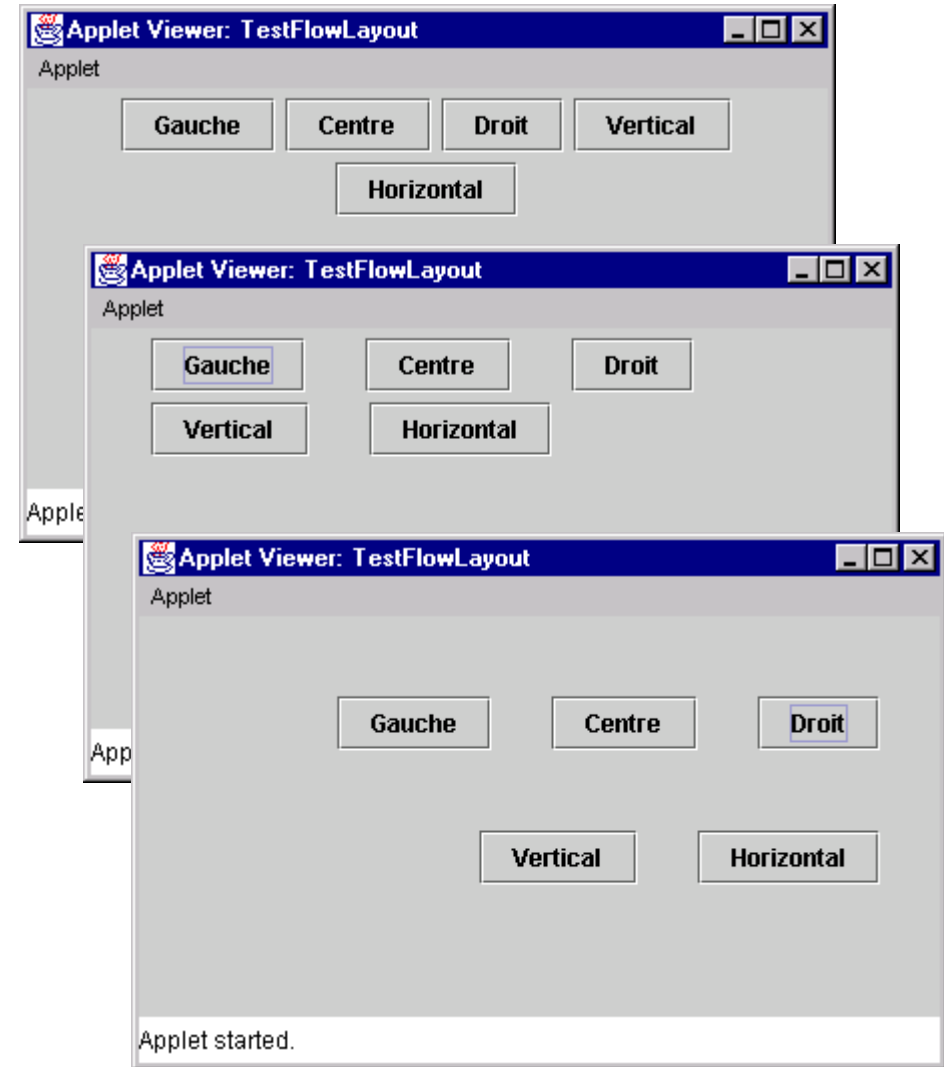
- **align** vaut **LEFT**, **CENTER** (défaut) ou **RIGHT**, **LEADING**, **TRAILING** (par rapport à l'orientation) et indique comment chaque ligne est remplie.
- **hgap** (= 5) et **vgap** (=5) sont les espaces entre composants.
- les marges sont régies par le membre **insets** du conteneur.

Changer les paramètres



TestFlowLayout.bat

- Les paramètres s'obtiennent et se modifient par des méthodes **set** et **get**.
- Le changement ne prend effet
 - qu'après retaille,
 - ou en forçant par **layoutContainer()**.
 - ou en demandant une validation par **revalidate()**.



www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

Exemple

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TestFlowLayout extends JApplet {
    JButton gauche = new JButton ("Gauche");
    JButton centre = new JButton ("Centre");
    JButton droit = new JButton ("Droit");
    JButton espV = new JButton ("Vertical");
    JButton espH = new JButton ("Horizontal");
    JPanel contentPane = (JPanel) getContentPane();

