

6. Les composants Swing

- 6.1. JOptionPane

La gestion des évènements

3 intervenants (càd 3 objets différents) dans la gestion d'évènement:

1. Le **composant à écouter** (ex: un bouton)
2. L'**écouteur d'évènement** (ex: l'objet qui écoute le bouton)
3. L'**objet de type d'évènement** créé par java quand l'évènement a lieu
(ex: l'objet créé quand on clique sur le bouton)

A l'exécution, lors de l'évènement,

Java crée automatiquement 1 **objet de type Evenement**

qui contient des *infos sur l'évènement*

(ex: si l'utilisateur a coché ou décoché une case à cocher)

Cet objet est un objet d'une classe prédéfinie

qui est une *sous-classe de la classe Event*

Programmeur

1. Le programmeur **crée** le **composant à écouter** (*ex: un bouton*)
2. Le programmeur **crée** une **classe écouteur d'événement**

Il écrit le code de la **réaction** souhaitée dans une **méthode** de cette classe

Pour ce faire, il peut utiliser les informations stockées dans l'objet de type événement créé par Java

3. Le programmeur **crée** un **objet** de cette classe écouteur
4. Le programmeur **associe** cet **objet écouteur** au **composant à écouter**

Java

A l'exécution, lors de l'évènement (*ex: si clic sur le bouton*):

1. Java vérifie si le **composant** sur lequel a eu lieu l'évènement est écouté;
Si oui, il retrouve l'**objet écouteur d'évènement** associé
2. Java retrouve la **classe correspondant à cet objet écouteur**
3. Java **appelle automatiquement** la **méthode** de la **classe écouteur d'évènement** correspondant à la réaction à l'évènement

À condition que la méthode soit correctement déclarée!!!

classe écouteur d'événement



doit implémenter le bon interface!



Contient les **déclarations des méthodes** appelées
automatiquement par java en réaction aux
évènements



Syntaxe à respecter!

Classe interne

```
class MaClasseEnglobante
```

```
{ private ..... Variables d'instance  
..... Constructeurs et méthodes }
```

```
private class MaClasseInterne  
    { Variables d'instance  
      Constructeurs et méthodes  
    }
```

accès

```
}
```

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)


```
import javax.swing.*;
```

```
public class Fenetre extends JFrame
```

```
{
```

```
    public Fenetre( )
```

```
    { super("Titre de ma fenetre");
```

```
        setBounds(100,100,500,500);
```

Classe écouteur d'évènement

Créer un objet écouteur d'évènement

```
        MyWindowListener w = new MyWindowListener( );
```

```
        this.addWindowListener(w);
```

Associer l'écouteur d'évènement
au composant à écouter: ici, la
fenêtre courante

```
        setVisible(true);
```

```
    }
```

```
}
```

```
public class Fenetre extends JFrame
```

```
{
```

```
    public Fenetre( )
```

```
    {
```

```
        super("Titre de ma fenetre");
```

```
        setBounds(100,100,500,500);
```

```
        MyWindowListener w = new MyWindowListener ( );
```

```
        this.addWindowListener(w);
```

```
        setVisible(true);
```

```
    }
```

```
private class MyWindowListener extends WindowAdapter
```

```
{
```

```
    public void windowClosing( WindowEvent e)
```

```
    {System.exit(0);}
```

```
}
```

```
}
```

import javax.swing.*;


import java.awt.event.*;

classe qui implémente
l'interface WindowListener

classe interne

méthode appelée automatiquement
lors de la fermeture de la fenêtre:

Déclaration à respecter!!!

```
public abstract class WindowAdapter implements WindowListener, . . .
{
    public void windowClosing (WindowEvent e) { }  Implémentation vide

    public void windowOpened (WindowEvent e) { }

    public void windowClosed (WindowEvent e) { }

    public void windowIconified (WindowEvent e) { }

    public void windowDeiconified (WindowEvent e) { }

    public void windowActivated (WindowEvent e) { }

    public void windowDeactivated (WindowEvent e) { }
}
```

```
public class Principal
{
    public static void main (String[ ] args)
    {
        Fenetre f = new Fenetre() ;
    }
}
```

```
import ...
```

```
public class Fenetre extends JFrame
```

```
{
```

```
    public Fenetre( )
```

```
    { super("Titre de ma fenetre");
```

```
      setBounds(100,100,500,500);
```

```
      this.addWindowListener(new WindowAdapter()
```

```
      { public void windowClosing( WindowEvent e)
```

```
        { System.exit(0);  }
```

```
      }
```

```
    );
```

```
    setVisible(true);
```

```
}
```

```
}
```

Crée une occurrence
d'une sous-classe anonyme
de la classe WindowAdapter

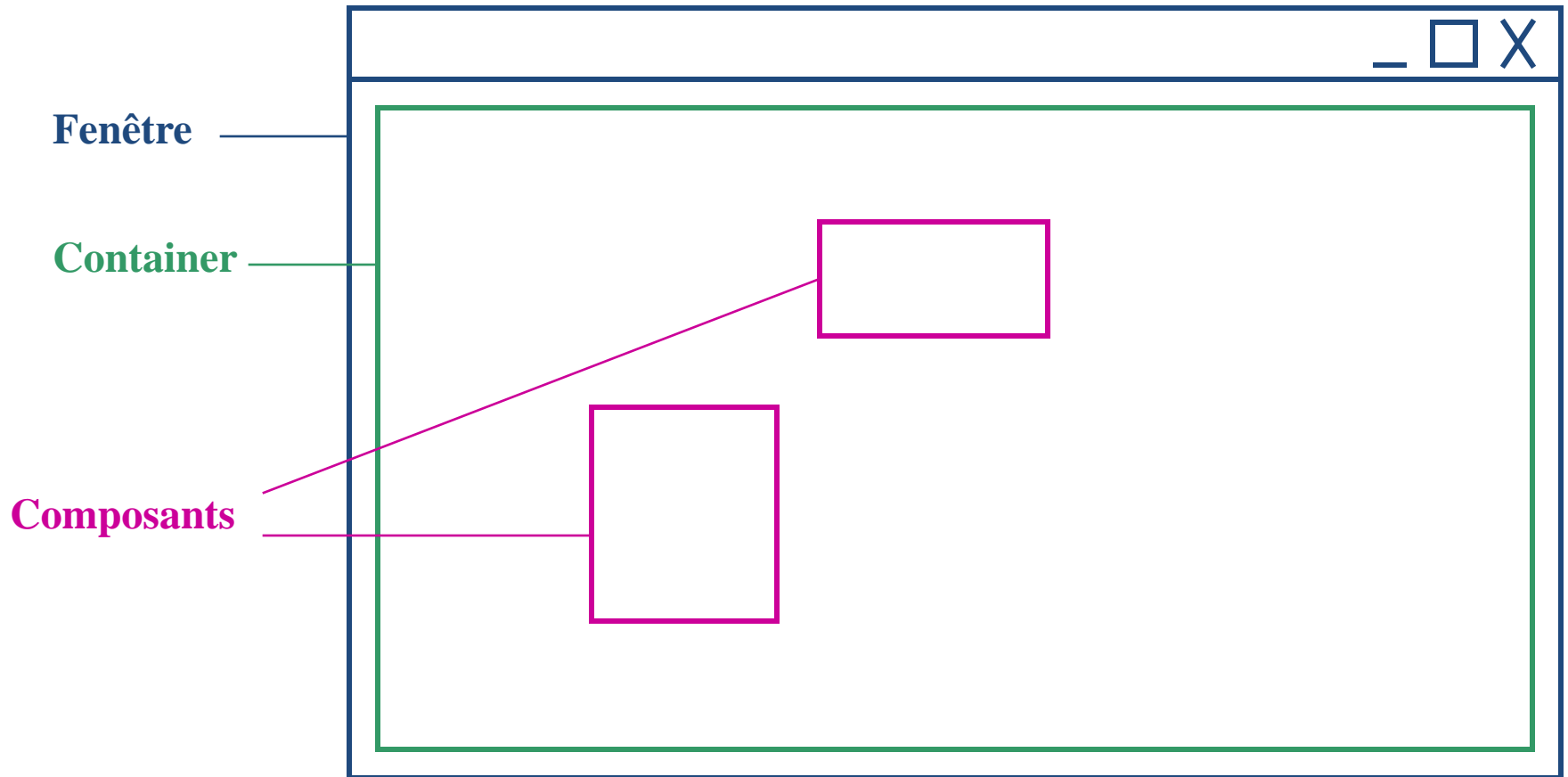


Redéfinition de la
méthode
windowClosing(...)
héritée

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)

