Cours de JAVA

Applications Graphiques avec AWT et Swing

Emmanuel ADAM

Institut des Sciences et Techniques de Valenciennes

Université de Valenciennes et du Hainaut-Cambrésis

source principale :
« Thinking in Java (B. Eckel) »

Généralités sur AWT et Swing

- AWT (Abstract Window Toolkit) et SWING comportent :
  - des éléments graphiques
  - des conteneurs
  - Un mécanisme de gestion d'événements, ...

Les objets graphiques de AWT & SWING

- Les éléments graphiques
  - AWT
    - Button → JButton
    - Canvas (zone de dessin)
    - Checkbox → JCheckbox
    - Choice → JComboBox
    - Label → JLabel
    - List → JList
    - MenuBar → JMenuBar
    - MenuItem → JMenuItem
    - TextArea → JTextArea
    - TextField → JTextField
  - SWING
    - JTable, JToolBar, JTree ……

Les conteneurs de AWT & Swing

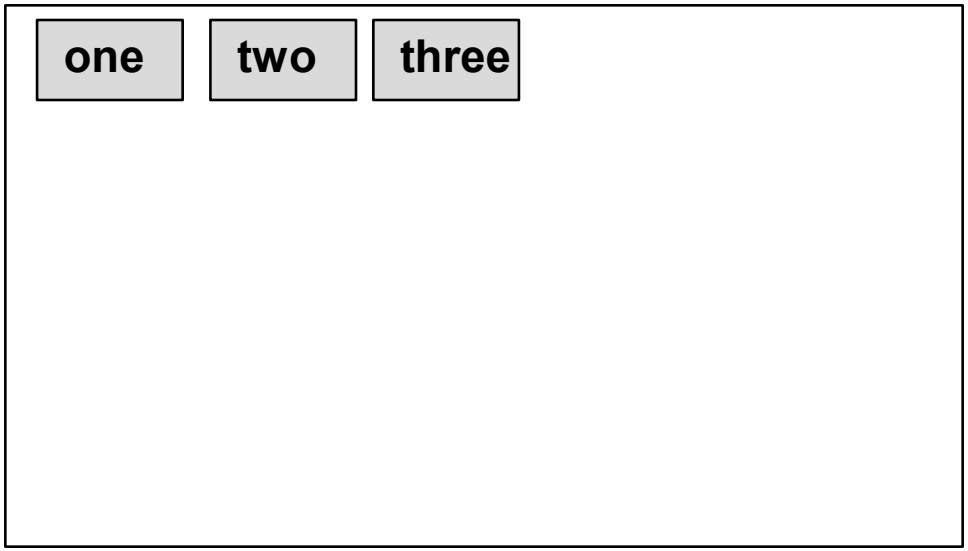
- Les conteneurs sont les couches sur lesquels seront dessinés les objets :
  - Frame // JFrame
  - Dialog // JDialog
  - Window // JWindow
  - Panel // JPanel (conteneur de base)

Les gestionnaires de mise en page

- Permettent de placer automatiquement les objets.
- Possibilité d'utiliser FlowLayout, GridLayout, SpringLayout, BorderLayout (par défaut) …
- Possibilité de créer des gestionnaires de mise en page
- Possibilité de ne pas en utiliser et de placer les éléments par leurs coordonnées
  (demander l'utilisation du gestionnaire null)