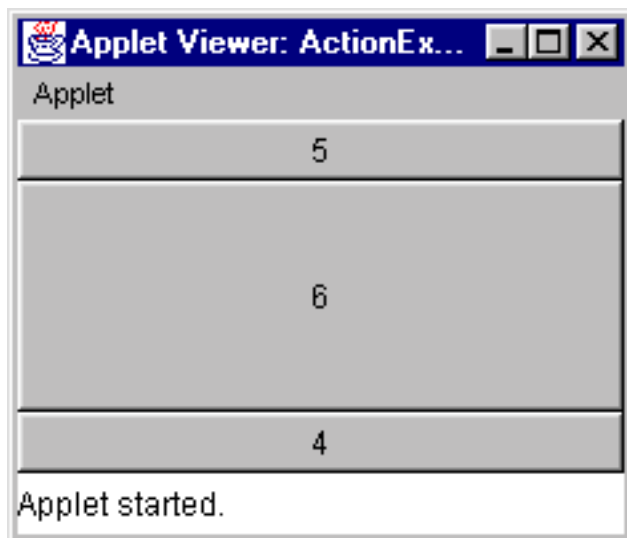
    public void init(){
        contentPane.setLayout(new FlowLayout());
        contentPane.add(gauche);
        contentPane.add(centre);
        contentPane.add(droit);
        contentPane.add(espV);
        contentPane.add(espH);
        gauche.addActionListener(new Aligner(FlowLayout.LEFT));
        centre.addActionListener(new Aligner(FlowLayout.CENTER));
        droit.addActionListener(new Aligner(FlowLayout.RIGHT));
        espV.addActionListener(new Espaceur(1));
        espH.addActionListener(new Espaceur(-1));
    }
    ...
}
```

Exemple (suite)

```
...
class Aigneur implements ActionListener {
    int sens;
    Aigneur(int s) {sens = s;}
    public void actionPerformed(ActionEvent e){
        FlowLayout f = (FlowLayout)contentPane.getLayout();
        f.setAlignment(sens);
        contentPane.revalidate();
    }
} // Aigneur
class Espaceur implements ActionListener {
    int dir;
    Espaceur(int d) {dir = d;}
    public void actionPerformed(ActionEvent e){
        FlowLayout f = (FlowLayout)contentPane.getLayout();
        if (dir == 1)
            f.setVgap(f.getVgap() + 5);
        else
            f.setHgap(f.getHgap() + 5);
        contentPane.revalidate();
    }
} // Espaceur
} // TestFlowLayout
```

BorderLayout

- divise ses composants en 5 régions : **nord**, **sud**, **ouest**, **est**, et **centre**.
- "nord" et "sud" occupent *toute la largeur*,
- "ouest" et "est" occupent *la hauteur qui reste*,
- "centre" occupe la *place restante*.
- On utilise la constante correspondante de **BorderLayout**.



```
import java.awt.*;
import java.applet.Applet;

public class ActionEx extends Applet {
    public void init() {
        setLayout(new BorderLayout());
        add(new Button("1"), BorderLayout.NORTH);
        add(new Button("2"), BorderLayout.NORTH);
        add(new Button("3"), BorderLayout.NORTH);
        add("South", new Button("4")); // old
        add(new Button("5"), BorderLayout.NORTH);
        add(new Button("6"), BorderLayout.CENTER);
    }
}
```


Taille

- **pack()** : les conteneurs prennent la taille *minimale* pour contenir leur composants, en fonction de la taille *préférée* des composants.
- Pour suggérer ou imposer une autre taille, on retaille par
setSize(largeur, hauteur)
setPreferredSize(dimension)
- Ou encore, on réécrit la méthode **getPreferredSize()**, par exemple

```
public Dimension getPreferredSize() {  
    return new Dimension(100,150);  
}
```
- Nécessaire pour un **Canvas**, dont la taille est *nulle* par défaut.

Marges

- Les marges sont entre les bords et les filles.
- L'ensemble des marges est de la classe **Insets** :
 - ◆ quatre champs
 - ◆ constructeur

Insets(top, left, bottom, right)

- Les marges comprennent le bord, et pour **Frame**, aussi la barre de titre !
- Pour changer de marges, on redéfinit la méthode

getInsets()



GridLayout

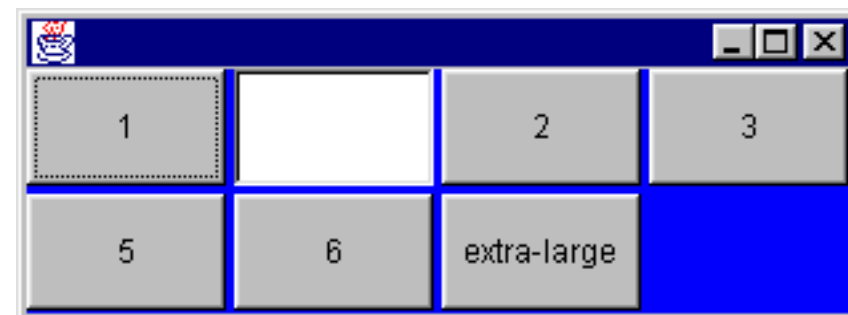
```
GridLayout(int lignes, int cols, int hgap, int vgap)
```