Ex. 1: Sans panneau



...

```
public class Fenetre extends JFrame
```

```
{ private Container cont;  
  private JLabel label;
```

} Variables d'instance privées

Etiquette

Container de la fenêtre

```
public Fenetre( )
```

```
{ super("Titre de ma fenetre");
```

```
  setBounds(100,100,500,500);
```

```
  addWindowListener( new WindowAdapter()
```

```
    {public void windowClosing( WindowEvent e) {System.exit(0);} } );
```

```
  label = new JLabel("Bienvenue");
```

Bulle d'aide

```
  label.setTooltipText("Ceci est un message de bienvenue");
```

```
  cont = getContentPane( );
```

Récupère le container associé à la fenêtre

```
  cont.setLayout(new FlowLayout( ));
```

```
  cont.add(label);
```

associe un gestionnaire de tracé au container

```
  setVisible(true);
```

Ajoute le composant au container

```
} }
```


...

```
public class Fenetre extends JFrame
```

```
{ private Container cont;
```

```
    private JLabel label;
```

```
public Fenetre( )
```

```
{ super("Titre de ma fenetre");
```

```
  setBounds(100,100,500,500);
```

```
  addWindowListener( new WindowAdapter()
```

```
    {public void windowClosing( WindowEvent e) {System.exit(0);}});
```

```
  label = new JLabel("Bienvenue");
```

```
  label.setToolTipText("Ceci est un message de bienvenue");
```

```
  cont = getContentPane( );
```

```
  cont.setLayout(new FlowLayout( ));
```

```
  cont.add(label);
```

```
  setVisible(true);
```

```
} }
```

instruction obligatoire!?!
↑

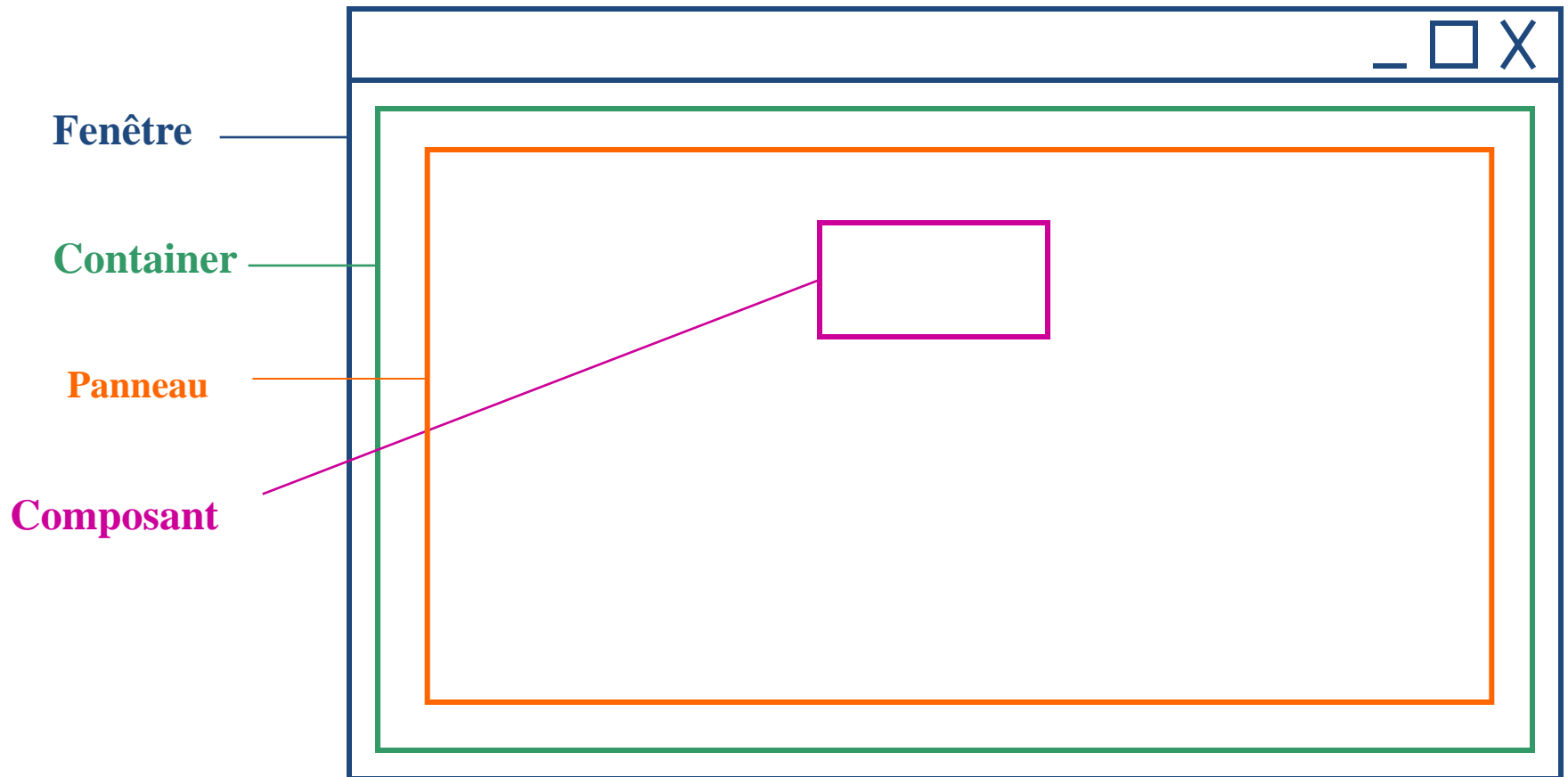
cont.setLayout(*null*);

*label.set**Bounds*(200,200,100,50);

↓
OK seulement si aucun
gestionnaire de tracé

soit | soit

Ex. 2: Avec 1 panneau: une seule classe (sous-classe de JFrame)



```
public class Fenetre extends JFrame
{ private Container cont;
  private JPanel panneau;
  private JLabel label;
```

```
public Fenetre ( )
{ super("Bienvenue");
  setBounds(100,100,500,500);
  addWindowListener( ...);
```

import java.awt.Color;

dessine une bordure bleue autour du panneau

```
panneau = new JPanel ( );
panneau.setBounds(50,50,380,200);
panneau.setBorder (BorderFactory.createLineBorder (Color.BLUE) );
```

