

## Chapitre 6

### Interfaces graphiques

- Les Applets
- Les composants graphiques
  - conteneurs
  - widgets
- Les gestionnaires de placement (Layout)
- Les évènements
- Les écouteurs d'évènements

Programmation Java 2003/2004 - D.Pastre

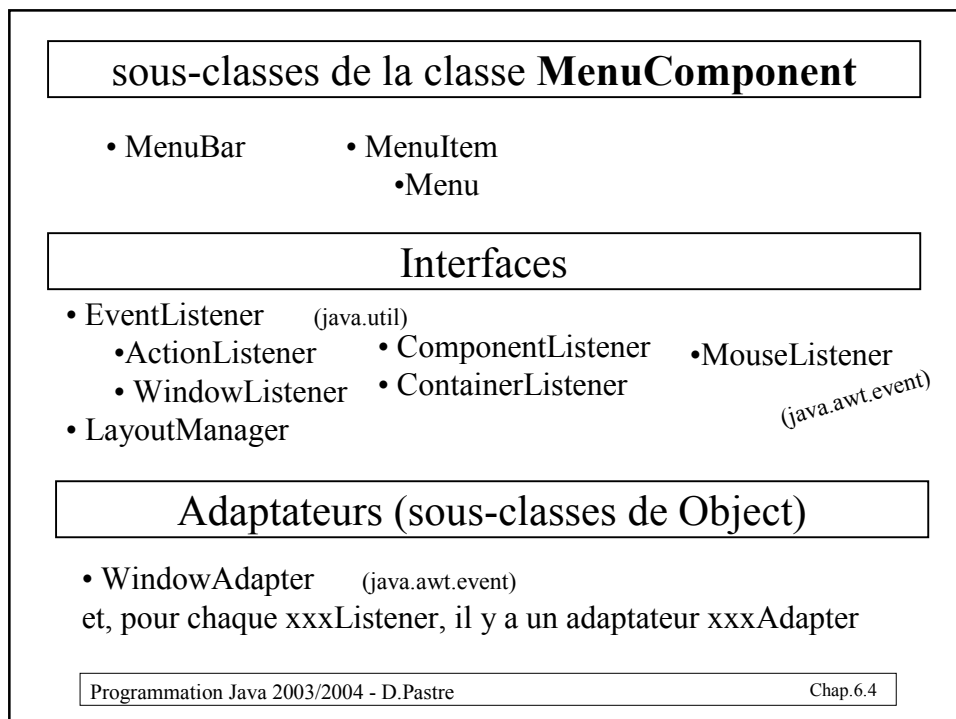
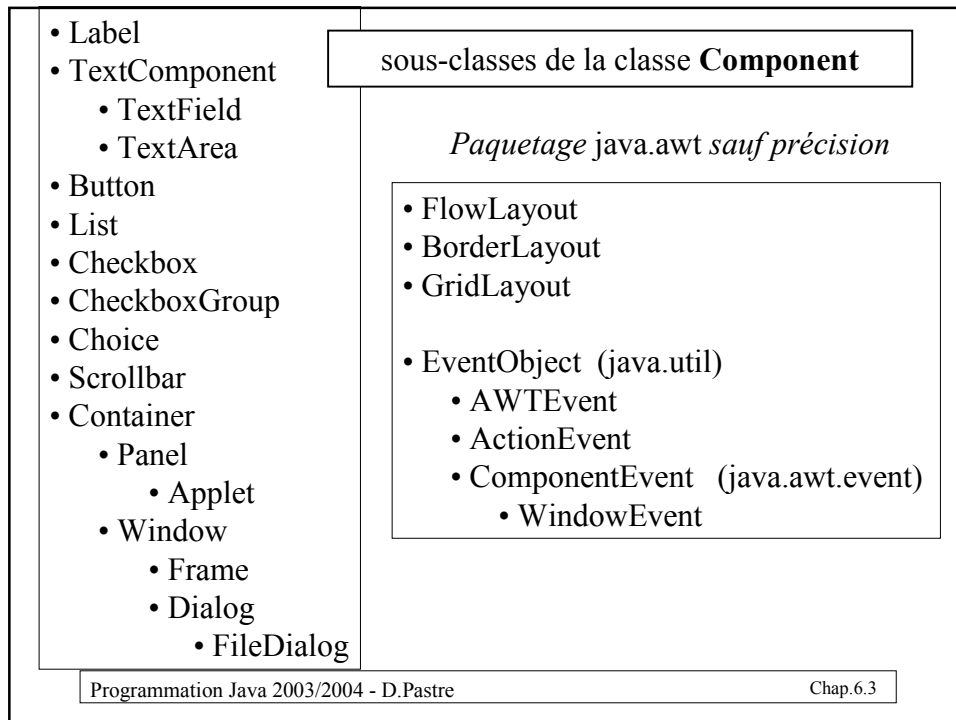
Chap.6.1

