Java Swing

Organisation

- **Introduction à Swing**
- **Composants et Conteneurs**
- Gestion de la Mise en Page
- Gestion des Evénements et Ecouteurs
- Modèles
- Gestion des Dialogues
- Menus
- Outils Graphiques

Qu'est-ce que Swing ?

- Partie des Classes de Fondation Java (Java Foundation Classes)
- Qui dit Java, dit nécessité de comprendre
  - Classes/Objets
  - Héritage
  - Interfaces
  - Polymorphisme…
- Swing
  - 'Look' and 'Feel'
    - Look: couleurs, formes, disposition
    - Feel: comportement des éléments dynamiques (boutons, boîtes, menus)
  - Accessibilité
  - 'Drag' and 'Drop'
- Utilisé pour développer des applications autonomes, mais également des 'applets' et 'servlets'

Débuts avec Swing

- Java Swing Trail

  java.sun.com/docs/books/tutorial/uiswing

- Compilation et Exécution
  - Swing est standard dans Java 2 (JDK >= 1.2)
  - Utiliser NetBeans (installé sur les ordis de l'IUT)
- Swing, comme le reste des éléments de l'API Java, consiste en plusieurs paquetages :

  javax.swing, javax.accessibility, javax.swing.border…
- Au début de votre code – toujours
  - import javax.swing;
  - import javax.swing.event;
- Beaucoup de programmes Swing requièrent également
  - import java.awt.*;
  - import java.awt.event.*;

Différences entre Swing et AWT

- Ne jamais mélanger les composants Swing avec les composants AWT
- Si vous connaissez AWT, ajoutez un ‘J’ devant tout!
  - Ex. AWT : `Button` VS. Swing : `JButton`
- Swing fait tout ce que fait AWT mais en mieux et il y a en plus…
  - Boutons et labels peuvent afficher des images
  - Comportement et apparence des composants
  - Forme des composants
  - Bordures
  - ‘Look and feel’ de l’IHM du programme

Un Programme Swing Typique

- Consiste en :
  - Conteneurs
  - Composants
  - Evénements

Conteneurs et Composants

- Composants
  - Les ‘briques’ de base
  - Variété d’utilisations et de complexités
- Conteneurs
  - Le ‘ciment’
  - Organisation hiérarchique

Conteneurs

- Eléments de base d'une application graphique
  - Pas possible de créer une application graphique sans conteneur pour maintenir le reste des éléments
- Types de conteneurs haut-niveau
  - **JWindow**
    - Fenêtre la plus basique.
    - Simplement un conteneur affichable à l'écran : pas de barre de titre, pas de boutons de fermeture/redimensionnement.
  - **JDialog**
    - Fenêtre destinée aux boîtes de dialogue.
    - Peut être modale, i.e. bloque une autre fenêtre tant qu'elle est ouverte.
    - Destinées à travailler de pair avec la fenêtre principale
  - **JFrame**
    - Voir transparent suivant
  - **JApplet**

`javax.swing.JFrame`

- Destinée à être la fenêtre principale de l'application
- Fenêtre avec titre et bordure
- JFrame permet les barres de menu
- Possède un bouton de fermeture, un bouton de redimensionnement et un bouton pour l'iconifier
- Création de JFrames
  - **`JFrame frame = new JFrame();`**
  - Ou classe héritant de la classe JFrame (bien mieux!)

Exemple d'utilisation d'une JFrame non "thread-safe" (cf. plus loin)

```
public class JFrameExample {

  public JFrameExample() {
     JFrame f = new JFrame();
     f.setVisible(true);
  }

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

}
```

