GUI

Interfaces graphiques en Java

Interfaces graphiques

- Passer d'une console à une métaphore graphique est une révolution en soi
- Mais ça ne suffit pas
- L'interface graphique est la première des IHM, elle doit faire l'objet d'une attention particulière

Philosophie

- Fenêtres hôtes
- Même fonctionnement quelle que soit la plateforme
- Evènements

Fenêtre hôtes

- Frame = Fenêtre de l'OS
- Comportements par défaut
- Composants standards

Création d'une fenêtre

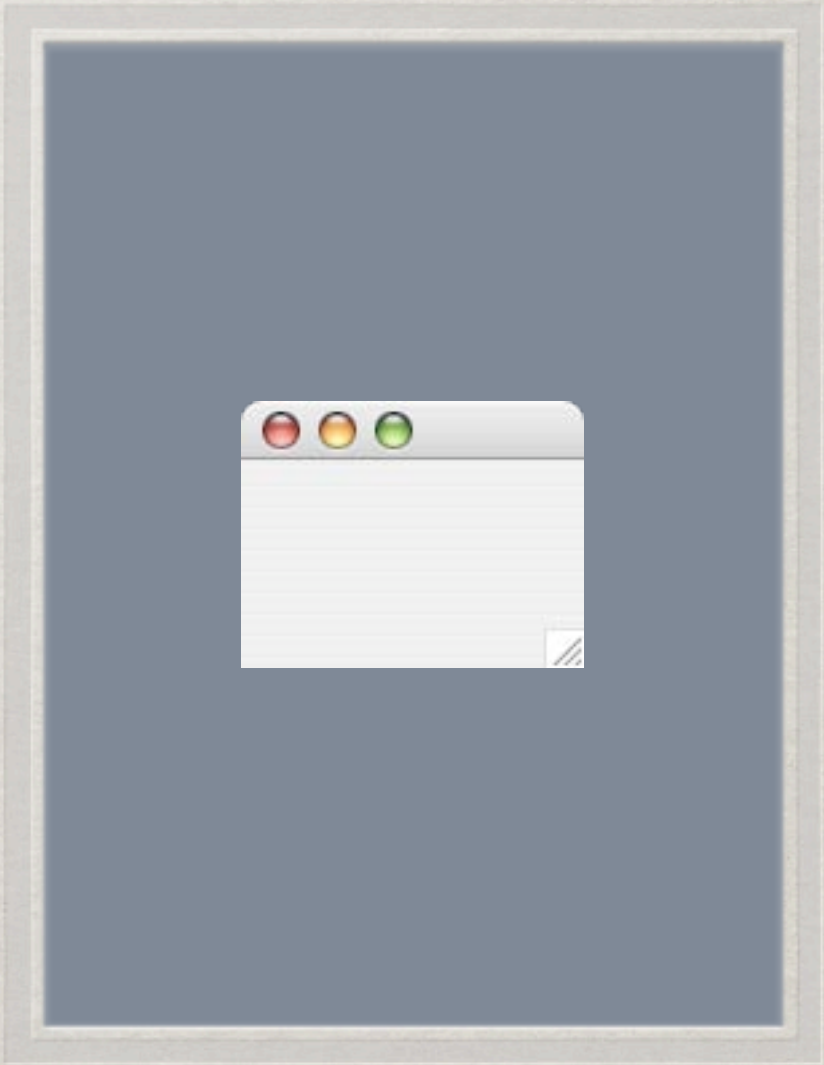
- Une fenêtre est un objet comme les autres
- On le crée avec un new
- Il "marche" par défaut

Exemple

```
Frame f = new Frame();

f.setSize(100,100);

f.setVisible(true);
```



Philosophie objet

- On peut avoir une Frame comme variable d'instance d'une classe (modèle "contrôleur")
- On peut avoir une classe qui hérite de Frame

Différences entre les façons de faire

- Système par contrôleur
  - Permet de gérer plusieurs fenêtres avec un seul objet maître
  - Instanciation dite "externe"
  - Assez rigide

Différences entre les façons de faire

- Système par héritage
  - Permet de répliquer à chaque fois la même fenêtre, avec son comportement
  - Très souple, très objet, mais une seule à la fois