### Quelques classes et leurs méthodes

+java.awt.Component : setVisible  
*(conteneurs) composants qui peuvent contenir d'autres composants*  
+java.awt.Container : add, setLayout  
+java.awt.Panel  
+java.applet.Applet : init, start, paint, stop, destroy  
+java.awt.Window : pack, show  
+java.awt.Frame :  
*(widgets) composants qui ne sont pas des conteneurs*  
+java.awt.Button :  
+java.awt.Label : set/getText  
+java.awt.TextComponent : getText, setForeground, set/getSize,  
+java.awt.TextField : setText  
+java.awt.MenuComponent

Programmation Java 2003/2004 - D.Pastre

Chap.6.2



## Les Applets

Programmation Java 2003/2004 - D.Pastre

Chap.6.5

Les **applets** sont des application Java pouvant s'exécuter via un navigateur, par l'interprétation du pseudo-code inclus dans une page *html*.

On exécute alors la page html

- par la commande Unix `appletviewer`
- à partir d'un gestionnaire de fichiers (click)
- à partir d'un navigateur

On peut aussi exécuter l'applet directement sous JBuilder.

On peut aussi créer une applet à partir d'une autre application java.

Les applets sont des extensions de la classe *Applet*, dans lesquelles des méthodes doivent être définies.

Programmation Java 2003/2004 - D.Pastre

Chap.6.6

*Exemple : fichier **AppletBonjour.java***

```
package bonjour;
import java.applet.Applet;
import java.awt.Graphics;
public class AppletBonjour extends Applet{
    public void paint(Graphics g) {
        g.drawString("bonjour tout le monde", 10,20);
        g.drawString("bonjour", 10,40);
        g.drawString(" tout", 15,50);
        g.drawString(" le", 15,60);
        g.drawString(" monde", 15,70);
        g.drawOval(20,20,200,100);
        g.drawRect(20,140,200,100);
    }
}
```

Et le fichier **AppletBonjour.html** associé

```
<HTML>
<HEAD>
<TITLE> fichier html appelant l'applet AppletBonjour </TITLE>
</HEAD>

<BODY>Texte avant d'appeler AppletBonjour </H1>
<APPLET
    code="bonjour.AppletBonjour.class"
    codebase = "../classes"
    width = "300"
    height = "300" >
</APPLET>
</BODY>
</HTML>
```

## Quelques méthodes

- De la classe **Graphics**

drawString  
drawLine  
drawOval  
drawRect  
drawArc  
getColor  
setColor

- De la classe **Applet**

paint  
init  
start  
stop  
destroy

## Les "Frame" et les événements

- premières fenêtres
- les fenêtres de lecture de notes
- les calculettes

## Une fenêtre minimale

```
package fenetres;
import java.awt.Frame; import java.awt.Label;
import java.awt.TextField; import java.awt.FlowLayout;
/** Une fenêtre minimale : composants et affichage */
public class Fenetre0 extends Frame {
    public Fenetre0() {
        Label etiq = new Label("message de bienvenue"); // étiquette
        TextField tf = new TextField(15); // champ
        this.setLayout(new FlowLayout()); // agencement
        this.add(etiq); this.add(tf);
        tf.setText("Bonjour tout le monde"); // affichage
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.6.11

## Le programme principal et le résultat