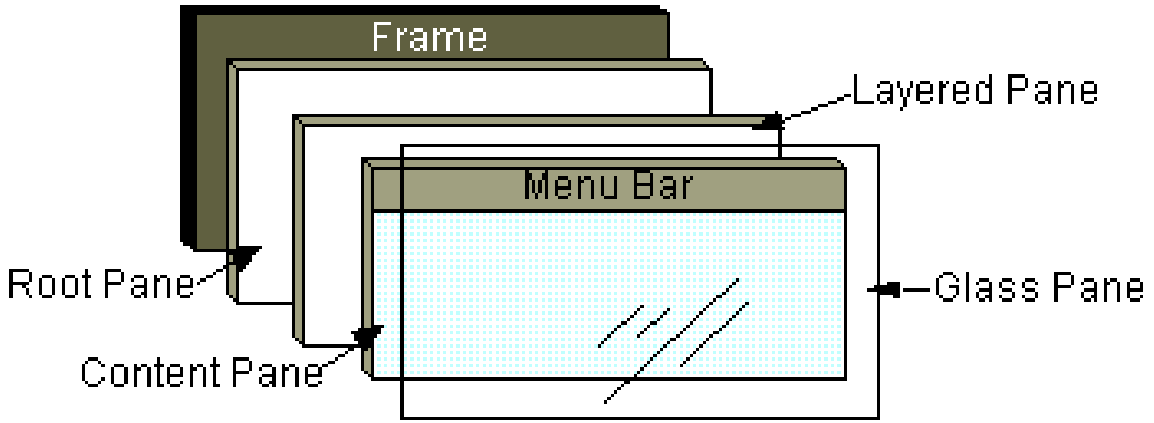
Exemple d'utilisation d'une JFrame

```
import javax.swing.JFrame;

public class SimpleFenetre extends JFrame{

 public SimpleFenetre(){
  super();
 }
}
```

```
import javax.swing.JFrame;

public class SimpleFenetre extends JFrame{

 public SimpleFenetre(){
  super();

  build();//On initialise notre fenêtre
 }

 private void build(){
  setTitle("Ma première fenêtre"); //On donne un titre à l'application
  setSize(320,240); //On donne une taille à notre fenêtre
  setLocationRelativeTo(null); //On centre la fenêtre sur l'écran
  setResizable(false); //On interdit la redimensionnement de la fenêtre

 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //On dit à l'application de se fermer
 }
}
```

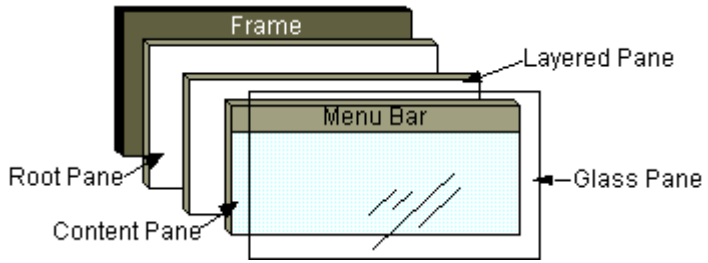
Conteneurs Intermédiaires – Panels (ou ‘panes’)

- *Root pane*
  - *Layered pane*
  - *Content pane*
  - *Glass pane*
    - couche par dessus le tout utilisée pour intercepter les actions de l'utilisateur avant qu'elles ne



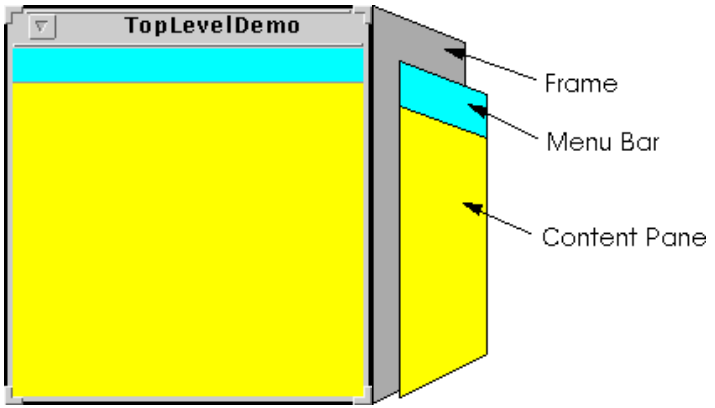
Root pane

- Rattaché de manière ‘invisible’ au conteneur de haut-niveau
- Créé par Swing dès la mise en oeuvre de la fenêtre
- A pour rôle de tout gérer entre le conteneur de haut-niveau et les composants
- Place la barre de menu et le *content pane* dans une instance de *JLayeredPane* (cf. plus loin)