```
GridLayout(int lignes, int cols)
```



- composants sur une *grille*, ligne par ligne (dans l'ordre d'adjonction),
- les cellules ont la *même taille*,
- à la retaille, les cellules se taillent,
- **hgap** et **vgap** sont nuls par défaut.

- On fixe le nombre de lignes *ou* de colonnes,
- *si* `lignes > 0`, *alors* `cols` est ignoré,
- *si* `lignes = 0`, *alors* `lignes` est ignoré.





Package

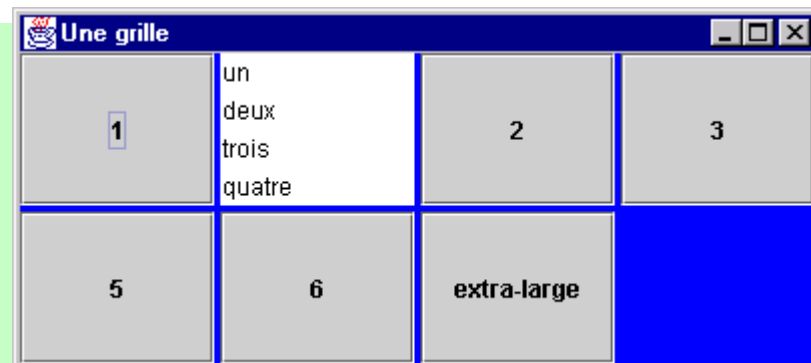
GridLayout (exemple)

```

class Grille extends JFrame {
    public Grille() {
        super("Une grille");
        JPanel c = (JPanel) getContentPane();
        c.setLayout(new GridLayout(2,0,3,3));
        c.setBackground(Color.blue);
        c.add(new JButton("1"));
        String[] noms = {"un","deux", "trois", "quatre"};
        c.add(new JList(noms));
        c.add(new JButton("2"));
        c.add(new JButton("3"));
        c.add(new JButton("5"));
        c.add(new JButton("6"));
        c.add(new JButton("extra-large"));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setVisible(true);
    }

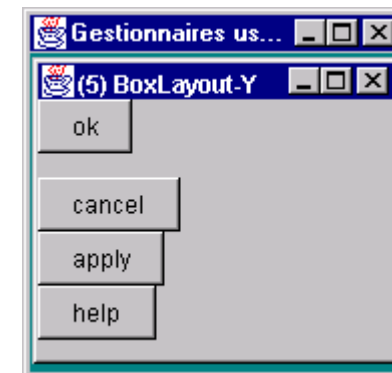
    public static void main(String [] args) {
        new Grille();
    }
}

```



BoxLayout

- Arrange ses filles en une *ligne* ou une *colonne*, en *respectant* la taille préférée de ses filles.
- On peut ajouter
 - des “*struts*”, blocs de taille fixe
 - de la *glue* qui fait du remplissage
 - des *aires* rigides
- Le composant **Box** est un conteneur
 - dont le gestionnaire est **BoxLayout**
 - qui est enfant de **Container**, donc n’est pas dans **Swing**.
 - mais il est transparent.



```
Box b = new Box(BoxLayout.Y_AXIS);
b.add(new JButton("ok"));
b.add(Box.createVerticalStrut(12));
b.add(new JButton("cancel"));
```

```
Container c = new Container();
c.setLayout(new BoxLayout(c, BoxLayout.X_AXIS));
```

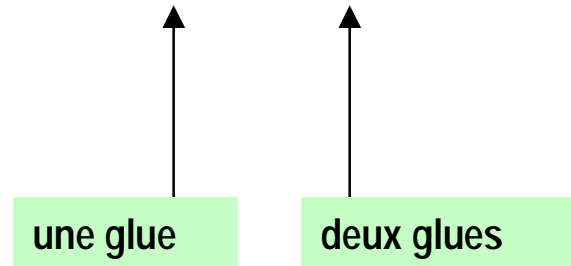


- La glue est en fait un ressort.
- Tout espace supplémentaire est absorbé par la glue.
- Plusieurs glues se partagent l'espace libre.