```
package fenetres;
public class TestFen0 {
    public static void main(String[] args) {
        Fenetre0 f = new Fenetre0();
        f.pack(); // pour assembler
        f.show(); // pour afficher
    }
}
```



Programmation Java 2003/2004 - D.Pastre

Chap.6.12

### Gestion d'un événement : lecture

```
...
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/** gestion d'un évènement : lecture */
public class Fenetre0 extends Frame implements ActionListener {
    private TextField tfNom = new TextField(15);
    private TextField tfMessage = new TextField(20);
    public Fenetre0(String titre) {
        super(titre); // constructeur de la classe mère
        Label etiq = new Label("nom");
        this.setLayout(new FlowLayout());
        this.add(etiq); this.add(tfNom); this.add(tfMessage);
        this.tfNom.addActionListener(this); // pour écouter
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.6.13

```
// définition de la méthode abstraite de ActionListener
public void actionPerformed(ActionEvent e) {
    String nom = this.tfNom.getText();
    String message = "Bonjour "+nom;
    this.tfMessage.setText(message);
}
}
```

```
// dans le programme principal : un titre
Fenetre0 f = new Fenetre0("Fenetre0");
```



Programmation Java 2003/2004 - D.Pastre

Chap.6.14

### Arrêt du programme

```
// arrêt du programme au click sur la case de fermeture */
...
public Fenetre0(String titre) {
    ...
    this.addWindowListener(new GestionnaireExit());
}
```

On doit définir la classe **GestionnaireExit** qui *implémente* l'interface **WindowListener**.

On pourra aussi définir plus simplement **GestionnaireExit** comme sous-classes de l'*adaptateur* **WindowAdapter**

```
package fenetres;
import java.awt.event.WindowListener;
import java.awt.event.WindowEvent;

public class GestionnaireExit implements WindowListener {
    public void windowClosing(WindowEvent e) {System.exit(0);}
    // les méthodes suivantes ne font rien mais il faut les définir
    // l'adaptateur WindowAdapteur permettra de l'éviter
    public void windowActivated(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowOpened(WindowEvent e) {}
}
```



## Fenêtres de lectures de notes dans une interface graphique

- boutons
- gestion de plusieurs évènements possibles
- diverses possibilités d'agencement
- gestion des exceptions

Programmation Java 2003/2004 - D.Pastre

Chap.6.17

```
...
import java.awt.FlowLayout; // composants en ligne
import java.awt.GridLayout; // composants en colonne
import java.text.DecimalFormat;
/** Le programme Bonjour avec une interface graphique simple.
 * Nom, prénom, age et deux notes. */
public class Fenetre1 extends Frame implements ActionListener {
    private TextField tfNom =new TextField(10); // identite
    private TextField tfPrenom =new TextField(10);
    private TextField tfAnnee =new TextField(10); // naissance
    private double note1, note2; // notes
    private TextField tfNote1, tfNote2;
    private Button bValider = new Button("valider"); // boutons
    private Button bQuitter = new Button("quitter");
    private TextField tfAge = new TextField(10); // réponses
    private TextField tfMoyenne = new TextField(10);
```

Fenetre1

Programmation Java 2003/2004 - D.Pastre

Chap.6.18

```

public Fenetre1(String titre) {
    super(titre);
    // this.setLayout(new FlowLayout()); // composants en ligne
    this.setLayout(new GridLayout(15,1)); // composants en colonne
    this.add(new Label("nom"));this.add(tfNom);
    this.add(new Label("prenom"));this.add(tfPrenom);
    this.add(new Label("année de naissance"));this.add(tfAnnee);
    this.add(new Label("notes"));
    tfNote1=new TextField(5); tfNote2=new TextField(5);
    this.add(tfNote1);this.add(tfNote2);
    this.add(bValider); // bouton pour enregistrer les données
    this.add(new Label("age"));this.add(tfAge);
    this.add(new Label("moyenne"));this.add(tfMoyenne);
    this.add(bQuitter); // bouton pour arrêter le programme
    bValider.addActionListener(this); // cette fenêtre doit écouter
    bQuitter.addActionListener(this); // les deux boutons
    this.addWindowListener(new GestionnaireExit());}