Exemple

```
public class MaFrame extends Frame {
  public MaFrame() {
    super();
    // ajout de mes comportements et éléments graphiques
    this.setVisible(true);
  }

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

Comparatif

Contrôleur	Héritage
Code difficile a maintenir	Code objet simple
Peut gérer plusieurs fenêtres en même temps	Peut gérer elle-même
Difficilement sous-classable	Facilement sous-classable
Adapté à un système avec plusieurs fenêtres pour les mêmes données	Adapté à un système avec des fenêtres au comportement particulier

Solutions plus complexes

- Fenêtres cascadées
  - Chaque fenêtre doit créer et maintenir à jour des "sous-fenêtres"
  - Ex : panneau d'info, représentation graphique, etc...
- Contrôleur de fenêtres "héritées"
  - Chaque fenêtre a un type particulier, le contrôleur les gère
  - "MVC"

Ordre de préférence

- MVC
- Fenêtres cascadées
- Héritage
- Contrôleur

Eléments de choix

- Une seule fenêtre ou plusieurs fenêtres
- Application centrée sur l'interface ou les données
- Evolutions probables du code

Homogénéité

- Comportement “Java” vs Comportement “Système”
- “Look and Feel” spécifique ou système

Programmation évènementielle

- Evènements système (souris, clavier, fenêtres, etc...)
- Asynchronisme (aucune garantie sur les délais)

Composants système

AWT ou Swing?

- AWT et Swing marchent **exactement** de la même manière
- Swing = AWT v2
- AWT = base minimale
- Swing = meilleure intégration au système / moins bonne compatibilité

AWT vs Swing

- Frame vs JFrame
- TextField vs JTextField
- Button vs JButton
- etc... le fonctionnement est quasiment identique