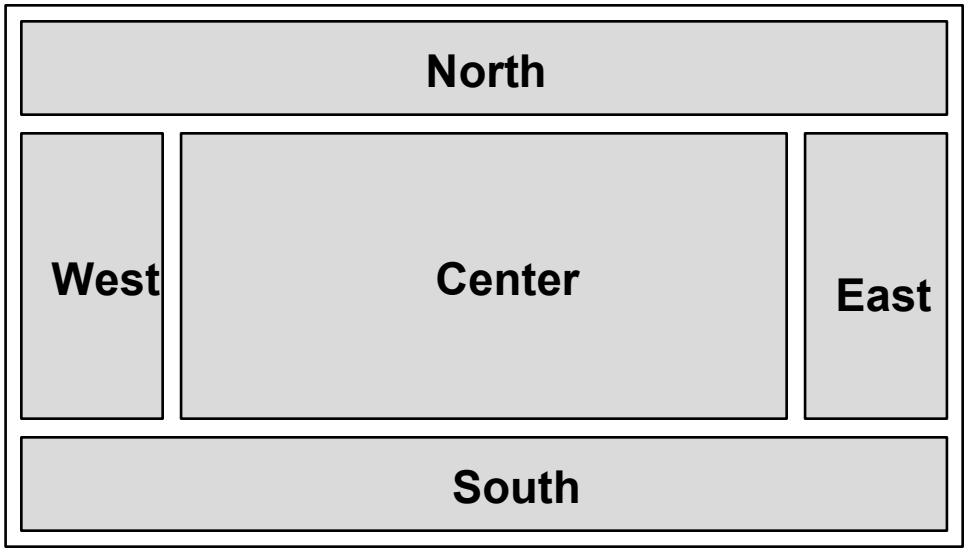
Mise en page : FlowLayout

- java.awt.FlowLayout : de gauche à droite, de haut en bas



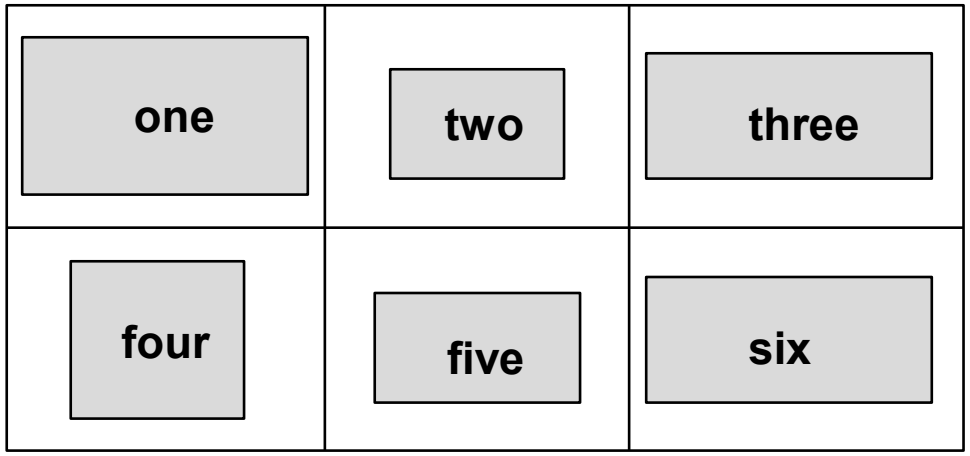
Mise en page : BorderLayout

- java.awt.BorderLayout découpe la zone en Nord, Sud, Est, Ouest et Centre



Mise en page : GridLayout

- java.awt.GridLayout découpe la zone en une grille dont on peut choisir les dimensions



AWT

- Avantages :
  - Permet de créer une interface graphique d'exécutant sous toutes les interfaces (Windows, Xwindows, …)
  - Simplifie la tâche du concepteur de l'IHM (presque tous les objets sont présents)...
- Inconvénients : graphiquement pauvre

Événements

- Chaque objet graphique (ou presque) possède sa propre gestion d'événements,
- Java propose des interfaces d'écoute sur les actions fenêtres, souris, … *(Window, Mouse, ...)Listener*
- Certaines sont implémentées par des classes ‘creuses’ … *(Window, Mouse, ...)Adapter*
  - Avantage, si une classe hérite d'une de ces classes, elle n'a qu'à surcharger la méthode désirée

Swing :
Qu'est ce que c'est ?

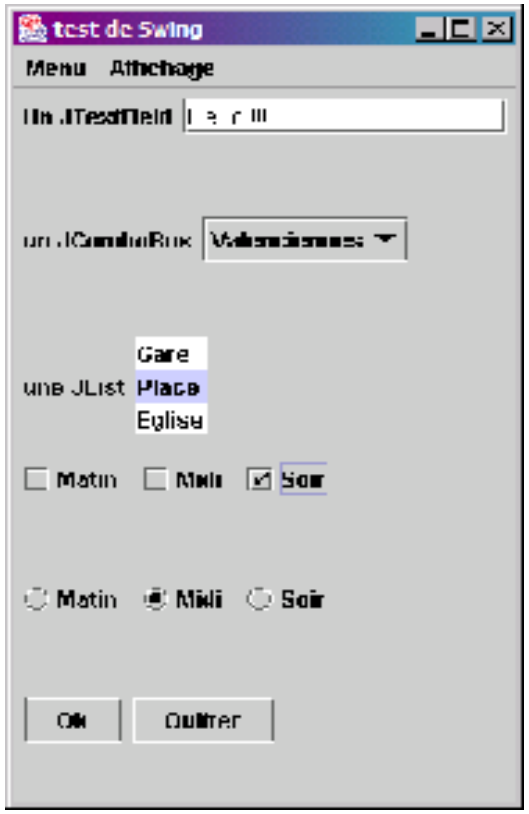
- Swing est un ensemble de librairies graphiques plus évoluées et plus complètes que les librairies AWT.
- Swing est livrée avec le JDK 1.2,
- pour JDK 1.1, il est possible de télécharger les classes swing.