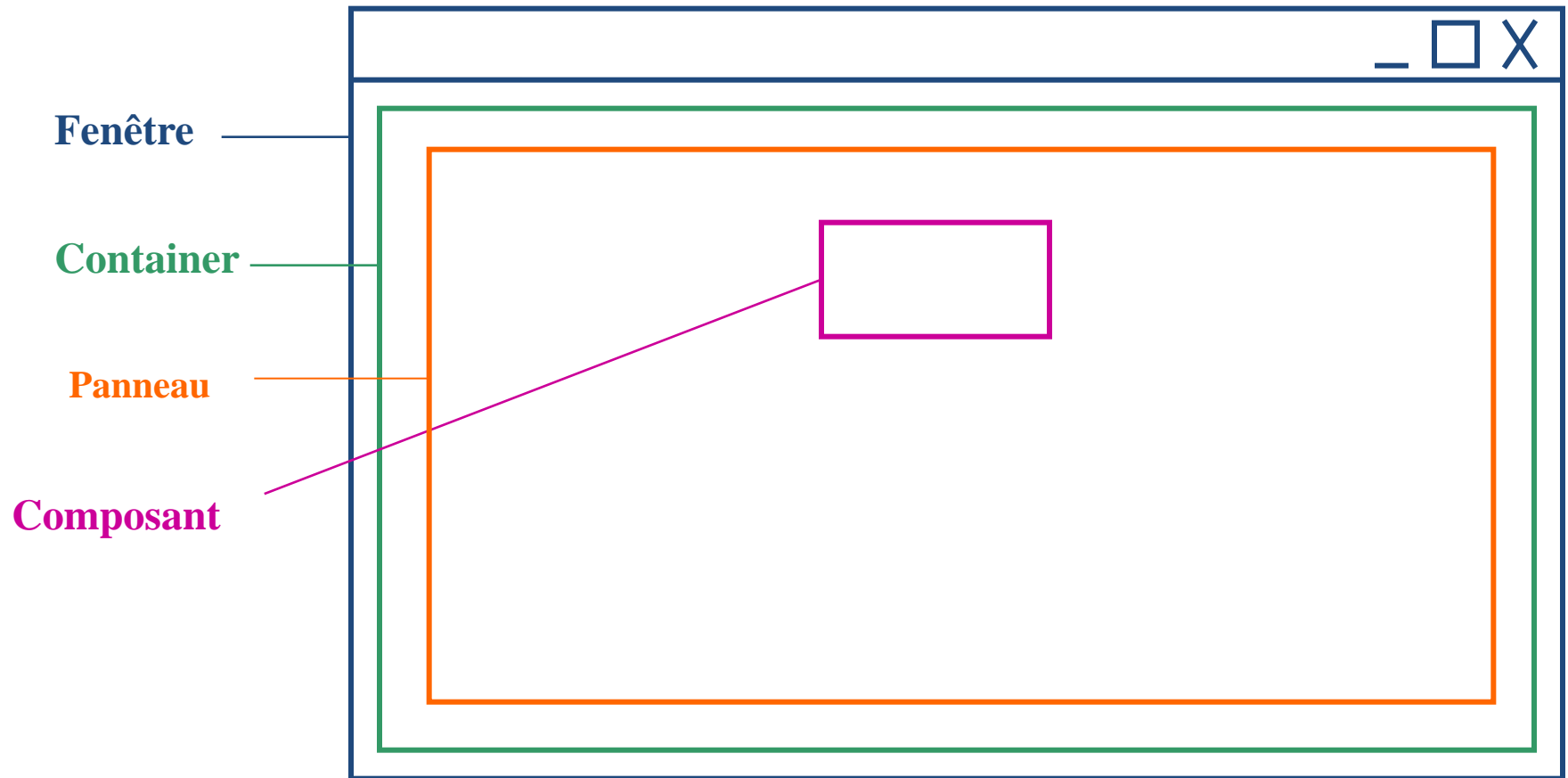
```
label = new JLabel("Bienvenue à l'IESN");
label.setBounds(150,50,300,100);
```

```
panneau.setLayout(null);
panneau.add(label);
```

```
cont = getContentPane();
cont.setLayout(null);
cont.add ( panneau );
setVisible(true); } }
```

Ex. 3: Avec 1 panneau:

2 classes (1 sous-classe de JFrame) et (1 sous-classe de JPanel)



```
import javax.swing.*;

public class PanelBienvenue extends JPanel

{private JLabel label;

    PanelBienvenue( )

    { this.setBounds(50,50,380,200);
      this.setBorder(BorderFactory.createLineBorder(Color.BLUE));

      label = new JLabel("Bienvenue à l'IESN");
      label.setBounds(150,50,300,100);

      this.setLayout(null);
      this.add(label);
    }
}
```

```
public class Fenetre extends JFrame
```

```
{  private Container cont;  
    private PanelBienvenue panneau;
```

```
    public Fenetre ( )
```

```
    {  
        super("Bienvenue");  
        setBounds(100,100,500,500);  
        addWindowListener( ...);
```

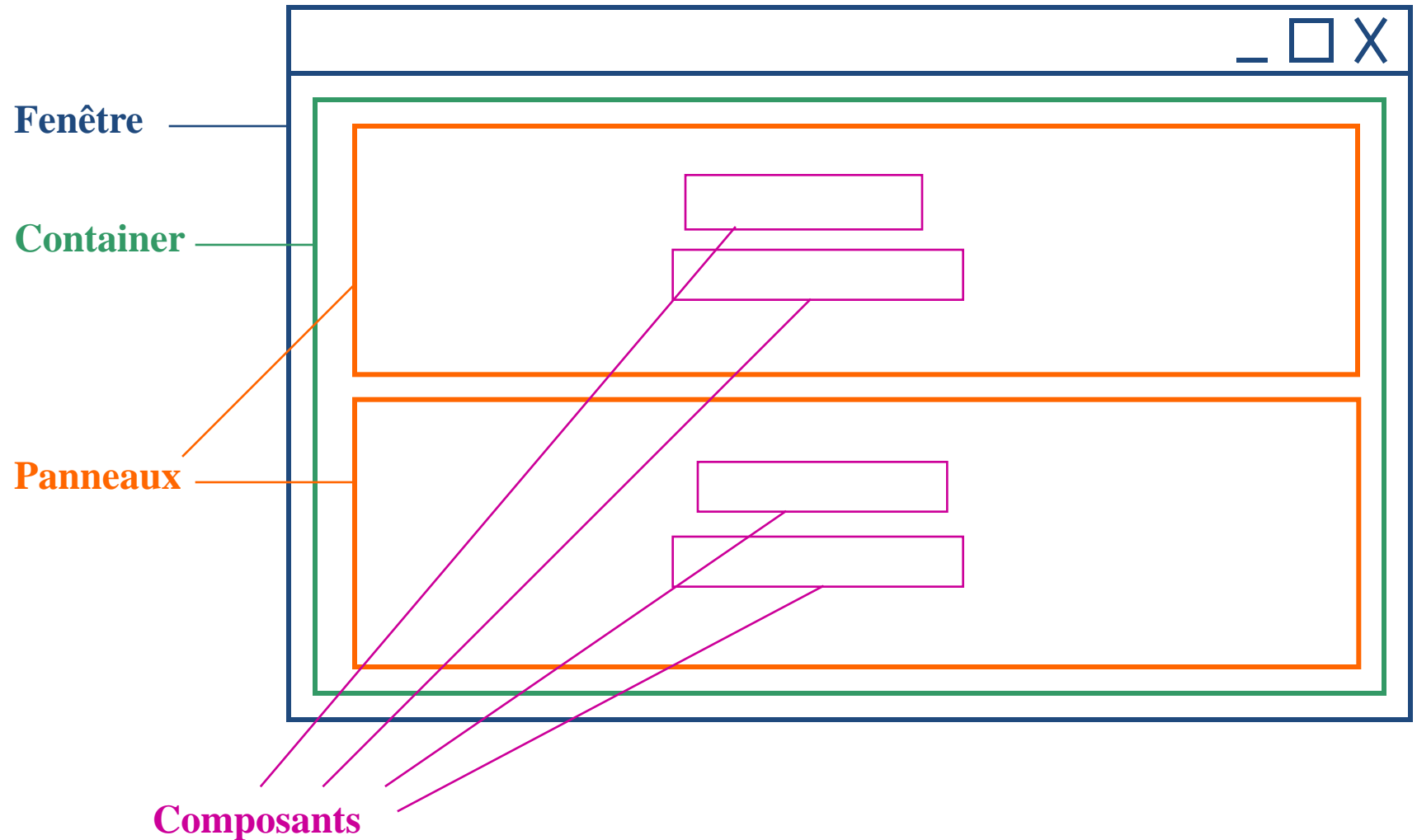
```
        panneau = new PanelBienvenue( );
```

```
        cont = getContentPane();  
        cont.setLayout(null);  
        cont.add ( panneau );
```

```
        setVisible(true);
```

```
    }  
}
```