```

Programmation Java 2003/2004 - D.Pastre

Chap.6.19

```

public void actionPerformed(ActionEvent e) {
    if (e.getSource()== bQuitter) System.exit(0) ;
    DecimalFormat df = new DecimalFormat("0.###");
    String nom = tfNom.getText(); // identite
    String prenom = tfPrenom.getText();
    String annee = tfAnnee.getText(); // naissance
    int age = 2004 - Integer.parseInt(annee);
    tfAge.setText(""+age);
    // notes
    note1=Double.parseDouble(tfNote1.getText());
    note2=Double.parseDouble(tfNote2.getText());
    this.add(new Label("moyenne"));
    tfMoyenne.setText(""+(note1+note2)/2); }
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.6.20

Fenetre2

```

...
import java.awt.Panel;
/** 2ème version du programme Bonjour.
 * Agencement par panneaux. Exception. */
public class Fenetre2 extends Frame implements ActionListener {
    ...
    public Fenetre2(String titre) { super(titre);
        this.setLayout(new GridLayout(6,1)); // 6 composants en colonne

        // identite (1er Panel)
        Panel identite = new Panel(new FlowLayout()); this.add(identite);
        identite.add(new Label("nom")); identite.add(tfNom);
        identite.add(new Label("prenom")); identite.add(tfPrenom);

        // naissance (2ème Panel)
        Panel naissance = new Panel(new FlowLayout()); this.add(naissance);
        naissance.add(new Label("année de naissance"));
        naissance.add(tfAnnee);

```

Programmation Java 2003/2004 - D.Pastre
Chap.6.21

```

// notes (3ème Panel)
Panel pNotes = new Panel(new FlowLayout()); this.add(pNotes);
pNotes.add(new Label("notes"));
tfNote1=new TextField(5); tfNote2=new TextField(5);
pNotes.add(tfNote1); pNotes.add(tfNote2);

// boutons (4ème Panel)
Panel boutons = new Panel(new FlowLayout()); this.add(boutons);
boutons.add(bValider); boutons.add(bQuitter);

// age et moyenne (5ème et 6ème Panel)
Panel pAge = new Panel();this.add(pAge);
pAge.add(new Label("age"));pAge.add(tfAge);
Panel pMoyenne = new Panel();this.add(pMoyenne);
pMoyenne.add(new Label("moyenne"));
pMoyenne.add(tfMoyenne);

...

```

Programmation Java 2003/2004 - D.Pastre
Chap.6.22

```

public void actionPerformed(ActionEvent e) {
    ... // identite
    if (nom.equals("")) System.out.println("nom absent");
    try { // naissance
        String annee = tfAnnee.getText();
        int age = 2004 - Integer.parseInt(annee);
        tfAge.setText(""+age);
    } catch(NumberFormatException ex) {
        System.out.println("année de naissance absente ou incorrecte");}
    try { // notes
        note1=Double.parseDouble(tfNote1.getText());
        note2=Double.parseDouble(tfNote2.getText());
        tfMoyenne.setText(""+df.format((note1+note2)/2));
    } catch(NumberFormatException ex) {
        System.out.println("note absente ou incorrecte");}
    }
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.6.23

The screenshot shows a Java Swing window with the title bar '2'. Inside the window, there is a form with the following elements:

- Two text input fields at the top: 'nom' containing 'Dupond' and 'prenom' containing 'Jule'.
- A text input field labeled 'année de naissance' containing '1981'.
- Two text input fields labeled 'notes' containing '12' and '16'.
- Two buttons labeled 'valider' and 'quitter' below the notes fields.
- A text input field labeled 'age' containing '23'.
- A text input field labeled 'moyenne' containing '14'.