Nouvelles Fonctionnalités

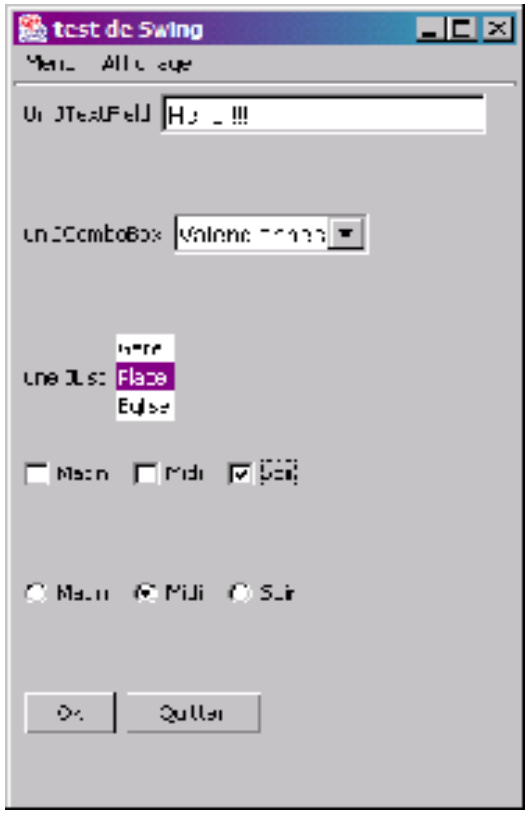
- Swing propose de nouveaux composants tels que
  - Les Tables, les Arbres, la Barre de progression…
  - Les ‘anciens’ objets graphiques AWT ont été surchargés (on trouve maintenant les types JButton, JPanel, ...)
- Avec Swing, il est possible de masquer certains événements
- Il est possible de "zapper" entre les différentes interfaces :
  - l'interface Windows
  - l'interface Motif
  - une interface 3D « métal », …

Exemple d'interfaces SWING (1/3)