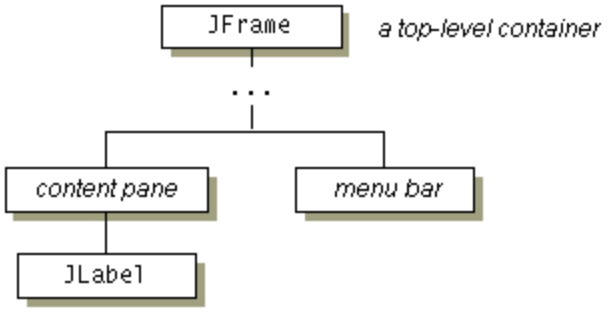


Content pane

- Utilise habituellement un *JPanel*
- Contient tout sauf la barre de menu pour la plupart des applis Swing
- Peut être créé implicitement ou explicitement
  - Cf. code

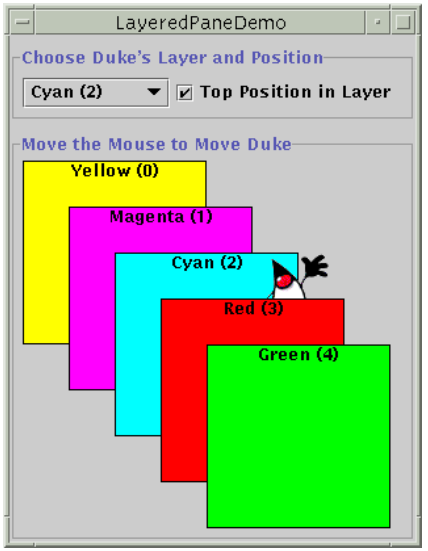



```
//Création d'un JPanel et ajout de composants
JPanel panneau = new JPanel();
panneau.add(unComposant);
panneau.add(unAutreComposant);

//En faire le content pane
panneau.setOpaque(true);
topLevelContainer.setContentPane(panneau);
```



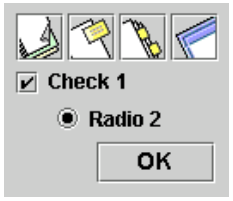
Layered pane

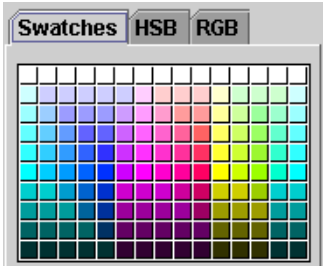
- Fourni par le *root pane*, mais peut aussi être créé
- Forme un panneau composé du *content pane* et de la barre de menu
- Attribue une profondeur aux composants
  - Contenu Cadre (-30000, content pane, barre de menu)
  - Défaut (0, composants)
  - Palette (100, barres d'outils et palettes)
  - Modal (200, dialogues internes)
  - Popup (300, dialogues externes)
  - Drag (400, composant glissé)



Composants

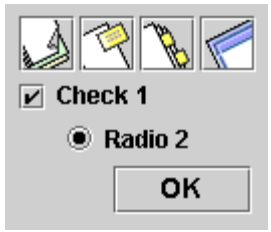
- Contenu de l'IHM
  - http://java.sun.com/docs/books/tutorial/uiswing/components/jcomponent.html
- Objets
  - `JButton bouton_ok = new JButton("OK");`
  - `JTextField txt = new JTextField();`
  - `JLabel lab = new JLabel();`
- Impossible de voir en détail chacun des composants
  - Grande diversité entre simple (e.g. `JButton`) à compliqué (e.g. `JColorChooser`)