Programmation Java 2003/2004 - D.Pastre

Chap.6.24

### Fenetre3

```
...
import java.awt.TextArea;
import java.awt.BorderLayout;

/** 3ème version du programme Bonjour.
 *   Meilleur agencement et zone de texte.
 *   Meilleure gestion des exceptions
 *   et affichage des messages d'erreur. */

public class Fenetre3 extends Frame implements ActionListener {
    ...
    private TextArea ta = new TextArea(6,10); // zone de texte
```

this

questions

identite    nom     prénom

naissance    année

pNotes    notes

boutons

valider

quitter

ta

```

public Fenetre3(String titre) { super(titre);
    this.setLayout(new BorderLayout()); // N S Centre O E
    // questions (1er Panel au nord)
    Panel questions= new Panel(new GridLayout(3,1));
                                // 3 composants en colonne
    this.add("North", questions);
        Panel identite = ... ; questions.add(identite); ... // identite
        Panel naissance= ... ; questions.add(naissance); ... // naissance
        Panel pNotes = ... ; questions.add(pNotes); ... // notes
    // boutons (2ème Panel au centre)
        Panel boutons = ... ;
    this.add("Center",boutons);
    ...
    // zone de texte (3ème composant au sud);
    this.add("South",ta);
    ... }

```

Programmation Java 2003/2004 - D.Pastre

Chap.6.27

```

public void actionPerformed(ActionEvent e) {
    ...
    message += "bonjour "+prenom+" "+nom;
    if (nom.equals("")) message += "\n(nom absent)";
    try { // naissance
        String annee = tfAnnee.getText();
        if (annee.equals(""))
            throw new Exception("l'année de naissance absente");
        int age = 2004 - Integer.parseInt(annee);
        message += "\nvous avez "+age+" ans";
    } catch(NumberFormatException ex) {
        message += "l'année de naissance incorrecte";
    }
    catch(Exception ex) {message += ex.getMessage();}
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.6.28

```

try { // notes
    String s = tfNote1.getText();
    if (s.equals("")) throw new Exception("\nnote absente");
    note1=Double.parseDouble(s);
    s = tfNote2.getText();
    if (s.equals("")) throw new Exception("\nnote absente");
    note2=Double.parseDouble(s);
    message += "\n\nvos notes sont "+df.format(note1)
                +" et "+df.format(note2);
    message += "\nmoyenne "+df.format((note1+note2)/2);
} catch(NumberFormatException ex) {
    message += "\n\nnote incorrecte";}

    catch(Exception ex) {message += ex.getMessage();}

// affichage du message
ta.setText(message);    }

```

Programmation Java 2003/2004 - D.Pastre

Chap.6.29



Programmation Java 2003/2004 - D.Pastre

Chap.6.30

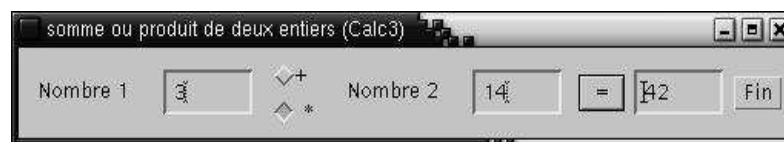
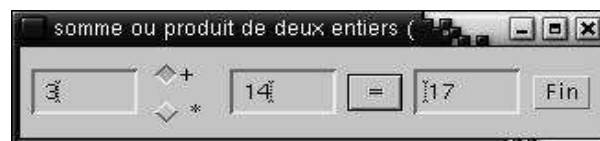
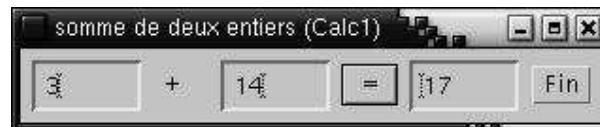
## Les calculettes

- cases à cocher
- gestionnaire de fermeture
- menus

Programmation Java 2003/2004 - D.Pastre

Chap.6.31

### Les 4 calculettes



Programmation Java 2003/2004 - D.Pastre

Chap.6.32



### Une classe principale ouvrant les 4 calculettes