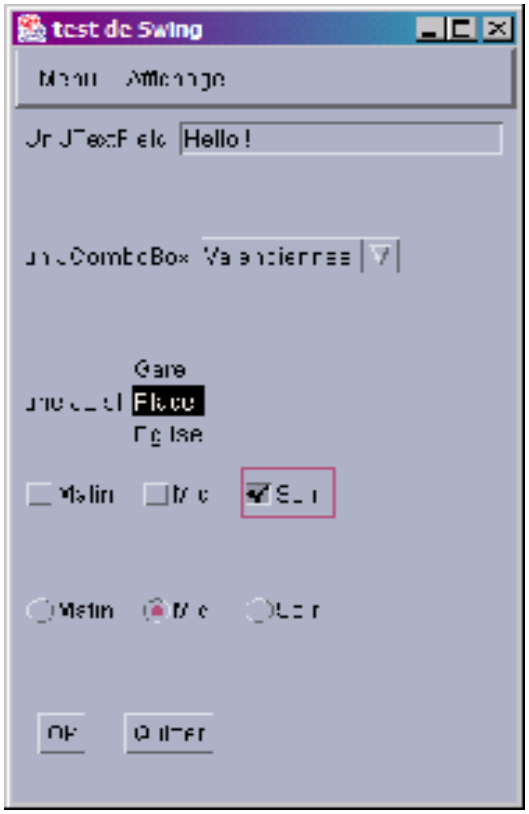
look Motif



look Window



look Metal

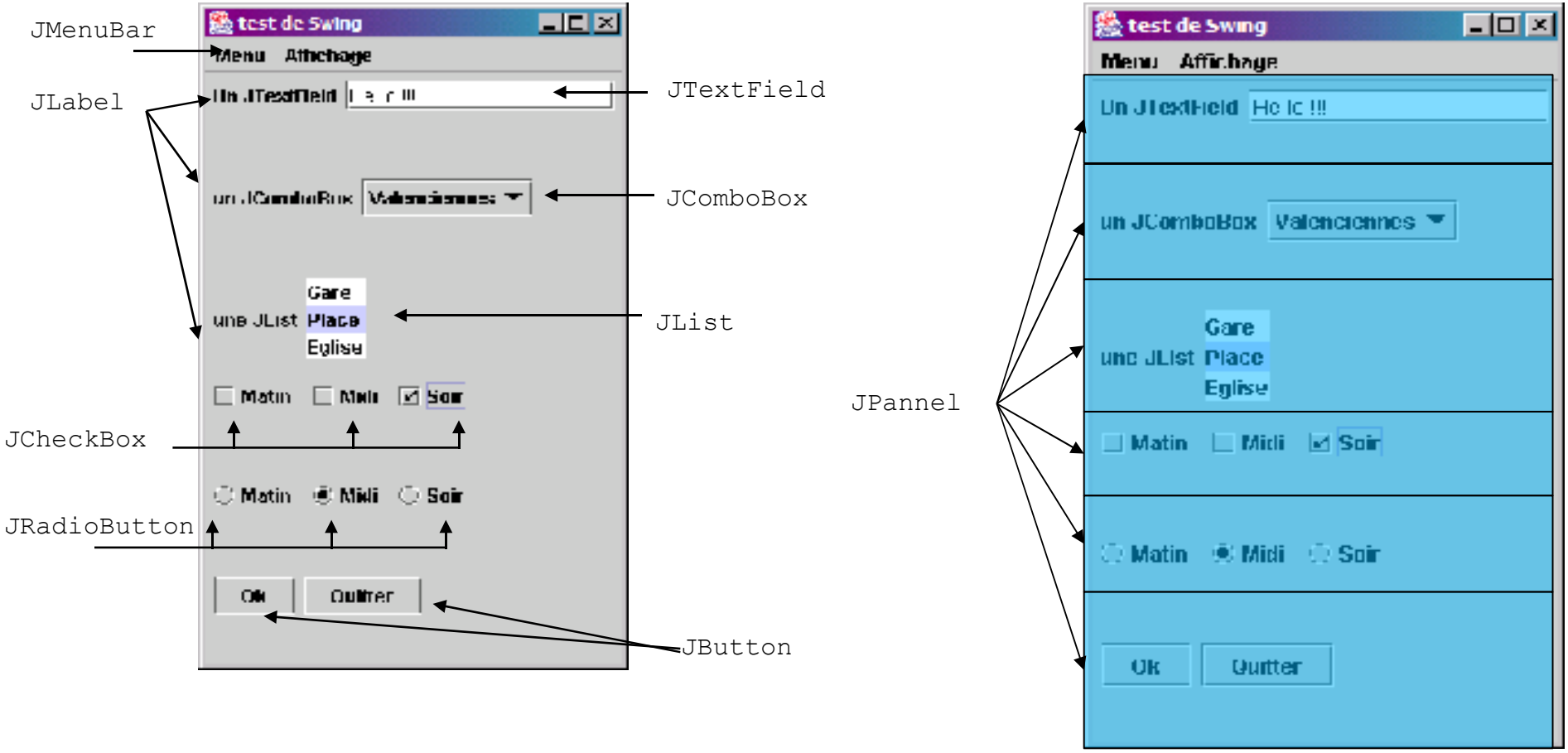


Exemple d'interfaces SWING (2/3)

- Choix des différents modes d’affichage au lancement :
- java TestIHMSwing
- java -Dswing.defaultlaf=com.sun.java.swing.plaf.windows.WindowsLookAndFeel TestIHMSwing
- java -Dswing.defaultlaf=com.sun.java.swing.plaf.motif.MotifLookAndFeel TestIHMSwing

- Possibilité de changer dynamiquement le mode d’affichage

Exemple d'interfaces SWING (3/3)



Exemple de label et de champs texte

```
/** retourne un panneau contenant un label et un champs de texte*/
private JPanel donnerPanneauChampsTexte() {
  JPanel panneau = new JPanel(new FlowLayout(FlowLayout.LEFT));
  JLabel label = new JLabel("Un JTextField");
  panneau.add(label);
  champsTexte = new JTextField(15);
  panneau.add(champsTexte);
  return panneau;
}
```

Exemple de choix

```
/** retourne un panneau contenant un label
    et une liste deroulante */
private JPanel donnerPanneauChoix() {
  JPanel panneau = new JPanel(new FlowLayout(FlowLayout.LEFT));
  JLabel label = new JLabel("un JComboBox");
  panneau.add(label);
  listeDeChoix = new JComboBox();
  listeDeChoix.addItem("Valenciennes");
  listeDeChoix.addItem("Lille");
  listeDeChoix.addItem("Paris");
  panneau.add(listeDeChoix);
  return panneau;
}
```

Exemple de liste

```
/** retourne un panneau contenant un label et une liste */
private JPanel donnerPanneauListe() {
  JPanel panneau = new JPanel(new FlowLayout(FlowLayout.LEFT));
  JLabel label = new JLabel("une JList");
  panneau.add(label);
  liste = new JList(new String[]{"Gare", "Place", "Eglise" });
  panneau.add(liste);
  return panneau;
}
```

Exemple de cases

```
/** retourne un panneau contenant trois cases a cocher */
private JPanel donnerPanneauCases() {
  JPanel panneau = new JPanel(new FlowLayout(FlowLayout.LEFT));
  caseMatin = new JCheckBox("Matin");
  panneau.add(caseMatin);
  caseMidi = new JCheckBox("Midi");
  panneau.add(caseMidi);
  caseSoir = new JCheckBox("Soir");
  panneau.add(caseSoir);
  return panneau;
}
```

Exemple de boutons radio

```
/** retourne un panneau contenant trois boutons radio*/
private JPanel donnerPanneauRadio() {
  JPanel panneau = new JPanel(new FlowLayout(FlowLayout.LEFT));
  //les boutons radio ont un comportement dependant d'un groupe
  ButtonGroup groupeBoutonsRadio = new ButtonGroup();
  radioMatin = new JRadioButton("Matin");
  panneau.add(radioMatin);
  radioMidi = new JRadioButton("Midi");
  panneau.add(radioMidi);
  radioSoir = new JRadioButton("Soir");
  panneau.add(radioSoir );
  groupeBoutonsRadio.add(radioMatin);
  groupeBoutonsRadio.add(radioMidi);
  groupeBoutonsRadio.add(radioSoir);
  // selection du bouton radio
  radioMidi.setSelected(true);
  return panneau;
}
```