```
Container c = new Container();  
c.setLayout(new BorderLayout(c, BorderLayout.X_AXIS));  
c.add(Box.createHorizontalGlue());
```

- on centre en mettant une glue des deux côtés
- on maintient aux bords en insérant une glue au milieu

- Deux glues valent mieux qu'une : elle absorbent le double de place.



GridBagLayout

- **GridBagLayout** répartit ses filles *dans une grille*, selon les désidérata exprimées par chaque fille.
- Chaque fille s'exprime dans un objet **GridBagConstraints**.
- Les contraintes sont ajoutées avec la fille.
- Les "contraintes" d'une fille concernent
 - la *zone* réservée dans la grille
 - la *place* occupée dans cette zone

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com



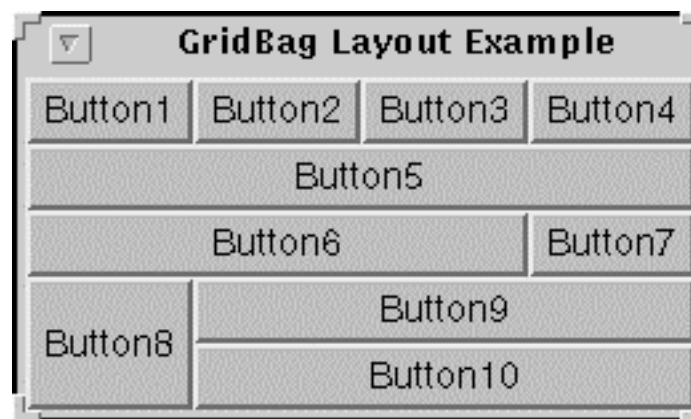
GridBagConstraints

```
GridBagLayout gbl = new GridBagLayout();
container.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();
gbc.anchor      = GridBagConstraints.NORTH;
gbc.gridwidth  = GridBagConstraints.REMAINDER;

Label title    = new Label("Bonjour");

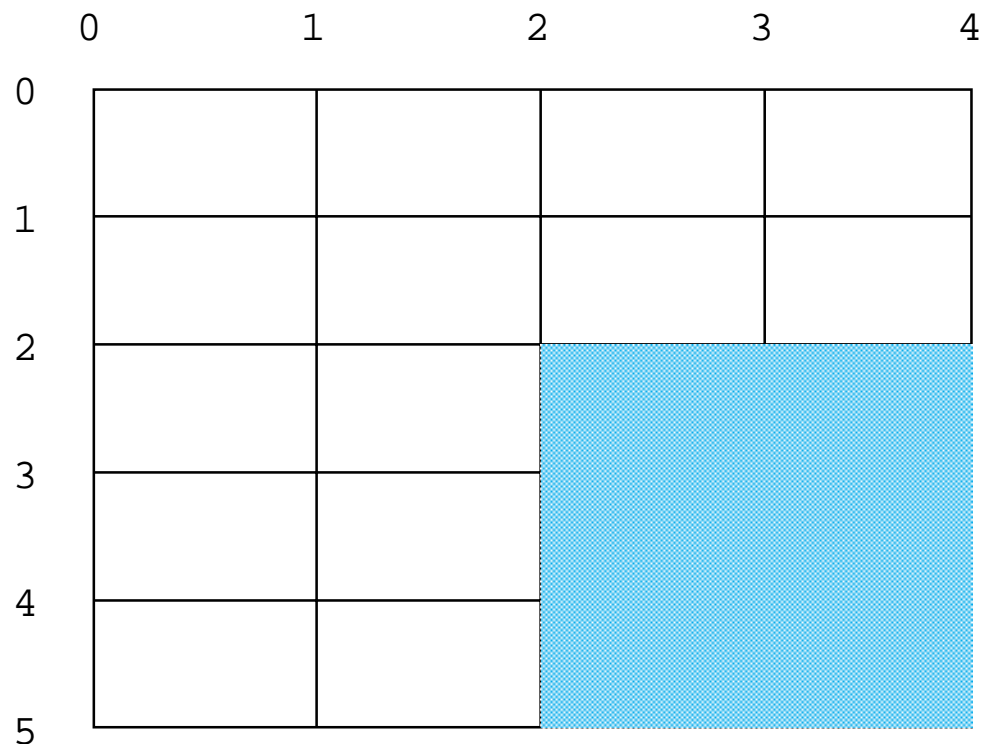
add(title, gbc); // ajout de title selon les contraintes gbc
...
```



Zone d'un composant

- Chaque fille réserve une *zone* dans la grille par son origine et sa dimension.
- **gridx = 2, gridy = 2, gridwidth = 2, gridheight = 3**
- Défaut : **width = height = 1** .

- Si **gridx = RELATIVE**,
l'origine est à droite du précédent.
- Si **gridwidth = REMAINDER**,
la zone qui reste dans la ligne



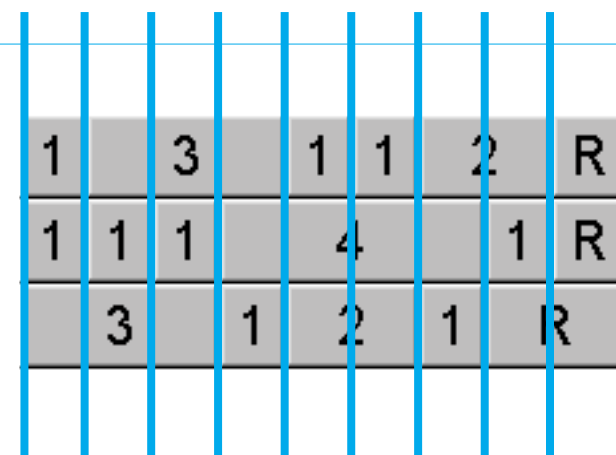
Exemple (gridwidth)