```
public class TestCalc {  
    public static void main(String[] args) { // plusieurs calculettes  
        Calc1 c1= new Calc1();  
        c1.pack(); // Calc1 : somme de deux entiers  
        c1.show();  
        Calc2 c2= new Calc2();  
        c2.pack(); // Calc2 : somme ou produit de deux entiers  
        c2.show();  
        Calc3 c3= new Calc3();  
        c3.pack(); // Calc3 : idem Calc2 avec étiquettes  
        c3.show();  
        Calc4 c4 = new Calc4();  
        c4.pack(); // Calc : somme ou produit de deux rationnels  
        c4.show();  
    }
```

Programmation Java 2003/2004 - D.Pastre

Chap.6.33

### Calc1

```
public class Calc1 extends Frame implements ActionListener {  
    /** crée une fenêtre pour dessiner une 1ère version de calculette  
        effectuant des sommes d'entiers, avec  
        deux champs pour recevoir des entiers  
        un bouton = pour demander le calcul  
        un champ pour donner le résultat  
        un bouton Fin pour quitter  
        un champ pour les messages d'erreur */  
    public Calc1(String nom) {...}  
  
    /** effectue le calcul si possible après un click sur le bouton =  
        sinon affiche un message d'erreur  
        un click sur le bouton Fin arrête le programme  
        un click sur la case de fermeture ferme le programme */  
    public void actionPerformed(ActionEvent e) {...}  
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.6.34

import ...

Calc1 (suite)

```
public class Calc1 extends Frame implements ActionListener {  
    private Button bFin; // pour sortir  
    private Button bCalcul; // pour demander le calcul  
    private TextField top1; // pour rentrer le premier nombre entier  
    private Label lab ; // juste pour le decor dans cette version  
    private TextField top2; // pour rentrer le deuxieme nombre entier  
    private TextField tRes; // pour afficher le resultat  
    private TextField tErr; // pour les messages d'erreur  
    public Calc1(){  
        super("somme de deux entiers (Calc1)");  
        top1 = new TextField(5); // créent les composants pour rentrer  
        top2 = new TextField(5); // les deux nombres (entiers)  
        tRes = new TextField(5); // crée le composant pour le résultat  
        bFin = new Button("Fin"); // créent  
        bCalcul = new Button(" = "); // les boutons  
        lab = new Label (" + "); // et l'étiquette  
    }  
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.6.35

Calc1 (suite)

```
// avec un champ pour les messages d'erreur  
this.setLayout(new BorderLayout()); // p et tErr en positions N et S  
Panel p = new Panel();  
p.setLayout(new FlowLayout()); // les composants seront  
p.add(top1);p.add(lab);p.add(top2); // mis a la suite  
p.add(bCalcul); p.add(tRes); p.add(bFin); // les uns des autres dans p  
this.add("North",p);  
this.tErr = new TextField();  
this.add("South",tErr); this.tErr.setVisible(false);  
  
this.bCalcul.addActionListener(this); // évènements créés au click  
this.bFin.addActionListener(this); // définis dans actionPerformed  
this.addWindowListener(new GestionnaireFermeture(this)); // pour  
// fermer la fenêtre
```

Programmation Java 2003/2004 - D.Pastre

Chap.6.36

```

public void actionPerformed(ActionEvent e){
    if (e.getSource() == bFin) {System.exit(0);};
    if (e.getSource() == bCalcul){
        try {
            int i1= Integer.parseInt(top1.getText()); // chaine -> entier
            int i2 = Integer.parseInt(top2.getText());
            tRes.setText(""+(i1+i2));
            this.tErr.setVisible(false);this.pack();
        } catch (Exception ex) {
            System.out.println("vous devez rentrer des entiers");
            tRes.setText("???");
            this.tErr.setVisible(true);
            this.tErr.setText("vous devez rentrer des entiers");
            this.pack();
        }
    }
}

```

Calc1 (fin)

Programmation Java 2003/2004 - D.Pastre

Chap.6.37