Ex. 4: Avec 2 panneaux: une seule classe (sous-classe de JFrame)



```
public class Fenetre extends JFrame
```

```
{  private Container cont;  
    private JPanel panPrive, panProf;  
    private JLabel nomPrive, adrPrive, nomProf, adrProf;
```

```
public Fenetre ( )  
{  
    super("Coordonnées");  
    setBounds(100,100,500,600);  
    addWindowListener( ... );
```

```
    panPrive = new JPanel();  
    panPrive.setBounds(50,50,380,200);  
    panPrive.setBorder(BorderFactory.createTitledBorder("Coordonnées privées"));
```

Affiche un titre au panneau



```
    nomPrive = new JLabel("Pierre Dupond");  
    nomPrive.setBounds(150,40,300,100);  
    adrPrive = new JLabel("1, rue de fer, 5000, Namur");  
    adrPrive.setBounds(100,60,300,100);
```

```
    panPrive.setLayout(null);  
    panPrive.add (nomPrive);  
    panPrive.add (adrPrive);
```



```

public class Fenetre extends JFrame
{
    ...
    private JPanel panPrive, panProf;
    private JLabel nomPrive, adrPrive, nomProf, adrProf;

    public Fenetre ( )
    {
        ...
        panProf = new JPanel();
        panProf.setBounds(50,300,380,200);
        panProf.setBorder(BorderFactory.createTitledBorder("Coordonnées professionnelles"));

        nomProf = new JLabel("InfoBel S.A.");
        nomProf.setBounds(150,40,300,100);
        adrProf = new JLabel("10, rue haute, 1000 Bruxelles");
        adrProf.setBounds(100,60,300,100);

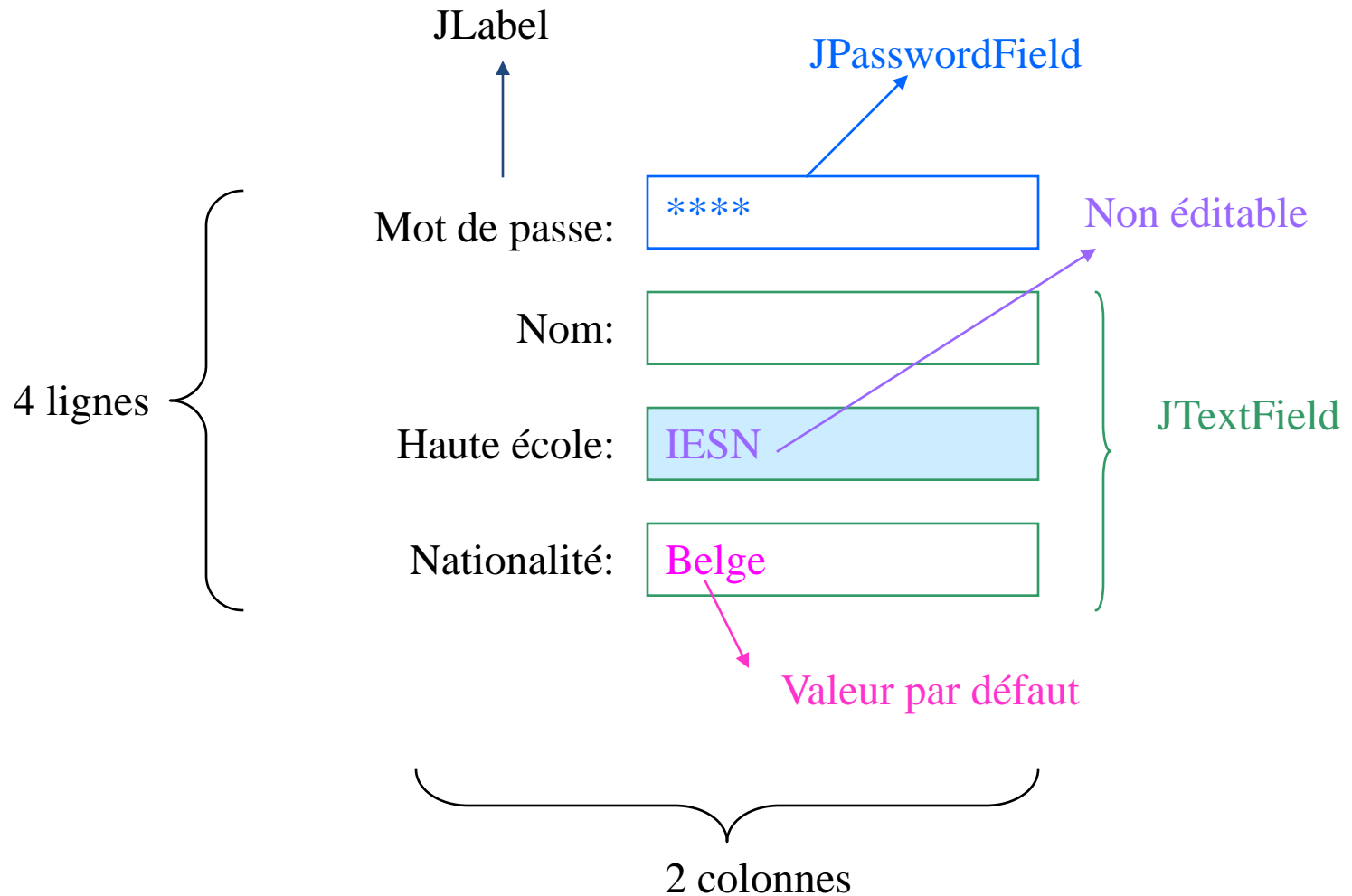
        panProf.setLayout(null);
        panProf.add (nomProf);
        panProf.add (adrProf);

        cont = getContentPane();
        cont.setLayout(null);
        cont.add (panPrive);
        cont.add (panProf) ;
        setVisible(true); } }

```

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
 - 6.4.1. JLabel
 - 6.4.2. JTextField
 - 6.4.3. JPasswordField



gestionnaire de tracé: ***GridLayout***

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

public class Fenetre extends JFrame {

    private Container cont;
    private JLabel labelMotPasse, labelNom, labelEcole, labelNationalite;
    private JTextField zoneTexteNom, zoneTexteEcole, zoneTexteNationalite;
    private JPasswordField zoneMotPasse;

    public Fenetre( ) {
        super("Inscription IESN");
        setBounds(200,200,500,150);

        addWindowListener( ... );

        cont = getContentPane( );
    }
}
```

```
...  
cont.setLayout(new GridLayout(4,2,5,5));
```

```
labelMotPasse = new JLabel("Mot de passe: ");  
cont.add(labelMotPasse);  
zoneMotPasse = new JPasswordField(20);  
cont.add(zoneMotPasse);
```