```
import java.awt.*;
import java.applet.Applet;

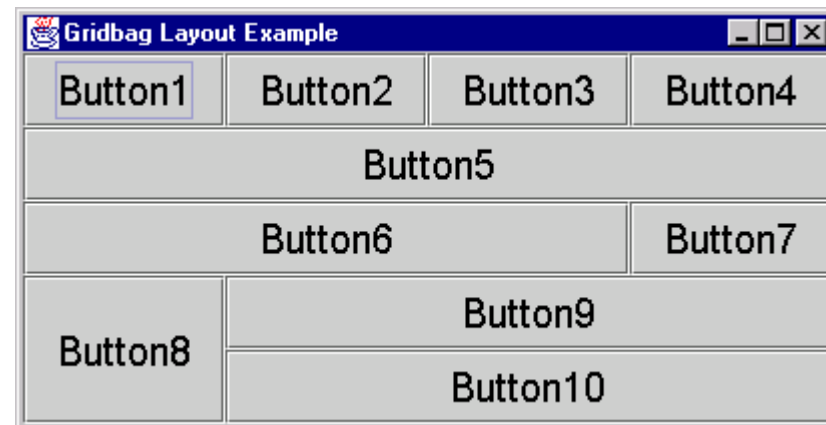
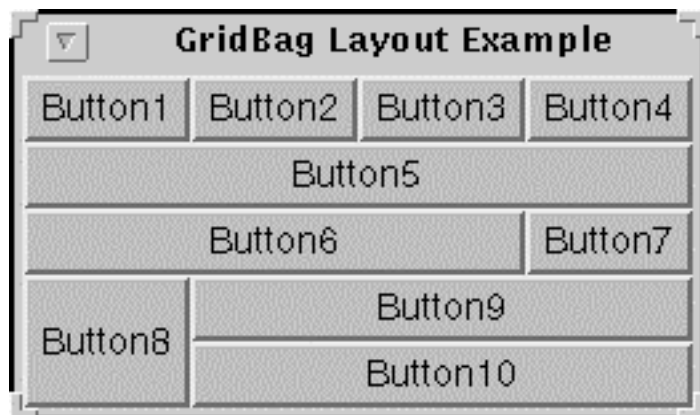
public class GB2 extends Applet {
    GridBagLayout gbl = new GridBagLayout();

    void mB(String nom, int larg) {
        GridBagConstraints gbc = new GridBagConstraints();
        Button b = new Button(nom);
        gbc.fill = GridBagConstraints.BOTH;
        gbc.gridwidth = larg;
        add(b, gbc);
    }

    public void init() {
        setFont(new Font("Helvetica", Font.PLAIN, 20));
        setLayout(gbl);
        mB("1", 1); mB("3", 3); mB("1", 1);
        mB("1", 1); mB("2", 2);
        mB("R", GridBagConstraints.REMAINDER);
        mB("1", 1); mB("1", 1); mB("1", 1);
        mB("4", 4); mB("1", 1);
        mB("R", GridBagConstraints.REMAINDER);
        mB("3", 3); mB("1", 1); mB("2", 2); mB("1", 1);
        mB("R", GridBagConstraints.REMAINDER);
    }
}
```



Un premier exemple



```
public class GridbagDoc extends JFrame {
    JPanel p = (JPanel) getContentPane();

    protected void mb(String name, GridBagConstraints c) {
        p.add(new JButton(name), c);
    }
    GridbagDoc() {
        super("Gridbag Layout Example");
        GridBagConstraints c = new GridBagConstraints();
        p.setLayout(new GridBagLayout());
        . . .
    }
}
```

Exemple (fin)



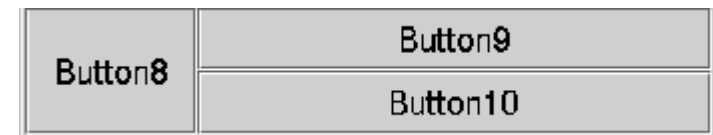
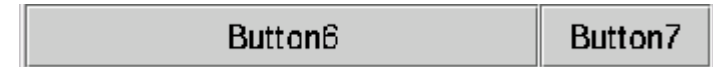
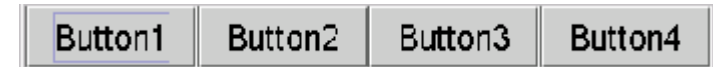
Gridbagdoc.bat