- **Ne pas mélanger les deux pour autant!**

Composants Graphiques

- Component (classe mère de **tous** les composants graphiques)
- Méthodes utiles
  - setSize(width,height)
  - setPosition(x,y)

Fenêtres

- Frame
- Méthodes utiles
  - setLayout(Layout)
  - add(widget)
  - setVisible(boolean)

Sous-Fenêtres

- Panel
- Méthodes utiles
  - pareil

Layouts

- GridLayout, GridBagLayout, BorderLayout, ... (cf plus tard)
- Méthodes utiles
  - aucune

Boutons

- Button
- Méthodes utiles
  - addMouseListener(MouseListener)
  - setLabel(String)

Champs texte

- Label (non-éditable), TextField (une ligne), TextArea (plusieurs lignes)
- Méthodes utiles
  - String getText()
  - setText(String)

Zone de dessin

- Canvas
- Cas particulier : Canvas ne fait rien toute seule. On ***∂oit*** la sous-classer
- Méthodes utiles
  - public void paint(Graphics g)
    On doit la réécrire pour qu'elle fasse des rectangles, ovales, texte, etc...

Design

Questions basiques

- Doit-on quitter quand on ferme la fenêtre?
- Doit-on faire des fenêtres d'alerte?
- Doit-on différencier simple et double clic?
- La réponse dépend du système

Java HIG

- http://java.sun.com/products/jlf/ed2/book/index.html (version officielle)
- versions pour chaque plateforme
- bon sens

Bonnes pratiques

On évite les menus si on n'en n'a pas besoin

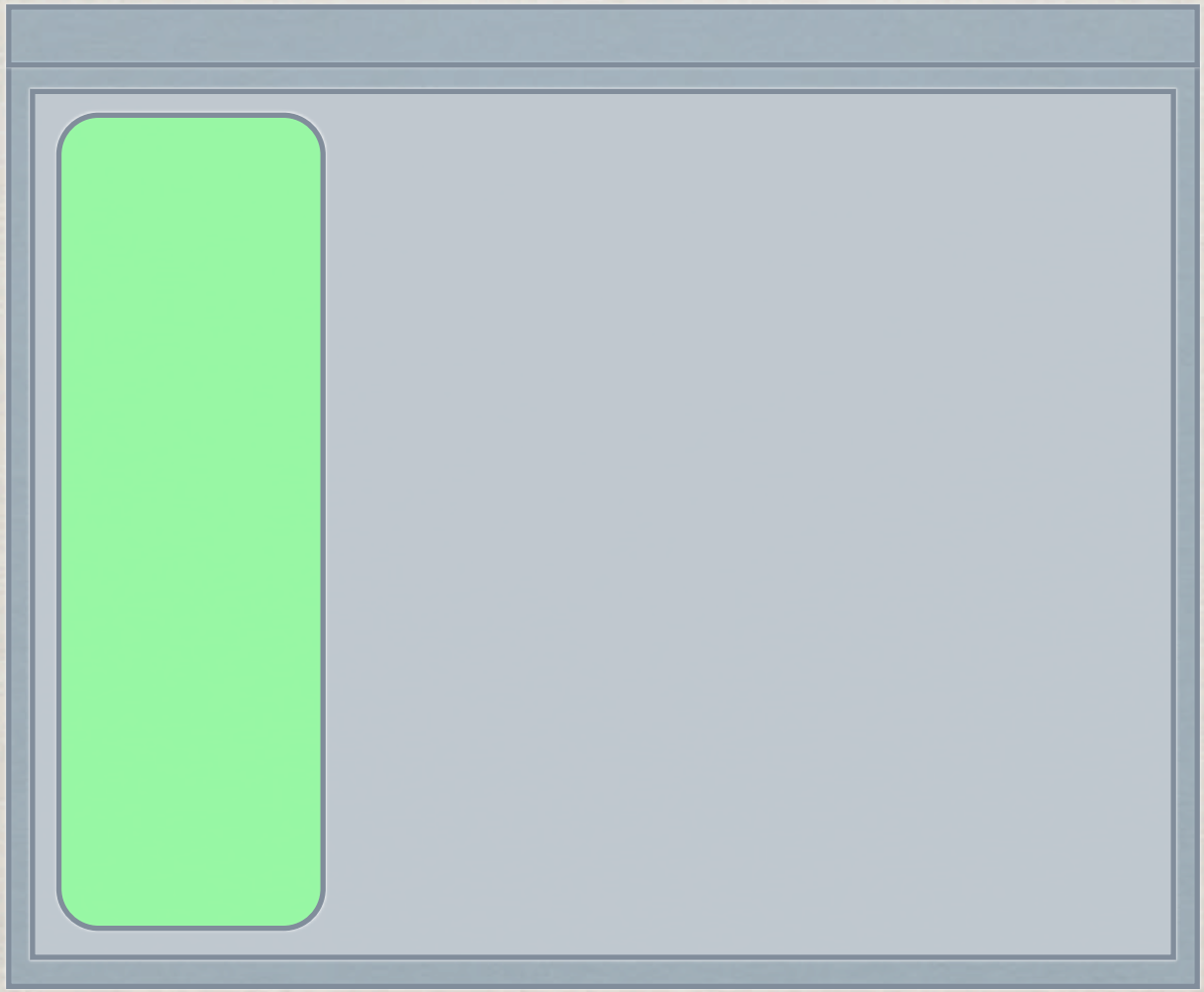
Bonnes pratiques

On laisse une marge de quelques pixels

Solutions

- Manuellement (avec un layout “null”)
- Automatiquement (avec un layout non nul)

Bonnes pratiques

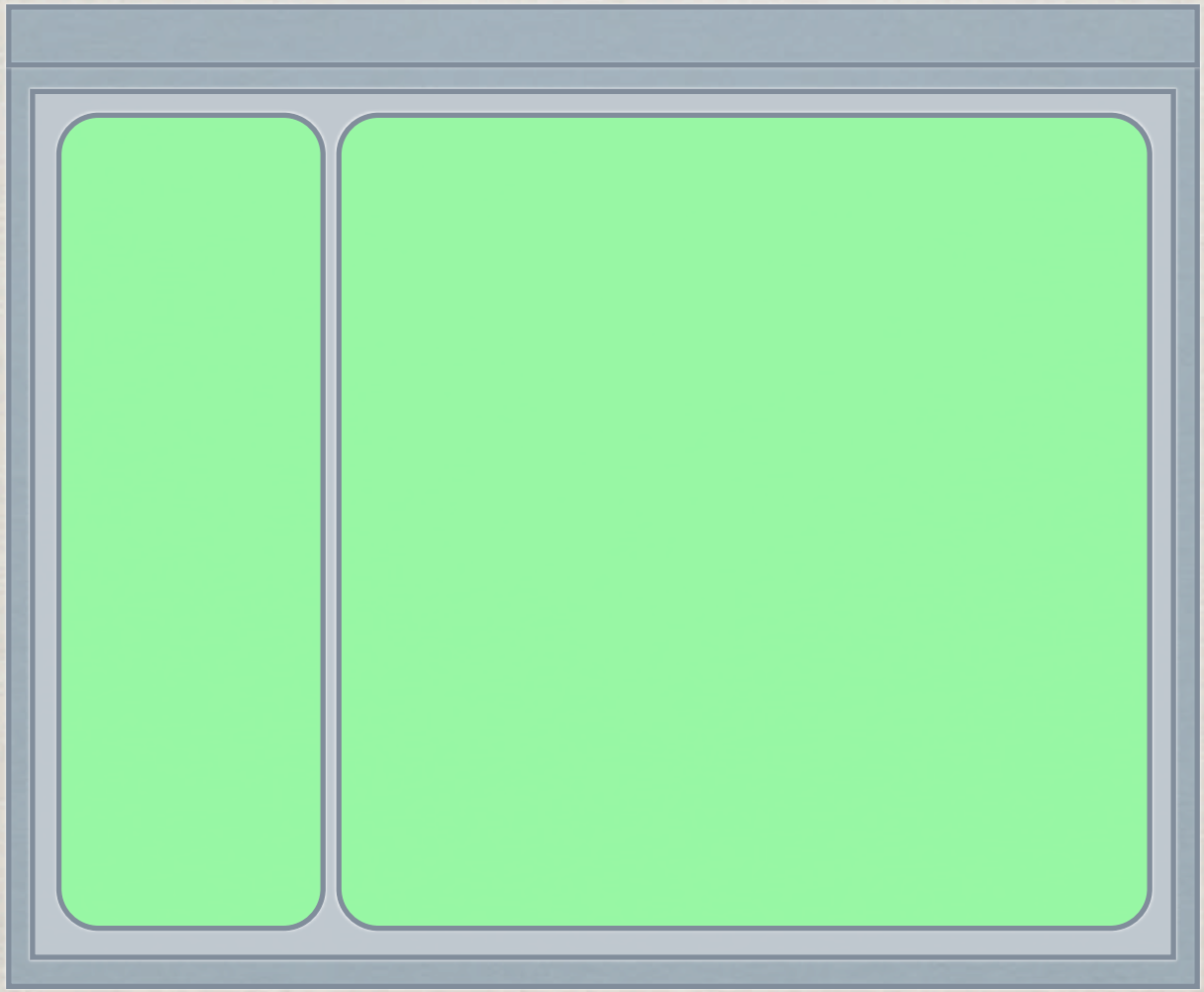