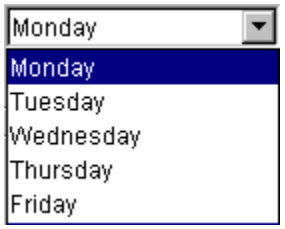


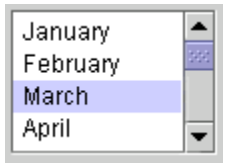


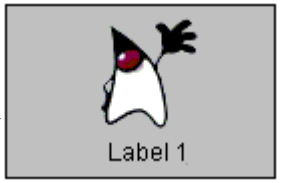
Exemples de Composants

- Bouton
- Combo box
- Liste
- Menu
- Barre glissante
- Champ Texte
- Label





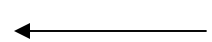




Ajout de Composants

- Une fois le composant créé, il est ajouté au conteneur en utilisant la méthode **`add`**

```
Container cp = getContentPane();
cp.add(new JButton("Annulation"));
cp.add(new JButton("Validation"));
```



requis

L'organisation visuelle à l'écran est déterminée par le gestionnaire de disposition (cf. prochain cours).

Fil Conducteur : Calculatrice

- But : mettre en œuvre une calculatrice simple permettant de faire des additions, soustractions, multiplications et divisions avec 2 nombres.
- Mise en œuvre de l'interface : fenêtre contenant 5 champs
  - Un champ pour saisir le premier nombre
  - Un champ pour saisir le deuxième nombre
  - Une liste pour choisir l'opérateur
  - Un bouton pour effectuer le calcul
  - Un champ pour afficher le résultat du calcul

Calculatrice : Création de la fenêtre principale (1)

- Utilisation d’une JFrame
  - Fixer sa taille à 400 sur 200, on l'adaptera plus tard en fonction des composants.
  - Titre : Calculatrice
  - Centrée à l'écran
  - Non redimensionnable
  - Fermeture de l'application lors de la fermeture de la fenêtre.

Organisation

- **Introduction à Swing**
- **Composants et Conteneurs**
- Gestion de la Mise en Page
- Gestion des Evénements et Ecouteurs
- Modèles
- Gestion des Dialogues
- Menus
- Outils Graphiques

Problématique de la Mise en Page

Comment spécifier l'emplacement d'un composant dans la fenêtre, sa taille ainsi que la manière dont il est géré lors du déplacement/redimensionnement de la fenêtre ?

- Positionnement Absolu
  - Placement des composants en utilisant un système de coordonnées (x,y)
  - Problème : pas pratique
    - Si ajout de composants, changement des coordonnées des autres composants
    - Difficile de faire du contenu redimensionnable avec cette méthode à moins de recalculer les coordonnées de tous les composants à chaque redimensionnement
- Gestionnaire de Placement
  - Place les composants dans la fenêtre en fonction des paramètres donnés et des composants eux-mêmes
  - Plus souple et plus pratique que le positionnement absolu

Gestionnaires de Disposition
Aspects Généraux

- Gestion des composants ajoutés
- Détermine taille et position
- Chaque conteneur a un gestionnaire de disposition (très souvent)
- Problématiques abordées :
  - Types des gestionnaires de disposition
  - Création des gestionnaires de disposition
  - Description des gestionnaires de disposition
  - Critères guidant le choix du gestionnaire de disposition
- **A lire**

  http://java.sun.com/docs/books/tutorial/uiswing/layout/using.html

Typologie des Gestionnaires de Disposition

- BorderLayout
- FlowLayout
- GridLayout
- BoxLayout

- Extra :
  - CardLayout
  - GridBagLayout

Création d'un Gestionnaire de Disposition

- Gestionnaires de disposition par défaut
  - JFrame, JDialog, JApplet considèrent le BorderLayout
  - JPanel a le FlowLayout
    - Sauf lorsqu'il est utilisé comme *Content Pane* (BorderLayout)

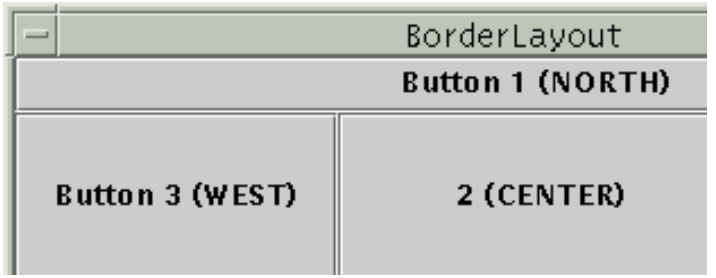
- Mise en place du gestionnaire de disposition pour un conteneur
  - ```
    JFrame frame = new JFrame();
    frame.setLayout(new FlowLayout());
    ```
  - ```
    JPanel contentPane = new JPanel();
    contentPane.setLayout(new BorderLayout());
    ```