```

import java.awt.Frame;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener; // 1ère version
public class GestionnaireFermeture implements WindowListener {
    private Frame f;
    public GestionnaireFermeture(Frame f) {this.f = f;}
    public void windowClosing(WindowEvent e) {f.setVisible(false);}

    public void windowActivated(WindowEvent e) {}    public void windowDeactivated(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}      public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {} public void windowOpened(WindowEvent e) {}
}

```

**Gestionnaire de fermeture**

```

import java.awt.event.WindowAdapter; // 2ème version
public class GestionnaireFermeture extends WindowAdapter {
    private Frame f;
    public GestionnaireFermeture(Frame f) {this.f = f;}
    public void windowClosing(WindowEvent e) {f.setVisible(false);}
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.6.38

### **Calc2**

- des cases à cocher pour choisir l'opération à effectuer
- des Panel pour agencer les champs et les cases

### **Calc3**

- des étiquettes

### **Calc4**

- calculs sur des rationnels

### **Calc2**

```
...
import java.awt.Checkbox;
import java.awt.CheckboxGroup;
public class Calc2 extends Frame implements ActionListener {
...
    private CheckboxGroup cgr = new CheckboxGroup();
    private Checkbox cPlus = new Checkbox("+",cgr,true);
    private Checkbox cMult = new Checkbox("*",cgr,false);
    public Calc2(){
        super("somme ou produit de deux entiers (Calc2)");
        ...
        Panel oper = new Panel();
        // les deux boutons + et * a cocher l'un au dessous de l'autre
        oper.setLayout(new GridLayout(2,1)); // cad 2 lignes 1 colonne
        oper.add(cPlus);oper.add(cMult);
    }
```

### Calc2 (suite)

```
this.setLayout(new BorderLayout());
Panel p = new Panel();
p.setLayout(new FlowLayout());
p.add(top1); p.add(oper); p.add(top2); // + ou *
p.add(bCalcul); p.add(tRes); p.add(bFin);
this.add("North",p);
this.tErr = new TextField();
this.add("South",tErr); this.tErr.setVisible(false);
...
```

### Calc2 (fin)

```
public void actionPerformed(ActionEvent e){
    if (e.getSource() == bFin) {System.exit(0);};
    if (e.getSource() == bCalcul){
        try {
            int i1= Integer.parseInt(top1.getText());
            int i2 = Integer.parseInt(top2.getText());
            if (cgr.getSelectedCheckbox()==cPlus) tRes.setText(""+(i1+i2));
                else tRes.setText(""+(i1*i2));
            this.tErr.setVisible(false);this.pack();
        }
        catch (Exception ex) {...}
    }
}
```

## java.awt

### Classes et méthodes

Programmation Java 2003/2004 - D.Pastre

Chap.6.43

- **pack()** *méthode de **Window** héritée par **Frame** et **Dialog***  
causes this Window to be sized to fit the preferred size and layouts of its subcomponents.
- **show()** *méthode de **Window** et **Dialog***  
Makes the Window/Dialog visible.
- **add(Component)**  
*méthode de **Container** héritée par **Frame**, **Panel**, **Applet**, ...*  
Add the specified component to this container.
- **setLayout(LayoutManager)**  
Sets the layout manager for this container
- **setVisible(boolean)**  
*méthode de **Component** héritée par tous les composants*  
Shows or hides this component depending on the value of parameter.

Programmation Java 2003/2004 - D.Pastre

Chap.6.44

*classes implémentant l'interface* **LayoutManager**

**FlowLayout** : les composants sont mis à la suite les uns des autres avec passage à la ligne si nécessaire (par défaut pour les Panel)

**BorderLayout** : 5 positions East, West, North, South, Center (par défaut pour les Frame)

**GridLayout** : grille de l lignes et c colonnes

- **addActionListener(ActionListener)**

*méthode de **Button**, **TextField** et **MenuItem***

Add the specified action listener to receive action events from this button/textfield/menu item.