On met les boutons et les infos à gauche, à droite,
en haut ou en bas, mais pas partout autour

Solutions

- Bon sens
- Goûts
- Cible

Bonnes pratiques



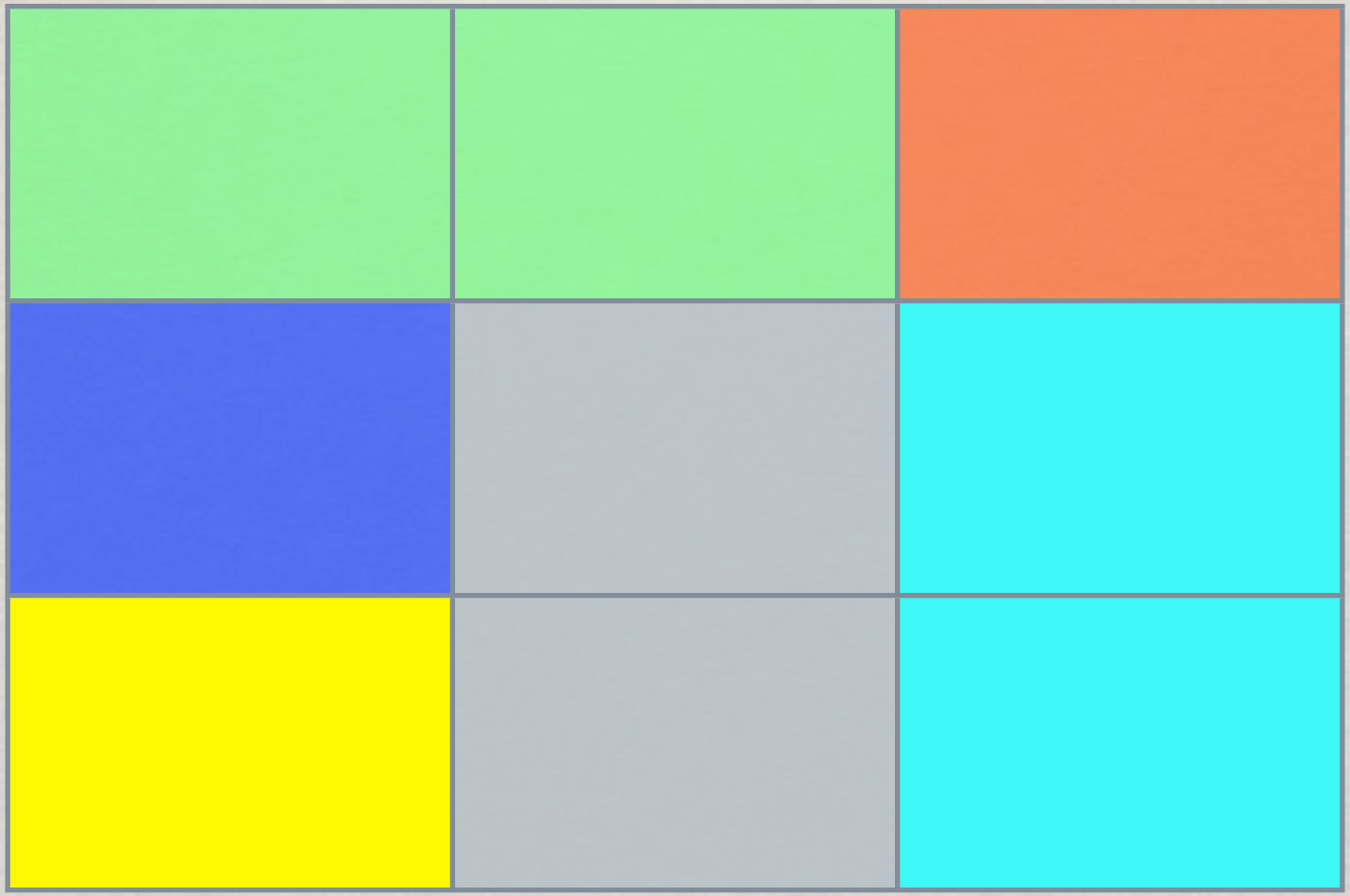
Si on veut que les choses se redimentionnent avec la fenêtre on utilise un Layout

Solutions

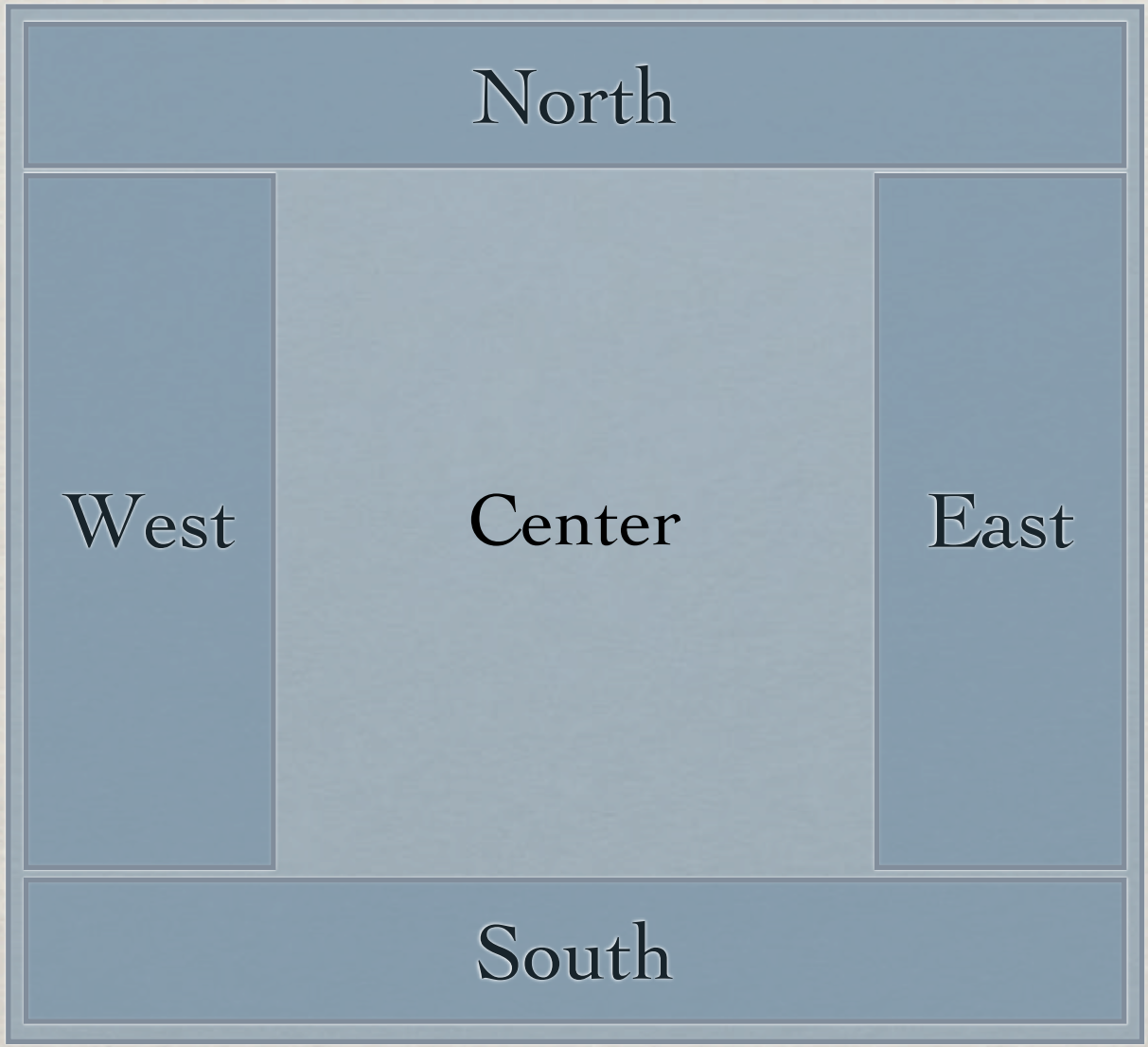
- GridLayout = tous les éléments ont la même taille
- GridBagLayout = tous les éléments ont une taille proportionnelle les uns par rapport aux autres
- BorderLayout = les éléments sont placés par rapport au bord de leur Container

GridLayout

GridBagLayout



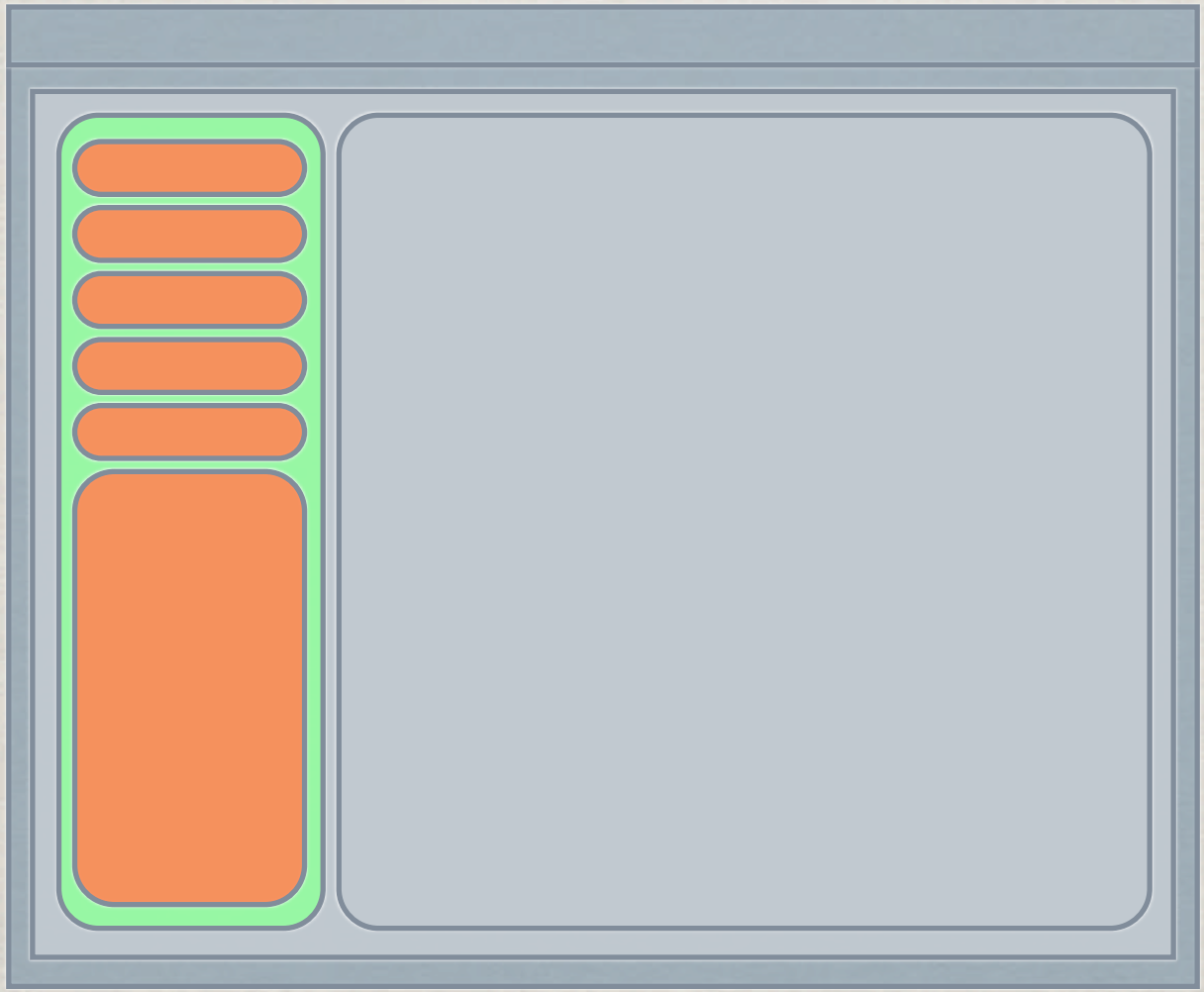
BorderLayout



Et tellement plus encore...

BoxLayout, CardLayout,
FlowLayout,OverlayLayout,
ScrollPaneLayout, SpringLayout,
ViewportLayout

Bonnes pratiques

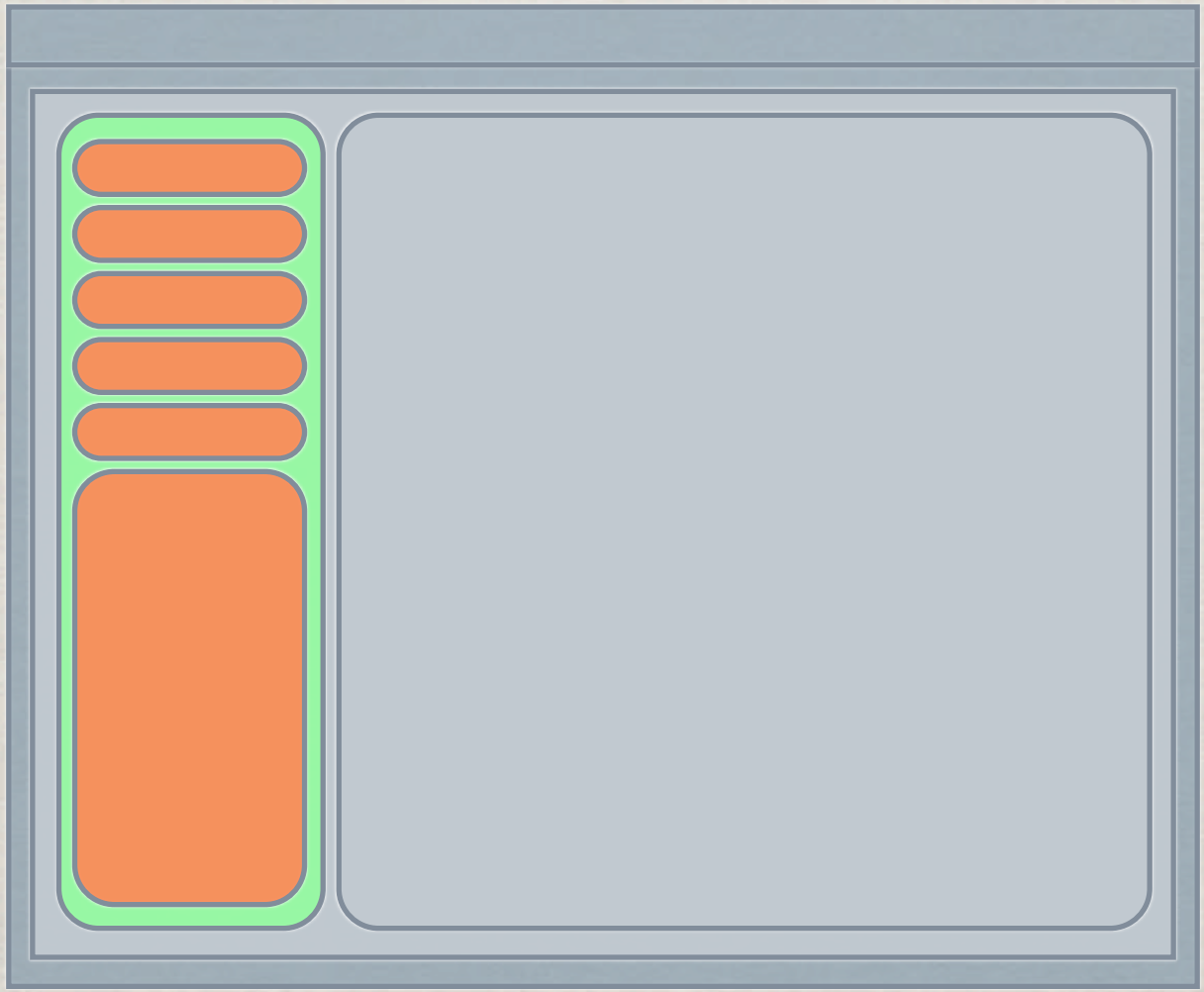