Taille Préférée des Composants

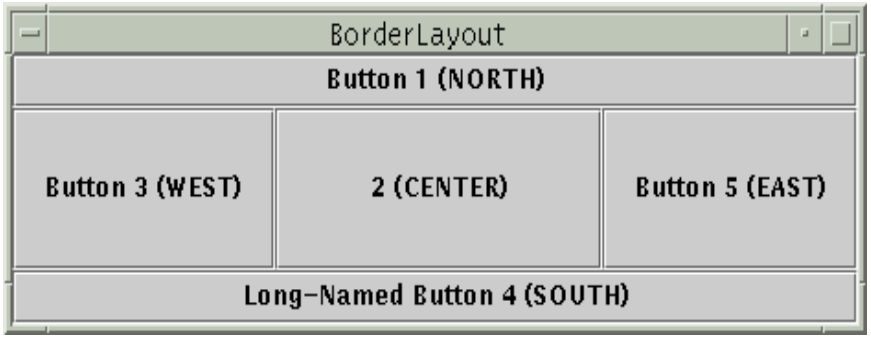
- Les objets composants de Swing ont chacun une taille ‘préférée’ – i.e. une taille qui convient parfaitement à la mise en place de leurs contenus (texte, icônes, etc.)
- Certains types de gestionnaires de disposition (e.g. `FlowLayout`) choisissent d’adopter la taille préférée des composants qu’ils gèrent alors que d’autres (e.g. `BorderLayout`, `GridLayout`) ne la considèrent pas et utilisent un autre procédé

Boutons avec taille ‘préférée’

Pas de taille ‘préférée’



BorderLayout



- Cinq Zones
  - NORTH, SOUTH, EAST, WEST et CENTER
  - Possibilité de ne pas caractériser certaines zones
  - Pas de zone par défaut pour les composants
  - Zone centrale occupe l'espace le plus grand possible
- Spécification de l'emplacement comme paramètre de la méthode d'ajout
  - `pane.setLayout(new BorderLayout());`
  - `pane.add(new JButton("Button 1 (NORTH)"), BorderLayout.NORTH);`
- Mise en place d'espacements entre composants (par défaut, 0)
  - `BorderLayout.setHgap(int gap);`
  - `BorderLayout.setVgap(int gap);`
  - `BorderLayout(int horizontalGap, int verticalGap)`

FlowLayout

```
public FlowLayout()
```

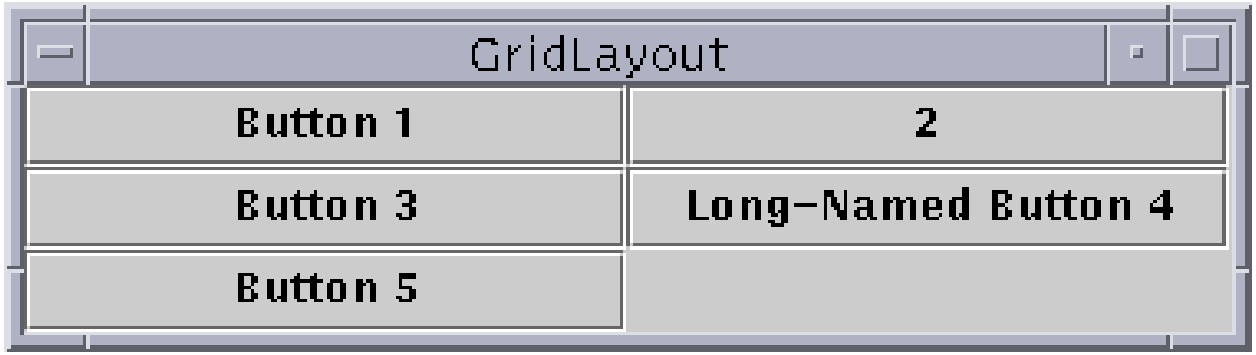
- Gère le conteneur dans le sens gauche-droite, haut-bas
- Attribue aux composants leur taille “préférée”
- Composants positionnés selon leur ordre d’ajout
- Si dépassement potentiel d’une ligne, le composant est disposé en début de ligne suivante

```
Container panel = new JPanel(new FlowLayout());
panel.add(new JButton("Button 1"));
```