```
labelNom = new JLabel("Nom: ");  
cont.add(labelNom);  
zoneTexteNom = new JTextField(30);  
cont.add(zoneTexteNom);
```

```
labelEcole = new JLabel("Haute école: ");  
cont.add(labelEcole);  
zoneTexteEcole = new JTextField("IESN",30);  
zoneTexteEcole.setEditable(false);
```

```
cont.add(zoneTexteEcole);
```

Non éditable

```
labelNationalite = new JLabel("Nationalite: ");  
cont.add(labelNationalite);  
zoneTexteNationalite = new JTextField("Belge",30);  
cont.add(zoneTexteNationalite);
```

```
setVisible(true);
```

```
}
```

Valeur d'initialisation

```
public class Fenetre extends JFrame
```

```
{ private JTextField zoneTexteNom, zoneTexteNationalite, ...;
```

```
...
```

```
public Fenetre ( )
```

```
{...
```

```
MonGestionnaireAction g = new MonGestionnaireAction( );
```

```
zoneTexteNom.addActionListener(g);
```

```
zoneTexteNationalite.addActionListener(g);
```

```
...
```

```
}
```

```
}
```

classe écouteur d'évènement

objet écouteur d'évènement

*associer l'écouteur
d'évènement aux
composants à écouter*

```
public class Fenetre extends JFrame
```

```
{ private JTextField zoneTexteNom, zoneTexteNationalite, ...;
```

```
...
```

```
public Fenetre ( )
```

```
{...
```

```
MonGestionnaireAction g = new MonGestionnaireAction( );
```

```
zoneTexteNom.addActionListener(g);
```

```
zoneTexteNationalite.addActionListener(g);
```

```
...
```

```
}
```

```
private class MonGestionnaireAction implements ActionListener
```

```
{
```

```
public void actionPerformed (ActionEvent e) {
```

```
if (e.getSource( ) == zoneTexteNom)
```

```
    JOptionPane.showMessageDialog(null,"Bienvenue à "+e.getActionCommand( ));
```

```
    else JOptionPane.showMessageDialog(null,"Dossier à compléter au secrétariat","",  
                                         JOptionPane.WARNING_MESSAGE);
```

```
}}
```

```
}
```

*objet e de type
ActionEvent: contient
des informations sur
l'événement.*

*Rappel: généré
automatiquement par
Java*

Interface



méthode appelée si enter dans JTextField

retourne la source de l'évènement

retourne la chaîne de caractères entrée

labelMotPasse = new **JLabel**("Mot de passe: ");
labelMotPasse.**setHorizontalAlignment**(**SwingConstants.RIGHT**);  *constante*  *aligne le label à droite*

N.B. Par défaut, alignement à gauche

zoneTexteNom = new **JTextField**(30);
String t = zoneTexteNom.**getText**();  *lire le contenu d'un JTextField*
zoneTexteNom.**setText**("Jules");  *modifier la valeur d'un JTextField*

zoneMotPasse = new **JPasswordField**(20);
String t = new String(zoneMotPasse.**getPassword**());  *lire un mot de passe*

Attention: retourne un tableau de char



à transformer en un String

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
 - 6.4.1. JLabel
 - 6.4.2. JTextField
 - 6.4.3. JPasswordField
 - 6.4.4. JTextArea

```
public class Fenetre extends JFrame {
```

```
    private Container cont;  
    private JTextArea texte;
```

```
    public Fenetre( ) {  
        super("JTextArea");  
        setBounds(100,100,500,300);
```

```
        cont = getContentPane( );  
        cont.setLayout(new FlowLayout( ));
```

```
        addWindowListener( new WindowAdapter( )  
            {public void windowClosing( WindowEvent e){ System.exit(0);} } );
```

```
        texte = new JTextArea(5,15);  
        cont.add(new JScrollPane(texte));
```

NB. Un JTextArea pas défilant par défaut!

```
        setVisible(true);
```

défilant

```
    }
```

```
}
```

```
String s = texte.getSelectedText( )
```



*récupère le texte sélectionné par
l'utilisateur*

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
 - 6.5.1. JButton

JLabel ← **Bienvenue**

JButton ← **tel**



Réaction à l'événement:
affichage du numéro de téléphone de l'IESN

```
public class Fenetre extends JFrame
{
    private Container cont;
    private JButton boutonIESN;
    private JLabel bienvenue, telIESN;
```

```
public Fenetre( )
{
    super("Accueil IESN");
    ...
    cont = getContentPane( );
    cont.setLayout(null); ..... ➔ Rappel: obligatoire si setBounds( ...)

    bienvenue = new JLabel("Bienvenue");
    bienvenue.setBounds(200,200,100,30); ←
    bienvenue.setHorizontalAlignment(SwingConstants.CENTER);
                                   centre le label ←

    telIESN = new JLabel("Tel. IESN: 081/46.86.10");

    boutonIESN = new JButton("Tel");
    boutonIESN.setBounds(200,400,100,30); ←

    cont.add(bienvenue);
    cont.add(boutonIESN); } Seulement deux des trois composants ajoutés au container!

    setVisible(true);

}}
```

```
public class Fenetre extends JFrame
{
    private Container cont;
    private JButton boutonIESN;
    private JLabel bienvenue, telIESN;
```

```
public Fenetre( )
```

```
{
    ...
    telIESN = new JLabel("Tel. IESN: 081/72.36.10");
```

→ *créé et initialisé mais pas affiché*

```
    MonGestionnaire g = new MonGestionnaire( );
    boutonIESN.addActionListener(g);
    ...
}
```

→ *crée un écouteur d'évènement*

→ *associe l'écouteur au composant à écouter*

interface

```
private class MonGestionnaire implements ActionListener
{
```

```
    public void actionPerformed( ActionEvent e)
```

→ *appelée si clic sur bouton*

```
    {
```

```
        cont.removeAll( );
```

→ *vide le contenu du container*

```
        telIESN.setBounds(200,200,200,30);
```

```
        cont.add(telIESN);
```

→ *ajoute le label au container*

```
        cont.repaint( );
```

```
        Fenetre.this.setVisible(true);
```

→ *redessine le container!!!*

```
    }
```

```
}}
```

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
 - 6.5.1. JButton
 - 6.5.2. JCheckBox

Quantité:

10

JCheckBox



Quantité pour remise

```

public class Fenetre extends JFrame
{
    private Container cont;
    private JCheckBox default;
    private JTextField texte;
    private JLabel label;

    public Fenetre( )
    {
        super("CheckBox");
        ...

        cont = getContentPane();
        cont.setLayout(new FlowLayout());

        label = new JLabel("Quantité:");
        cont.add(label);
        texte = new JTextField(20);
        cont.add(texte);
        default = new JCheckBox("Quantité pour remise");
        cont.add(default);

        ...
        setVisible(true);
    }
}

```

Nouveau type d'écouteur:

*Interface **ItemListener***

*capable de détecter deux états possibles: **coché** ou **décoché***