Exemple de boutons

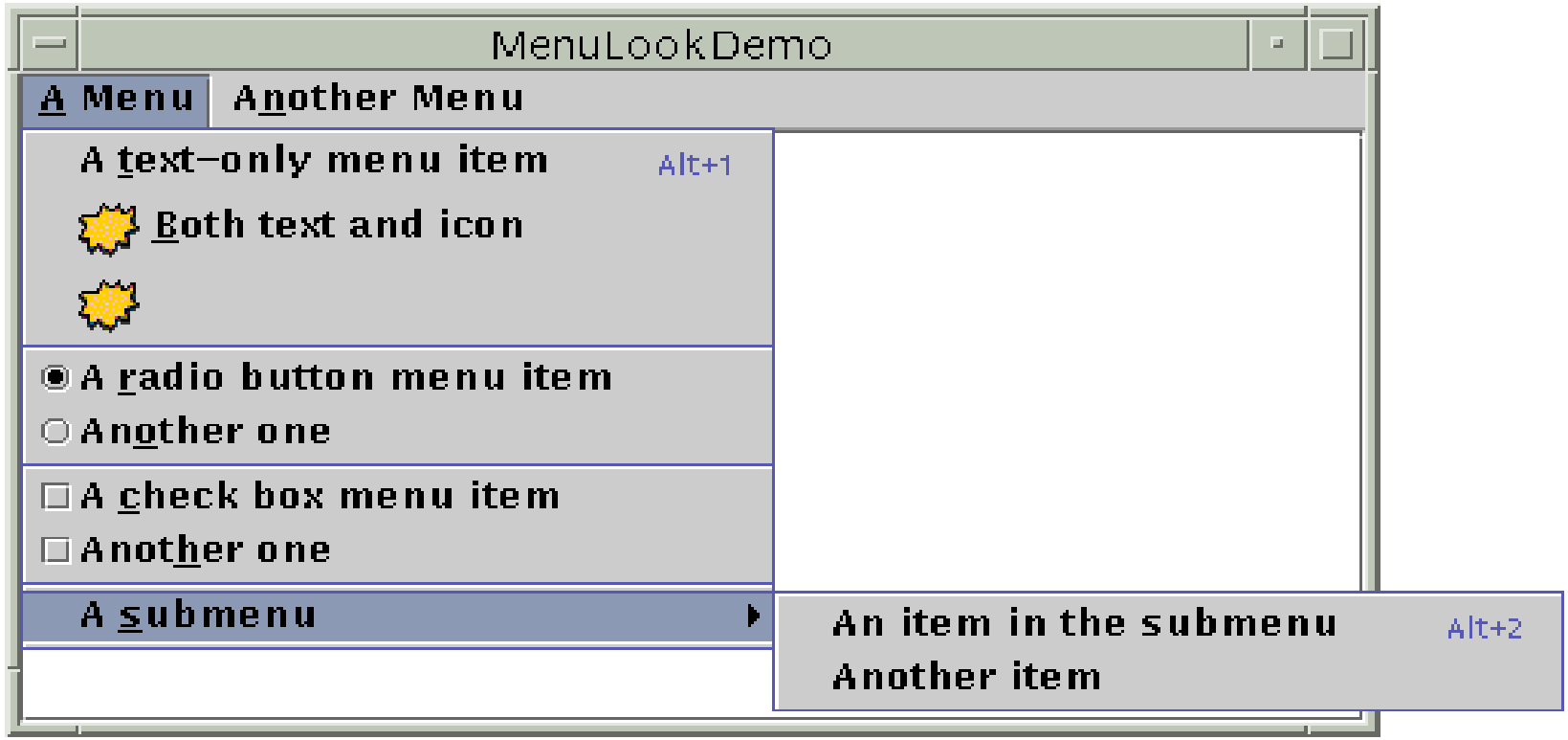
```
/** retourne un panneau contenant deux boutons Ok et Quitter  */
private JPanel donnerPanneauBoutons() {
  // creation d'un objet de gestion d'evenement sur les boutons
  MaGestionButton gestionBouton = new MaGestionButton();
  JPanel panneau = new JPanel(new FlowLayout(FlowLayout.CENTER));
  JButton  boutonOK = new JButton("Ok");
  // ajout de la gestion d'evenement sur le bouton
  boutonOK.addActionListener(gestionBouton);
  panneau.add(boutonOK);
  JButton boutonAnnuler = new JButton("Quitter");
  // ajout de la gestion d'evenement sur le bouton
  boutonAnnuler.addActionListener(gestionBouton);
  panneau.add(boutonAnnuler);
  return panneau;
}
```