GridLayout

```
public GridLayout(int rows, int columns)
```

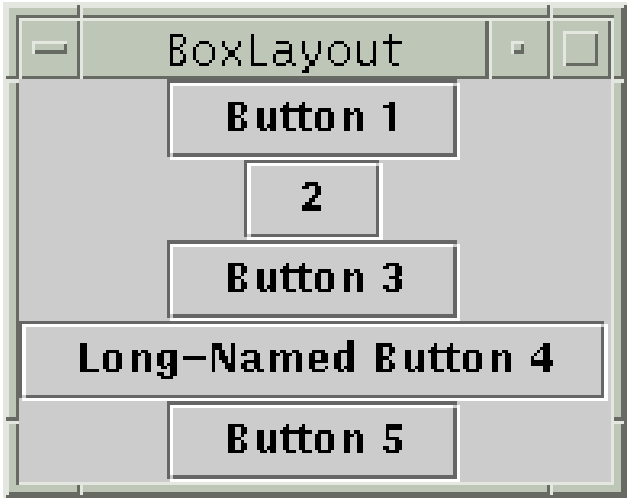
- Gère le conteneur comme une grille de lignes et de colonnes de même taille
- Attribue aux composants la même taille horizontale / verticale, ignorant la taille “préférée”



BoxLayout

- Aligne composants sur une seule ligne ou colonne
- Composants utilisent leur taille “préférée” et sont alignés selon leur alignement préféré
- Construction d’un conteneur avec BoxLayout

```
BoxLayout(panel, BoxLayout.X_AXIS)
BoxLayout(panel, BoxLayout.Y_AXIS)
```



Autres

- `CardLayout`
  couches de "cartes" empilées
  les unes sur les autres;
  une seule visible au temps t

- `GridBagLayout`
  compliqué, à ne pas utiliser

- sur mesure / pas de disposition spécifique
  - Mise en oeuvre de positions absolues en utilisant `setX/Y` et `setWidth/Height`

Méthodes des Gestionnaires de Disposition

- Méthodes n'aboutissant pas sur une nouvelle disposition
  - **`add(), remove(), removeAll()`**
  - **`getAlignmentX(), getAlignmentY()`**
  - **`getPreferredSize(), getMinimumSize(), getMaximumSize()`**

- Méthodes à l'origine d'une nouvelle disposition
  - **`JFrame.pack();`**
    - Force la fenêtre à s'ajuster à la taille préférée et aux dispositions de ses composants
  - **`JFrame.show() & JFrame.setVisible(true);`**
    - Montrent le composant
  - **`JComponent.revalidate();`**
    - Méthode appelée automatiquement sur le composant lors d'une modification. Examine les composants dépendants et appelle la méthode validate() pour ces derniers. Validate() force un conteneur à reconsidérer la disposition de ses composants.

Choix du Gestionnaire de Disposition (1)