```
c.fill = GridBagConstraints.BOTH;  
c.weightx = 1.0; //se partagent place libre  
mb("Button1", c); mb("Button2", c); mb("Button3", c);  
c.gridwidth = GridBagConstraints.REMAINDER;  
mb("Button4", c); //fin de ligne
```

```
mb("Button5", c);
```

```
c.gridwidth = GridBagConstraints.RELATIVE;  
mb("Button6", c); // sauf dernier  
c.gridwidth = GridBagConstraints.REMAINDER;  
mb("Button7", c); //fin de ligne
```

```
c.gridwidth = 1; c.gridheight = 2;  
c.weighty = 1.0; // 8 prend place verticale libre  
mb("Button8", c);  
c.gridwidth = GridBagConstraints.REMAINDER;  
c.gridheight = 1; // valeur par default  
mb("Button9", c); // termine la ligne  
// Ligne 5 terminee par bouton 10  
c.weighty = 0.0; // renonce à place verticale  
mb("Button10", c);
```





Gridbags.bat

Table des paramètres

- 1,4,5,7,8,10 ont `fill = BOTH`
- 2 est ancré à `NORTH`, 3 au centre,
- 6 est ancré à `WEST`, 9 à `EAST`
- pour 3,6,9,10, `gridwidth = REMAINDER`
- `weightx=2` pour 7, =1 en pour 8 et 9
- `weighty=1`.



<code>anchor</code>	<code>CENTER</code>	<code>CENTER EAST NORTH NORTHEAST, ... WEST</code>	où ancrer la fille dans sa zone
<code>fill</code>	<code>NONE</code>	<code>BOTH, HORIZONTAL, VERTICAL, NONE</code>	remplissage, par la fille, de sa zone
<code>gridx, gridy</code>	<code>RELATIVE</code>	ou entiers	position en absolu, ou à gauche resp. en dessous
<code>gridwidth, gridheight</code>	<code>1, 1</code>	<code>RELATIVE, REMAINDER, ou entiers</code>	largeur et hauteur de la zone, en absolu, à gauche, en dessous, ou ce qui reste sur la ligne (colonne)
<code>weightx, weighty</code>	<code>0, 0</code>	doubles	pois donné à la colonne (ligne), l'élargit en conséquence.



Commander.bat

Un deuxième exemple

```

JLabel titre    = new JLabel("Achetez maintenant !");
JLabel nom      = new JLabel("Nom:");
JLabel adresse  = new JLabel("Adresse:");
JLabel paiement = new JLabel("Mode de paiement:");
JLabel ville    = new JLabel("Ville:");
JLabel code     = new JLabel("Code:");

JTextField nomField      = new JTextField(30);
JTextField adresseField  = new JTextField(30);
JTextField villeField    = new JTextField(15);
JTextField codeField     = new JTextField(5);

JComboBox modePaiement  = new JComboBox();

```

```

public Commander() {
    JPanel p = (JPanel) getContentPane();
    p.setLayout(new GridBagLayout());
    p.setBackground(Color.yellow);
    GridBagConstraints gbc = new GridBagConstraints();
    ...
}

```

Exemple (suite)

■ Titre

```
titre.setFont(new Font("Times-Roman",
    Font.BOLD + Font.ITALIC, 16));
gbc.anchor      = GridBagConstraints.NORTH;
gbc.gridwidth   = GridBagConstraints.REMAINDER;
gbc.insets      = new Insets(5,0,10,0);
p.add(titre,gbc);
```

■ Nom et adresse

```
gbc.anchor      = GridBagConstraints.WEST;
gbc.insets      = new Insets(0,5,0,5);
gbc.gridwidth   = 1;
p.add(nom,gbc);

gbc.gridwidth   = GridBagConstraints.REMAINDER;
p.add(nomField,gbc);

gbc.gridwidth   = 1;
p.add(adresse, gbc);

gbc.gridwidth   = GridBagConstraints.REMAINDER;
p.add(adresseField,gbc);
```

■ Ville et Etat

```
gbc.gridwidth   = 1;
p.add(ville,gbc);
p.add(villeField,gbc);
p.add(code, gbc);
gbc.gridwidth   =
    GridBagConstraints.REMAINDER;
p.add(codeField, gbc);
```

Exemple (fin)

■ Et pour payer