Exemple de menu

```
/** retourne la barre de menu composee de deux sous menu : Menu et Affichage*/
private JMenuBar donnerMenu() {
   // creation d'un objet de gestion d'evt sur le menu
  MonMenuListener gestionEvtMenu = new MonMenuListener(this);
  // creation du menu "Menu"
  JMenu menu1 = new JMenu("Menu");
  // ajout de sous menus a "Menu"
  JMenu menu11 = new JMenu("Restaurants");
  JMenu menu12 = new JMenu("Brasseries");
  JMenu menu13 = new JMenu("Restauration Rapide");
  menu1.add(menu11); menu1.add(menu12); menu1.add(menu13);
  // ajout de feuille, interactive, donc associe a la gestion d'evenements
  JMenuItem menu111 = new JMenuItem("Auberge du bon chat");
  menu111.addActionListener(gestionEvtMenu);
  menu11.add(menu111);
  JMenuItem menu121 = new JMenuItem("Romagogo : spécialités créoles");
   menu121.addActionListener(gestionEvtMenu);
  JMenuItem menu122 = new JMenuItem("Steplé : spécialités légioises");
  menu122.addActionListener(gestionEvtMenu);
   menu12.add(menu121);  menu12.add(menu122);
   ...
   // definition d'une barre de menu et ajout des deux menus principaux
  JMenuBar menubar = new JMenuBar();
  menubar.add(menu1); menubar.add(menu2);
   return menubar;}
```

Un autre exemple de menu

inspiré de Sylvain LECOMTE & Vincent POIRRIEZ



Exemple de construction

```
/** constructeur prenant le titre de la fenetre en parametre<br>
 * fait appel aux fonction de création de zones et de menu*/
public TestSwing(String s) {
    super(s);
    // ajout du menu
    setJMenuBar(donnerMenu());
    // recuperation du conteneur
    Container content = getContentPane();
    // mise en forme : 1 colone, nb de lignes indefini espacement entre colones et lignes de 2
    content.setLayout(new GridLayout(0, 1, 2, 2));
    // ajout de la premiere ligne
    content.add(donnerPanneauChampsTexte());
    // ajout de la 2eme  ligne
    content.add(donnerPanneauChoix());
    // ajout de la 3eme ligne
    content.add(donnerPanneauListe());
    // ajout de la 4eme ligne
    content.add(donnerPanneauCases());
    // ajout de la 5eme ligne
    content.add(donnerPanneauRadio());
    // ajout de la 6eme ligne
    content.add(donnerPanneauBoutons());
    // par defaut, le clic sur la croix quitte l'application
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    // optimiser l'espace dans la fenetre
    pack();
}
```

Gestion d'événements AWT (1/2)

- AWT permet la gestion d'évenements simples:
  - sur les composants des interfaces
  - provenant du clavier, de la souris (clic, déplacement, roulette), d'autres objets
- voir le package java.awt.event

Exemple de gestion d'événement, sur boutons (1/2)

- Gestion des événements : clic sur le bouton Ok

```
public class MaGestionButton implements ActionListener {
    /** fonction declenchee automatiquement par une action sur les objets lies a la
    gestion d'evenement; recuperation du label de l'objet et declenchement des actions
    correspondantes*/
     public void actionPerformed(ActionEvent actionevent) {
        String nomBouton = actionevent.getActionCommand();
        if (nomBouton.equals("Ok")) actionBoutonOK();
        else if (nomBouton.equals("Quitter")) System.exit(0);
        else System.out.println("evenement non traite : " + nomBouton);
     }
...
```

Exemple de gestion d'événement, sur boutons (2/2)

```
/** fonction lancée en cas de clic sur ok, affiche les valeurs des composants swing<br>
     possible car cette classe est interne donc a accès aux éléments de la classe TestSwing*/
    private void actionBoutonOK() {
        System.out.println("champsTexte = " + champsTexte.getText());
        System.out.println("listeDeChoix = " + listeDeChoix.getSelectedItem());
        System.out.println("liste = " + liste.getSelectedValue());
        System.out.println("caseMatin=" + (caseMatin.isSelected() ? "Active" : "Non active"));
        System.out.println("caseMidi=" + (caseMidi.isSelected() ? "Active" : "Non active"));
        System.out.println("caseSoir=" + (caseSoir.isSelected() ? "Active" : "Non active"));
        System.out.println("radioMatin=" + (radioMatin.isSelected() ? "Active" : "Non active"));
        System.out.println("radioMidi=" + (radioMidi.isSelected() ? "Active" : "Non active"));
        System.out.println("radioSoir=" + (radioSoir.isSelected() ? "Active" : "Non active"));
    }
}
```