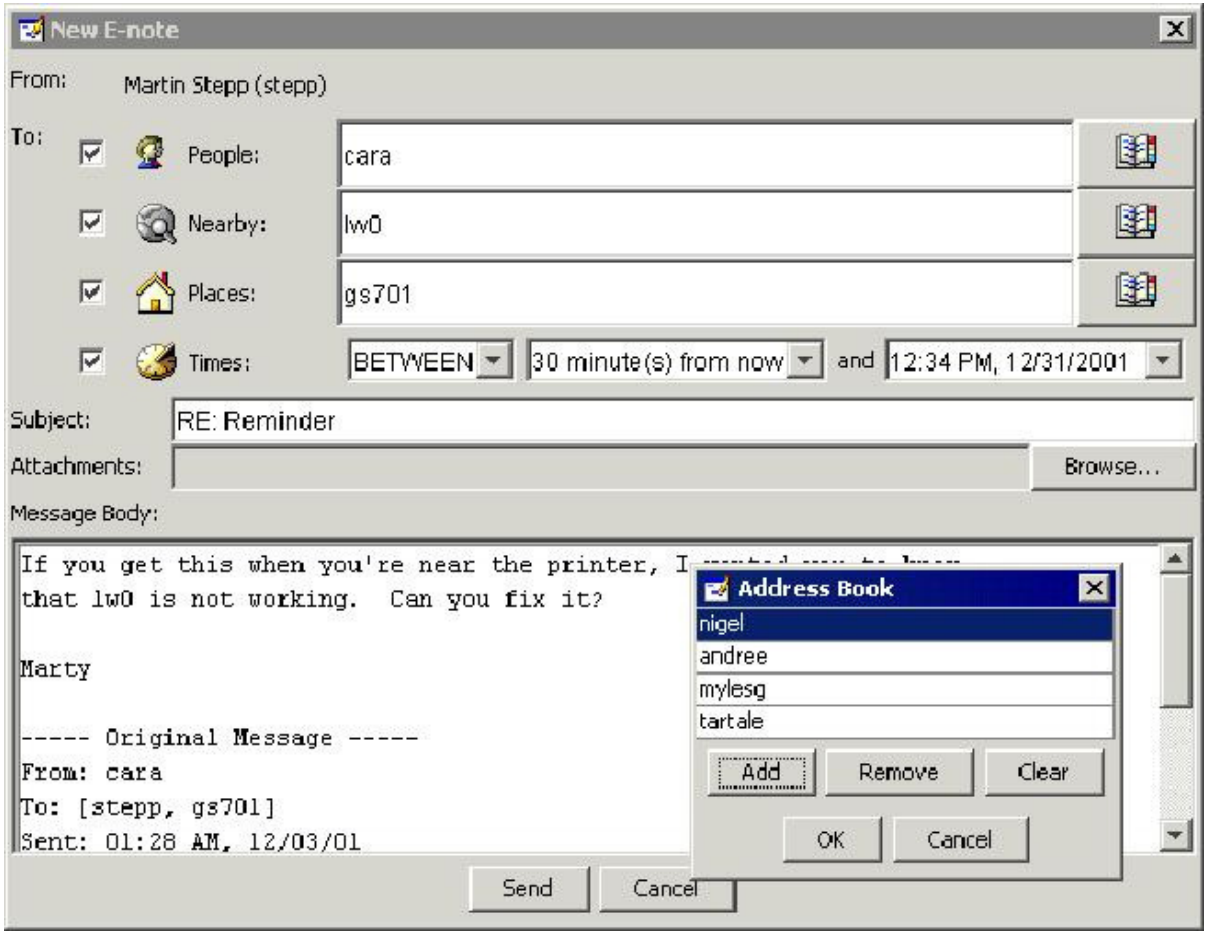
- Affichage d'un composant dans l'espace maximum pouvant lui être alloué
  - BorderLayout
    - Composant dans la zone CENTER
  - BoxLayout
    - Composant spécifie des tailles préférées/maximum très larges
- Affichage de composants sur une ligne compacte
  - FlowLayout
  - BoxLayout
- Affichage de composants de même taille dans des lignes et colonnes
  - GridLayout

Choix du Gestionnaire de Disposition (2)

- Affichage de composants sur une ligne ou colonne, avec différents espacements entre eux et tailles sur mesure
  - BoxLayout
- Affichage d'une disposition complexe avec plusieurs composants…

Problème avec Gestionnaires de Disposition

Comment créer une fenêtre complexe en utilisant les gestionnaires de disposition ?



Solution : Composition de Gestionnaires

- Création de *panels* à l'intérieur de *panels*
- Chacun a une disposition propre et en combinant les dispositions, obtention d'une disposition plus complexe
- Exemple :
  - Combien de *panels* ?
  - Quelle disposition pour chacun ?
  - A réaliser