```
public class Fenetre extends JFrame
```

```
{ ...
```

```
    private JCheckBox default;
```

```
public Fenetre( )
```

```
{    super("CheckBox");
```

```
    ...
```

```
    default = new JCheckBox("Quantité pour remise");
```

```
    ...
```

```
    MyItemList g = new MyItemList( );
```

```
    default.addItemListener(g);
```

```
    ...
```

```
}
```

Interface

```
private class MyItemList implements ItemListener
```

```
{
```

```
    public void itemStateChanged(ItemEvent e)
```

```
    { if ( e.getStateChange( ) == ItemEvent.SELECTED )
```

```
        texte.setText("10");
```

```
    else    texte.setText("");
```

```
    }
```

```
} }
```

crée un écouteur d'évènement

associe l'écouteur au composant à écouter



coché (DESELECTED: décoché)

retourne l'état du CheckBox

N.B. if (*e.getStateChange()* == *ItemEvent.SELECTED*)

≡

default.isSelected()



return true if selected

return false if not selected



nom du CheckBox

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
 - 6.5.1. JButton
 - 6.5.2. JCheckBox
 - 6.5.3. JRadioButton

☐ Homme

☐ Femme célibataire

☒ Femme mariée

Madame

```
public class Fenetre extends JFrame
{
    private Container cont;
    private JRadioButton bouton1, bouton2, bouton3;
    private ButtonGroup groupeBout;
    private JTextField texte;
```

—————→ *gère le groupe: un seul bouton radio coché à la fois*

```
public Fenetre( )
{
    super("Bienvenue");
    ...
    bouton1 = new JRadioButton("Homme",true); ———→ coché
    cont.add(bouton1);
    bouton2 = new JRadioButton("Femme Célibataire",false); ———→ décoché
    cont.add(bouton2);
    bouton3 = new JRadioButton("Femme mariée",false); ———→ décoché
    cont.add(bouton3);
    texte = new JTextField("Monsieur",20);
    cont.add(texte);

    groupeBout = new ButtonGroup();
    groupeBout.add(bouton1);
    groupeBout.add(bouton2);
    groupeBout.add(bouton3);
    ...
    setVisible(true);
}
```

① *ajouter les boutons au container*

② *ajouter les boutons au ButtonGroup*


```

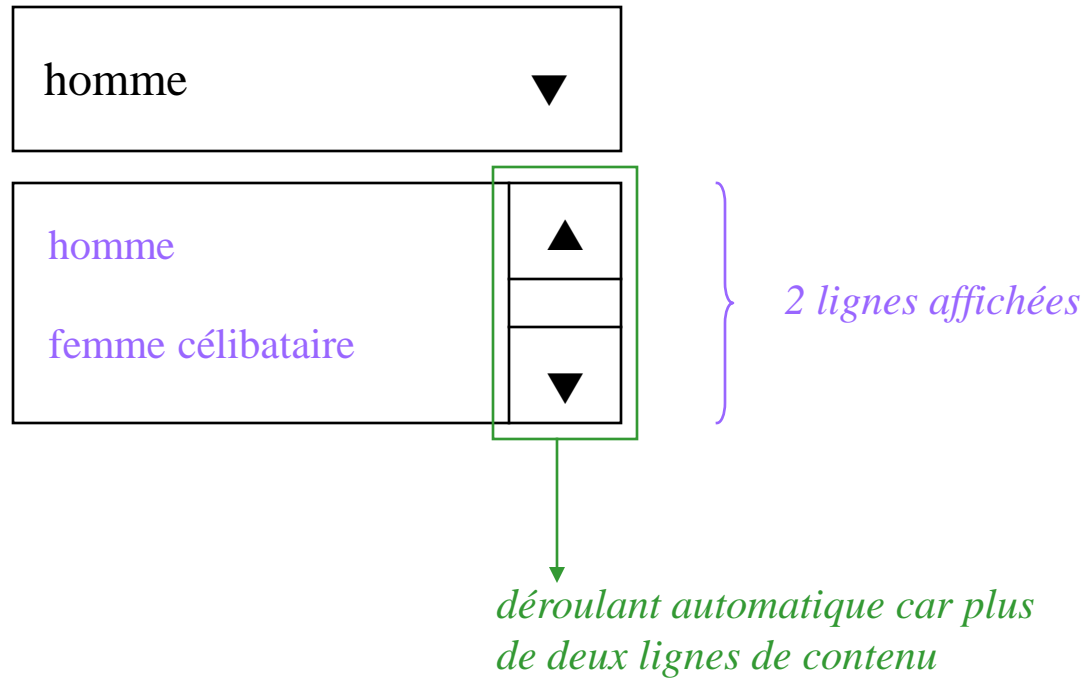
public class Fenetre extends JFrame
{
    private JRadioButton bouton1, bouton2, bouton3;
    private ButtonGroup groupeBout;
    ...
    public Fenetre( )
    {
        ...
        MyItemList g = new MyItemList( );
        bouton1.addItemListener(g);
        bouton2.addItemListener(g);
        bouton3.addItemListener(g);
        ...
    }
    private class MyItemList implements ItemListener
    {
        public void itemStateChanged( ItemEvent e)
        {
            if (e.getSource( ) == bouton1 && e.getStateChange( ) == ItemEvent.SELECTED)
                texte.setText("Monsieur");
            else if (e.getSource( ) == bouton2 && e.getStateChange( ) == ItemEvent.SELECTED)
                texte.setText("Mademoiselle");
            else if (e.getSource( ) == bouton3 && e.getStateChange( ) == ItemEvent.SELECTED)
                texte.setText("Madame");
        }
    }
}

```

crée un écouteur d'évènement
associe l'écouteur aux composants à écouter
Interface
appelée si un RadioButton (dé)coché
obligatoire, sinon réaction aussi quand décoché

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
- 6.6. Listes
 - 6.6.1. JComboBox



```
public class Fenetre extends JFrame
{ private Container cont;
  private JComboBox combox;
```

```
public Fenetre( )
{ ...
```

```
String[ ] contenu = {"homme", "femme célibataire", "femme mariée"};
```

```
combox = new JComboBox(contenu);
```

```
combox.setSelectedItem("homme");
```

—————→ *valeur sélectionnée par défaut*

```
combox.setMaximumRowCount(2);
```

—————→ *deux lignes du contenu affichées*

```
cont.add(combox);
```

↓
or, trois valeurs possibles

↓
déroulant automatique

```
...
}
```

```
combo.setEditable(true);
```



*L'utilisateur peut entrer une
valeur autre que celles proposées*

```

public class Fenetre extends JFrame
{
    private Container cont;
    private JComboBox combobox;

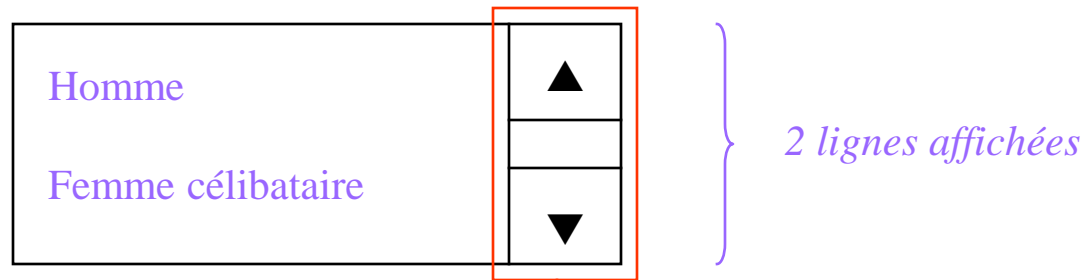
    public Fenetre( )
    {
        ...
        combobox = new JComboBox(contenu);
        ...
        MonGestionnaire g = new MonGestionnaire( );
        combobox.addItemListener(g);
        ...
    }

    private class MonGestionnaire implements ItemListener
    {
        public void itemStateChanged( ItemEvent e)
        {
            switch (combobox.getSelectedIndex( ) )
            {
                case 0: if ( e.getStateChange( ) == ItemEvent.SELECTED )
                {
                    JOptionPane.showMessageDialog(null,"Bonjour Monsieur");
                    break;
                }
                case 1: if (e.getStateChange( ) == ItemEvent.SELECTED)
                {
                    JOptionPane.showMessageDialog(null,"Bonjour Mademoiselle");
                    break;
                }
                case 2: if (e.getStateChange( ) == ItemEvent.SELECTED)
                {
                    JOptionPane.showMessageDialog(null,"Bonjour Madame");
                    break;
                }
            }
        }
    }
}

```

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
- 6.6. Listes
 - 6.6.1. JComboBox
 - 6.6.2. JList
 - A sélection simple



déroulant pas automatique sur JList


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