```
gbc.gridwidth = 1;
p.add(paiement,gbc);
gbc.insets = new Insets(5,5,5,0);
gbc.gridwidth = GridBagConstraints.REMAINDER;
gbc.fill      = GridBagConstraints.NONE;
// anchor est WEST depuis le début
modePaiement.addItem("Visa");
modePaiement.addItem("MasterCard");
modePaiement.addItem("COD");
p.add(modePaiement,gbc);
```

Achetez maintenant !

Nom: Dubois

Adresse: 5, bd Descartes

Ville: Marne-la-Vallée Code: 77454

Mode de paiement: MasterCard ▼

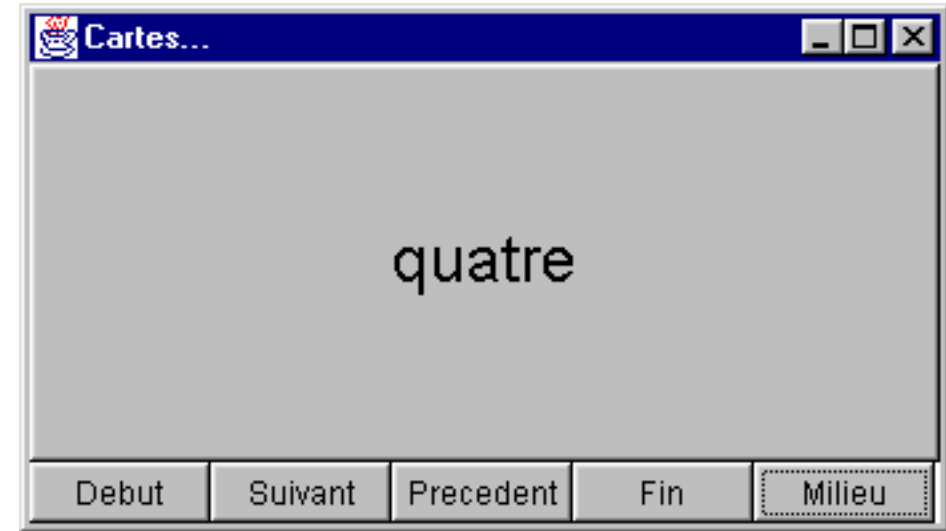
CardLayout



ExCard.bat

Un **CardLayout** organise des composants en une *séquence de fiches*

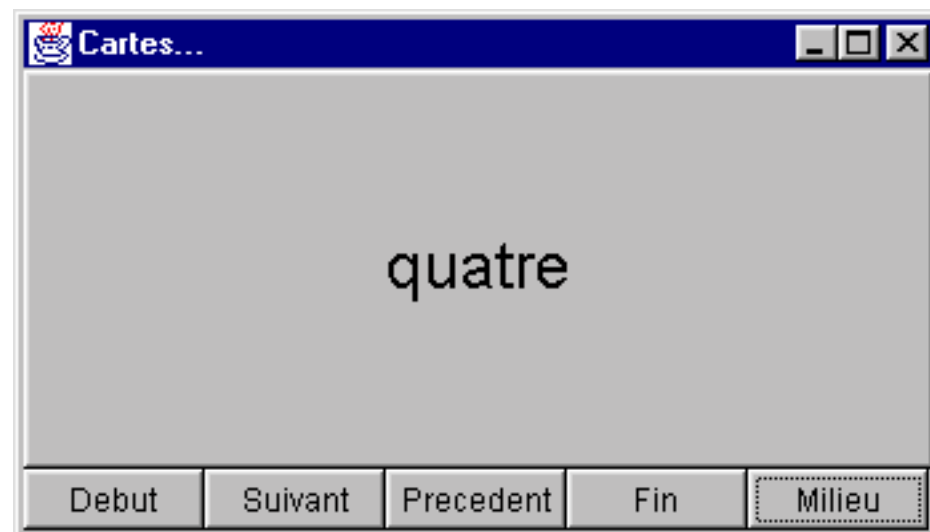
- Une seule fiche est visible à la fois
- Une méthode d'insertion : **add()**
- Des méthodes de parcours: **first()**, **next()**, **previous()**, **last()**
- Une méthode d'affichage d'une fiche nommée : **show(nom)**.
- **JTabbedPane** est plus plaisant.