Exemple de gestion d'événement, sur menu (1/2)

```
/** classe de gestion d'événements associés aux menus d'une fenêtre de type
      TestSwing*/
class MonMenuListener implements ActionListener {
   /** fenetre de type TestSwing a laquelle est liee la gestion d'evenement*/
   TestSwing fenetre;

   /** constructeur prenant une fenetre en parametre afin de pouvoir y acceder */
   public MonMenuListener(TestSwing _fenetre) {
      fenetre = _fenetre;
   }
   /** fonction declenchee automatiquement par une action sur les objets lies a la gestion
     d'evenement, ici, sur les feuilles des menu de la fenetre.
    * recuperation du label de l'objet et declenchement des actions correspondantes*/
   public void actionPerformed(ActionEvent actionevent) {
      String nom = actionevent.getActionCommand();
      System.out.println("choix du menu: " + nom );
      String look = fenetre.lookMotif;
```

Exemple de gestion d'événement, sur menu (2/2)

```
if (nom.equalsIgnoreCase("windows")) {
        System.out.println("...windows...");
        look = fenetre.lookWindows;
    }
    else
        if (nom.equalsIgnoreCase("motif")) {
            System.out.println("...motif...");
            look = fenetre.lookMotif;
        }
        else
            if (nom.equalsIgnoreCase("metal")) {
                System.out.println("...metal...");
                look = fenetre.lookMetal;
            }
    try {
        UIManager.setLookAndFeel(look);
        SwingUtilities.updateComponentTreeUI(fenetre);
    }
    catch(Exception exception) { exception.printStackTrace(); }
  }
}
```

changement du look de la fenêtre
(voir ses attributs pour les noms de look)

Gestion d'événements Swing

- Swing permet
  - la gestion d'événements sur les composants évolué (Arbre, Table, …)
  - propose de nouvelles écoutes d'événements
    - Exemple: déplacement dans un menu:

```
 class GestionMenu implements MenuListener {
public void menuCanceled(MenuEvent menuEvent) {  }
public void menuDeselected(MenuEvent menuEvent) { }
public void menuSelected(MenuEvent menuEvent) {
    JMenu menu = (JMenu)menuEvent.getSource();
    System.out.println("menu sélectionné: " + menu.getText());
} }
```

- Voir le package javax.swing.event

Swing dans une applet

Création d'applets utilisant des composants swing.

Pb : Il faut que les navigateurs puissent interpréter le swing.

```
import javax.swing.*;  import java.awt.*;
public class HelloSwingApplet extends JApplet
{
  public void init()
  {
    JLabel label = new JLabel(" Ceci est une applet Swing !");
    label.setHorizontalAlignment(JLabel.CENTER);
      //Ajout d'un cadre
      label.setBorder(BorderFactory.createMatteBorder(1,1,2,2,Color.black));
      getContentPane().add(label, BorderLayout.CENTER);
  }
}
```

Internationalisation (1/4)

- Modifier la langue des interactions sans modifier le code
- Définition de fichiers de traduction contenant « le paquet des messages »
- Utilisation de
  - java.util.Locale;
  - java.util.ResourceBundle;

Internationalisation : Exemple simple (2/4)

- Ex. Affichage en anglais et français de bonjour, …
- définition des dictionnaires :

le fichier [MesMessages_fr_FR.properties] contient

```
debutCommunication = Bonjour...
demande = Comment allez vous ?
finCommunication = Au revoir
```

le fichier [MesMessages _en_EN.properties] contient

```
debutCommunication = Hello...
demande = How are you ?
finCommunication = GoodBye…
```

+ définition du dictionnaire par défaut
[MesMessages.properties]

Internationalisation : Exemple simple (3/4)

- Deux indicateurs : pour la langue, pour la région
  - exemple : fr pour français, FR pour France, CA pour Canada, …

```
public static void main(String args[])
{
    String langage = new String(args[0]); // premier parametre
    String region = new String(args[1]); // second parametre
    Locale langueRegionale = new Locale(langage, region);
    ResourceBundle rb = ResourceBundle.getBundle("MesMessages", langueRegionale);
    System.out.println(rb.getString("debutCommunication"));
    System.out.println(rb.getString("demande"));
    System.out.println(rb.getString("finCommunication"));
}
```

Internationalisation : Exemple simple (4/4)

- Exécution :

```
/internationalisation>java BonjourHello en US
```

donne

```
Hello
How are you?
Goodbye

/internationalisation>java BonjourHello fr FR
```