Les choses qui vont ensemble de façon logique vont aussi ensemble dans l'interface graphique

Panels

- Un panel est une sous-zone dans la fenêtre
- Il marche comme une fenêtre dans la fenêtre
- Layouts, etc...

Panels



buttonsPanel
(FlowLayout)

viewPanel
(GridLayout)

mainFrame (BorderLayout)
buttonsPanel en West
viewPanel en Center

Evènements

Rappels : Interfaces

- interface ≠ interface ***graphique***
- implements
- Toutes les méthodes doivent être implémentées

MouseListener

- void mouseClicked(MouseEvent e)
  Press + Release

- void mouseEntered(MouseEvent e)
  On est dessus, sans clic

- void mouseExited(MouseEvent e)
  On n'est plus dessus

- void mousePressed(MouseEvent e)
  Le bouton est enfoncé (mais pas encore relâché)

- void mouseReleased(MouseEvent e)
  Le bouton est relâché

FAQ

- * ?
- -> Regarder dans le MouseEvent passé en paramètre!

MouseEvent

- Object getSource()
  L'objet source sur lequel l'évènement a eu lieu

- int getX() / int getY()
  La position du clic par rapport à l'objet cliqué

- int getModifiers()
  Quel bouton, quelles touches de fonction, etc...

Les autres Listeners

- MouseMotionListener (mouvements de souris)
- KeyListener (clavier)
- WindowListener (évènements liés aux fenêtres)
- TextListener (changements dans les champs texte)
- ActionListener (bazar)

Parenthèse sur ActionListener

- ActionListener est un **fourre-tout**
- Actions non typées, très peu renseignées. “Il s’est passé *quelque chose*”
- Il vaut mieux utiliser les listeners correspondants à ce qu’on veut
- Ne pas faire aveuglément confiance aux tutoriels en ligne!

Démo