```
class ExCard extends Frame {
    CardLayout index = new CardLayout();
    // boite a fiches :
    Panel boite = new Panel(index);
    // boutons de gestion
    Button deb, suiv, prec, der, show;
    ...
}
```

Exemple (début)

- La **boite** est formée de 6 fiches.
- Chaque fiche est repérée par un *nom* :
"Une" , "Deux" ,..., "Six"
- Une fiche est (ici) simplement constituée d'un bouton.
- Ici, on voit la fiche de nom "**Central**"



```
ExCard() {
    super("Cartes...");
    boite.setFont(new Font("Helvetica", Font.PLAIN, 24));
    boite.add("Une", new Button("une"));
    boite.add("Deux", new Button("deux"));
    boite.add("Trois", new Button("trois"));
    boite.add("Central", new Button("quatre"));
    boite.add("Cinq", new Button("cinq"));
    boite.add("Six", new Button("six"));
    ...
}
```

Exemple (fin)

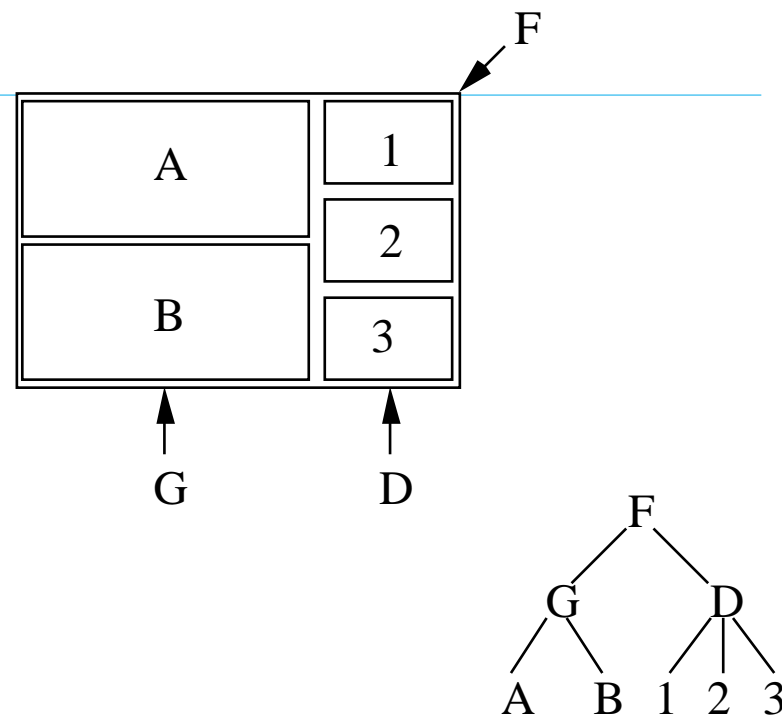
- Les *boutons de gestion* sont groupés dans un panneau **commandes**.
- Les 5 boutons sont tous associés au même auditeur.
- La gestion des fiches est déléguée au *gestionnaire de géométrie* **index**

```
ExCard() {...
    Panel commandes = new Panel(new GridLayout(1,0));
    commandes.setFont(
        new Font("Helvetica", Font.PLAIN, 12));
    commandes.add(deb = new Button("Debut"));
    deb.addActionListener(this);
    commandes.add(suiv = new Button("Suivant"));
    suiv.addActionListener(this);
    commandes.add(prec = new Button("Precedent"));
    prec.addActionListener(this);
    commandes.add(der = new Button("Fin"));
    der.addActionListener(this);
    commandes.add(show = new Button("Milieu"));
    show.addActionListener(this);
    ...
}
```

```
public void actionPerformed (ActionEvent e) {
    Button b = (Button) e.getSource();
    if (b == deb) index.first(boite);
    else if (b == suiv) index.next(boite);
    else if (b == prec) index.previous(boite);
    else if (b == der) index.last(boite);
    else if (b == show) index.show(boite,"Central");
}
```

Validate (AWT)

- Un composant est *valide* ou *invalidé* (on obtient l'état par `isValid()`)
- Un composant invalidé est réorganisé (et réaffiché) à la prochaine retaille, ou par :
 - `validate()`
 - réorganise un composant invalidé, et le marque comme valide;
 - ne fait rien sur un composant valide (même si un descendant est invalidé)
 - `invalidate()`
 - marque comme invalidé le composant *et tous ses ascendants*



- Si le composant 3 grandit, il est marqué invalidé
- `validate` sur F ou D ne fait rien
- `invalidate` de 3 parque F et D comme invalidé

Remarque

`Component.validate()` appelle `Component.doLayout()`

`Component.doLayout()` appelle `getLayout().layoutContainer()` du gestionnaire

Revalidate

- L'appel de `revalidate()` provoque
 - la remontée de l'arbre d'instance à la recherche du premier conteneur pour lequel `isValidateRoot()` est vrai.
 - Le *sous-arbre* de *ce conteneur* est réaffiché (après distribution des événements en attente).
- Pour `JRootPane`, `JScrollPane`, et `JTextField`, `isValidate` retourne vrai par défaut.