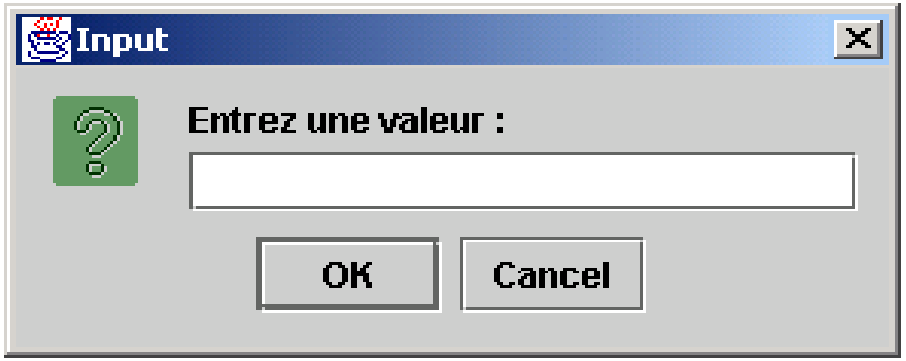
donne

```
Bonjour...
Comment allez vous ?
Au revoir...
```

Classes utiles : JOptionPane

- La classe JOptionPane permet d'afficher une fenêtre de dialogue *"rapidement"*
- *Exemple : demande d'une chaîne*

```
String valeur = JOptionPane.showInputDialog("Entrez une valeur : ");
```



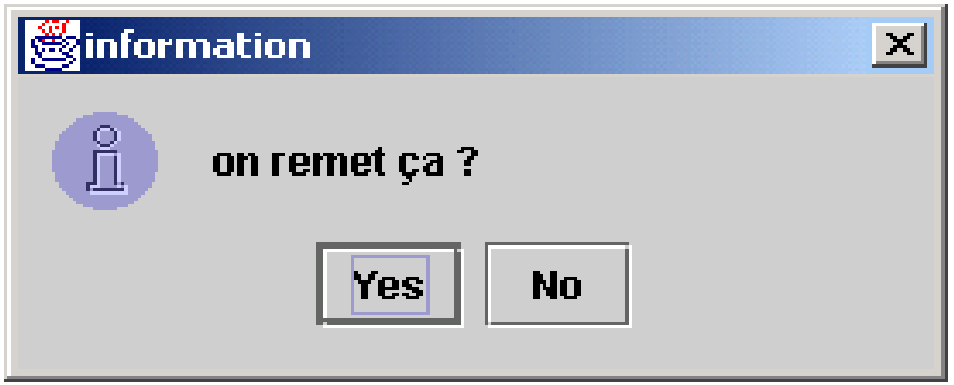
Classes utiles : JOptionPane

- **JOptionPane** = icône + message + valeur demandée + boutons
- ***Type de messages*** : ERROR_MESSAGE, PLAIN_MESSAGE, QUESTION_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE
- ***Type de boutons*** : DEFAULT_OPTION, YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION
- Exemple de message d'alerte :
  - JOptionPane.showMessageDialog (**null**, "attention", "probleme", JOptionPane.ERROR_MESSAGE);

Classes utiles : JOptionPane

- Exemple de message à choix

```
JOptionPane.showConfirmDialog(null,
    "on remet ça ?",
    "information",
    JOptionPane.YES_NO_OPTION,
    JOptionPane.INFORMATION_MESSAGE);
```

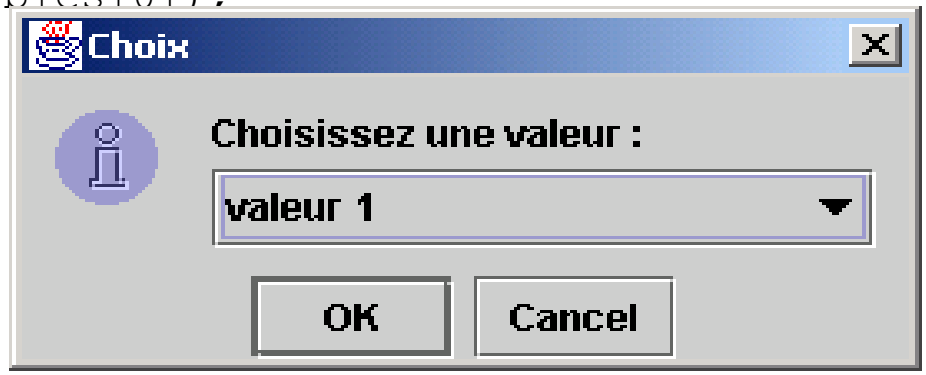


Classes utiles : JOptionPane

- Exemple de message à choix

```
Object[] valeursPossibles = { "valeur 1", "valeur 2", "valeur 3" };

Object valeurSelectionnee =
    JOptionPane.showInputDialog(null, "Choisissez une valeur : ",
    "Choix", JOptionPane.INFORMATION_MESSAGE, null,   valeursPossibles,
    valeursPossibles[0]);
```



Conclusion sur Swing

- Swing offre surtout beaucoup plus de composants que AWT (trop ?) :
  - Le rendu est meilleur
  - Les options nombreuses
- Mais l'API est encore difficile à connaître
- ne pas hésiter à utiliser le Tutorial...