- **actionPerformed(ActionEvent)** *méthode de **ActionListener***

Invoked when an action occurs.

- **getSource()** *méthode de **EventObject***

The object on which the Event initially occurred.

- **getText()** *méthode de **TextComponent** et **Label***

Gets the text that is presented by this text component / label

- **setText(String)** *méthode de **Label**, **TextComponent** et **TextField***

Sets the text that is presented by this label / text component to be the specified text.

- **addWindowListener**(WindowListener)

*méthode de **Window** héritée par **Frame** et **Dialog***

Adds the specified window listener to receive window events from this window.

- **windowClosing**(WindowEvent)

*méthode abstraite de l'interface **WindowListener***

*méthode concrète vide de **WindowAdapter***

Invoked when a window is in the process of being closed.

### Cases à cocher

- *classes*

- **Checkbox** is a graphical component that can be in either an "on" (true) or "off" (false) state.
- **CheckboxGroup** is used to group together a set of Checkbox buttons.

*Constructeurs et méthode*

- **Checkbox()** Creates a check box with no label.
- **Checkbox(String)** Creates a check box with the specified label.
- **Checkbox(String, boolean)**  
Creates a check box with the specified label and sets the specified state.
- **Checkbox(String, boolean, CheckboxGroup)** *ou*
- **Checkbox(String, CheckboxGroup, boolean)**  
Creates a check box with the specified label, in the specified checkbox group, and set to the specified state.
- **getSelectedCheckbox()**  
Gets the current choice from this check box group



## Les menus

- **MenuBar**

encapsulates the platform's concept of a menu bar bound to a frame.

- **Menu**

is a pull-down menu component that is deployed from a menu bar.

**Menu**(String) constructs a new menu with the specified label.

- **MenuItem**

All items in a menu must belong to the class MenuItem, or one of its subclasses.

**MenuItem**(String) constructs a new MenuItem with the specified label and no keyboard shortcut.

**MenuItem**(String, MenuShortcut)

creates a menu item with an associated keyboard shortcut.

## Autres

- **getLabel()** *méthode de **Button**, **MenuItem** et **Checkbox***

Gets the label for this button / menu item / check box

- **setLabel(String)** *méthode de **Button**, **MenuItem** et **Checkbox***

Sets the label for this button / menu item / check box to be the specified label

- **getActionCommand()** *méthode de **Button** et **MenuItem***

Gets the command name of the action event that is fired by this Button / menu item.

- **getActionCommand()** *méthode de **ActionEvent***

Returns the command string associated with this action.

- **setActionCommand(String)** *méthode de **Button** et **MenuItem***

Sets the command name of the action event that is fired by this button/menu item.

- **setEditable(boolean)** *méthode de **TextComponent***

Sets the flag that determines whether or not this text component is editable.

### Boîtes de dialogue : fenêtres temporaires sans barre de menus

- **Dialog**

A Dialog is a top-level window with a title and a border that is typically used to take some form of input from the user.

#### *Constructeurs*

- **Dialog(Frame)** Constructs an initially invisible, non-modal Dialog with an empty title and the specified owner frame.
- **Dialog(Frame, boolean)** Constructs an initially invisible Dialog with an empty title, the specified owner frame and modality.
- **Dialog(Frame, String)** Constructs an initially invisible, non-modal Dialog with the specified owner frame and title.
- **Dialog(Frame, String, boolean)** Constructs an initially invisible Dialog with the specified owner frame, title, and modality.

- **FileDialog**

Displays a dialog window from which the user can select a file.

#### *Constructeurs*

- **FileDialog(Frame)**  
Creates a file dialog for loading a file.
- **FileDialog(Frame, String)**  
Creates a file dialog window with the specified title for loading a file.
- **FileDialog(Frame, String, int)**  
Creates a file dialog window with the specified title for loading or saving a file.