```
public class Fenetre extends JFrame
{ private Container cont;
  private JList listeSimple;
```

```
public Fenetre( )
{ ...
  String[ ] contenu = {"Homme","Femme célibataire","Femme mariée"};
```

```
  listeSimple = new JList(contenu);
```

```
  listeSimple.setVisibleRowCount(2);
```

→ *deux lignes du contenu affichées*

```
  listeSimple.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

↓
détermine le type de sélection: l'utilisateur ne peut sélectionner qu'une seule option dans la JList

```
  cont.add(new JScrollPane(listeSimple));
```

↓
créer explicitement un déroulant

```
  ...
```

```
}
```

```
}
```

Attention: on place le déroulant dans le container et pas la JList!

Nouveau type d'écouteur:

*Interface **ListListener***

*capable de détecter la **sélection de une ou plusieurs valeurs**
dans une liste*

```
public class Fenetre extends JFrame
```

```
{ private Container cont;  
  private JList listeSimple;
```

```
public Fenetre( )
```

```
{ ...
```

```
  listeSimple = new JList(contenu);
```

```
  ...
```

```
  MonGestionnaire g = new MonGestionnaire(); —————→ crée un écouteur d'évènement
```

```
  listeSimple.addListSelectionListener(g); —————→ associe l'écouteur au composant à écouter
```

```
  ...
```

```
}
```

Interface

```
private class MonGestionnaire implements ListSelectionListener
```

```
{ public void valueChanged( ListSelectionEvent e) —————→ appelée à chaque nouvelle sélection dans JList
```

```
{ switch (listeSimple.getSelectedIndex( )) —————→ retourne l'index de la valeur sélectionnée
```

```
{ case 0: JOptionPane.showMessageDialog(null,"Bonjour monsieur");
```

```
    break;
```

```
  case 1: JOptionPane.showMessageDialog(null,"Bonjour mademoiselle");
```

```
    break;
```

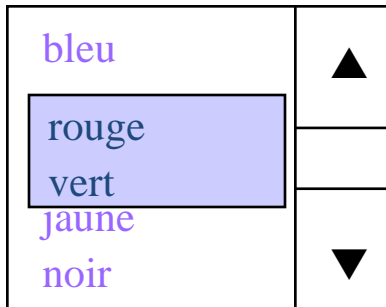
```
  case 2: JOptionPane.showMessageDialog(null,"Bonjour madame");
```

```
    break;
```

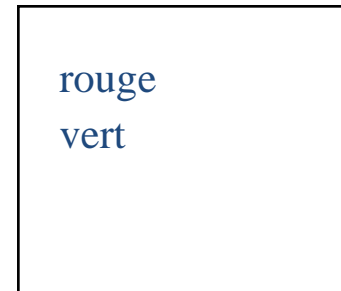
```
} } }
```

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
- 6.6. Listes
 - 6.6.1. JComboBox
 - 6.6.2. JList
 - A sélection simple
 - A sélection multiple



COPIER >>>




```
public class Fenetre extends JFrame
```

```
{ private JList listeCouleurs, listeCopie;  
  private JButton bouton;  
  private Container c;
```

```
  public Fenetre()  
  { ...
```

```
    String[ ] nomCouleurs = {"bleu","rouge","vert","jaune","noir","blanc","violet","rose","gris","brun"};
```

```
    listeCouleurs = new JList(nomCouleurs);
```

```
    listeCouleurs.setVisibleRowCount(5);  cinq lignes affichées
```

```
    listeCouleurs.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
```

 *détermine le type de sélection: l'utilisateur peut sélectionner plusieurs options dans la JList*



```
    c.add(new JScrollPane(listeCouleurs));
```

 *créer explicitement un déroulant*

...

```
listeCopie = new JList( );
```

```
listeCopie.setVisibleRowCount(5);
```



cinq lignes affichées

```
listeCopie.setFixedCellWidth(60);
```



détermine la largeur à l'affichage

```
listeCopie.setFixedCellHeight(15);
```



détermine la hauteur à l'affichage

```
listeCopie.setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
```



l'utilisateur ne peut sélectionner qu'une seule option dans la JList

```
c.add(new JScrollPane(listeCopie));
```

...

obligatoires car liste vide

...

```
bouton = new JButton("COPIER>>>");
```

```
MonGestionnaire g = new MonGestionnaire( );
```

→ crée un écouteur d'évènement

```
bouton.addActionListener(g);
```

→ associe l'écouteur au composant à écouter

```
c.add(bouton);
```

...

```
}
```

interface

```
private class MonGestionnaire implements ActionListener
```

```
{ public void actionPerformed( ActionEvent e)
```

→ appelée si clic sur bouton

```
{ listeCopie.setListData(listeCouleurs.getSelectedValues( ));
```

```
c.repaint( );
```

```
Fenetre.this.setVisible(true);
```

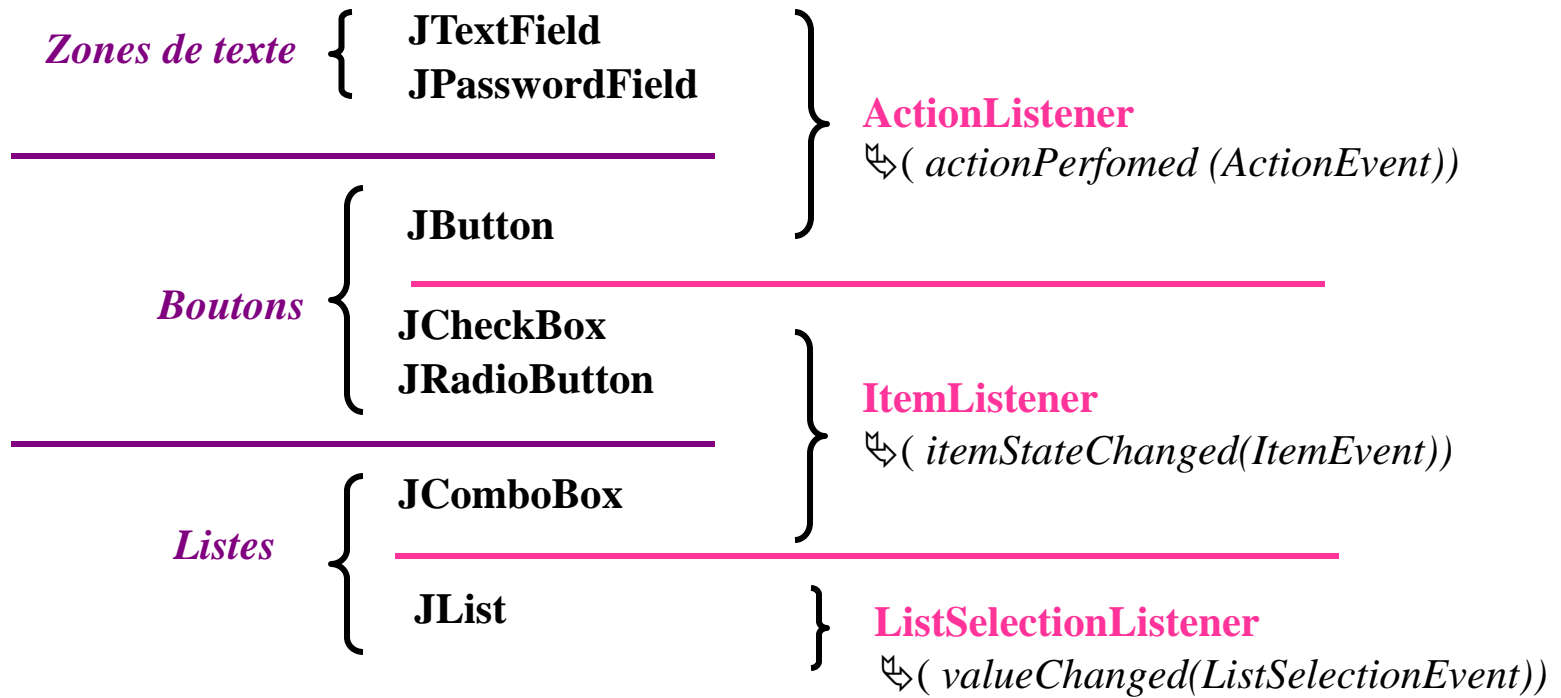
écrit dans la liste

récupère les valeurs sélectionnées

```
}  
}  
}  
pour redessiner le container
```

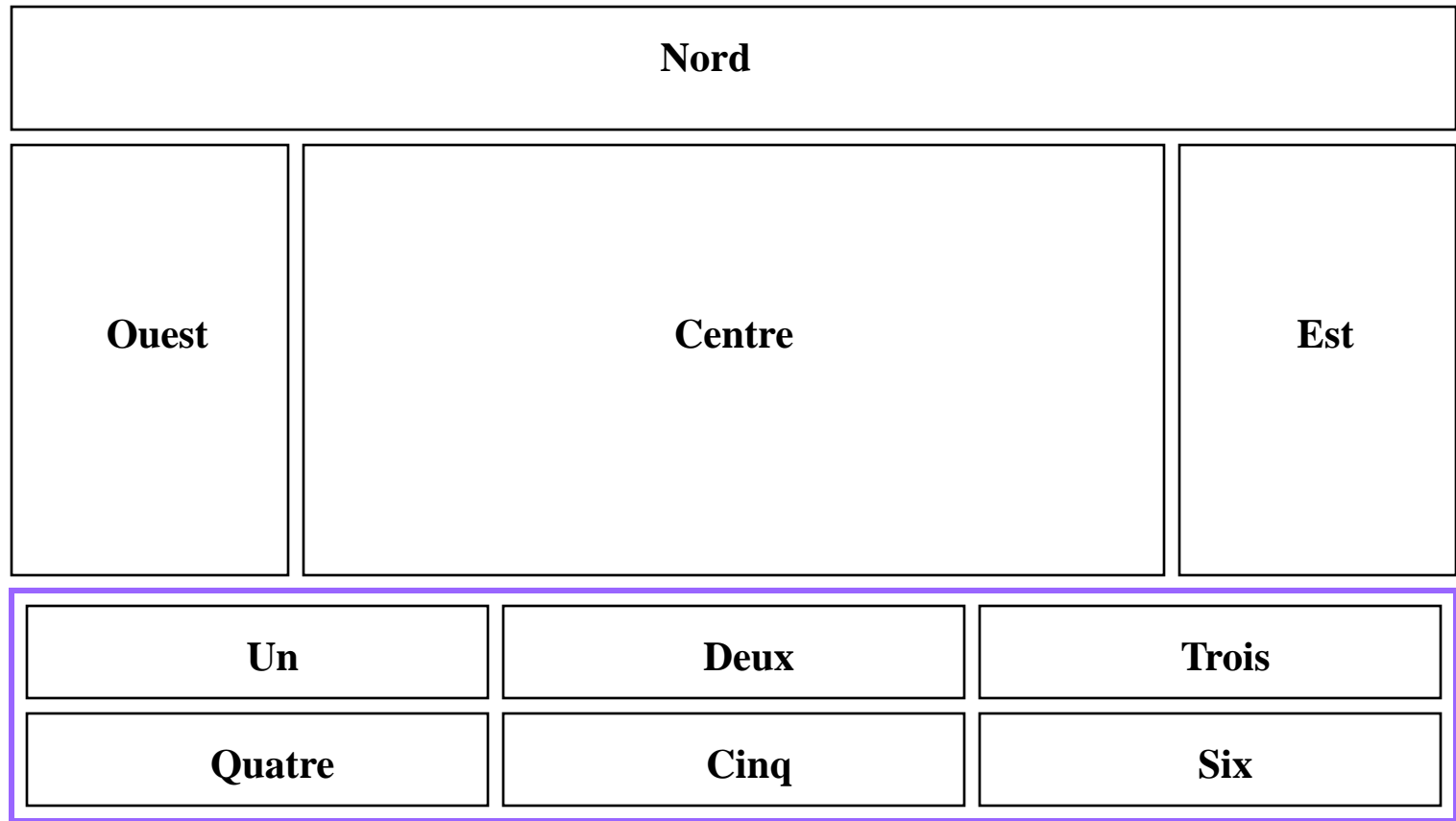
Gestion des évènements

RESUME



6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
- 6.6. Listes
- 6.7. Gestionnaires de tracé
 - 6.7.1. FlowLayout
 - 6.7.2. GridLayout
 - 6.7.3. BorderLayout



JPanel placé au sud

```

public class Fenetre extends JFrame{
    private Container c;
    private JButton nord, est, centre, ouest, un, deux, trois, quatre, cinq, six;
    private JPanel panneau;

    public Fenetre( )
    { ...
        c = getContentPane( );
        c.setLayout(new BorderLayout( ));

        nord = new JButton("Nord");
        c.add(nord,BorderLayout.NORTH);
        ouest = new JButton("Ouest");
        c.add(ouest,BorderLayout.WEST);
        centre = new JButton("Centre");
        c.add(centre,BorderLayout.CENTER);
        est = new JButton("Est");
        c.add(est,BorderLayout.EAST);

        ...

    }


```

...

```
un = new JButton(" un ");  
deux = new JButton(" deux ");  
trois = new JButton(" trois ");  
quatre = new JButton(" quatre ");  
cinq = new JButton(" cinq ");  
six = new JButton(" six ");
```

```
panneau = new JPanel( );  
panneau.setLayout( new GridLayout(2,3));
```

associe un
gestionnaire de
tracé au panneau



```
panneau.add(un);  
panneau.add(deux);  
panneau.add(trois);  
panneau.add(quatre);  
panneau.add(cinq);  
panneau.add(six);
```

ajout des boutons au panneau



```
c.add(panneau, BorderLayout.SOUTH); —→ ajout du panneau au container
```

...

6. Les composants Swing

- 6.7. Gestionnaires de tracé
 - 6.7.1. FlowLayout
 - 6.7.2. GridLayout
 - 6.7.3. BorderLayout

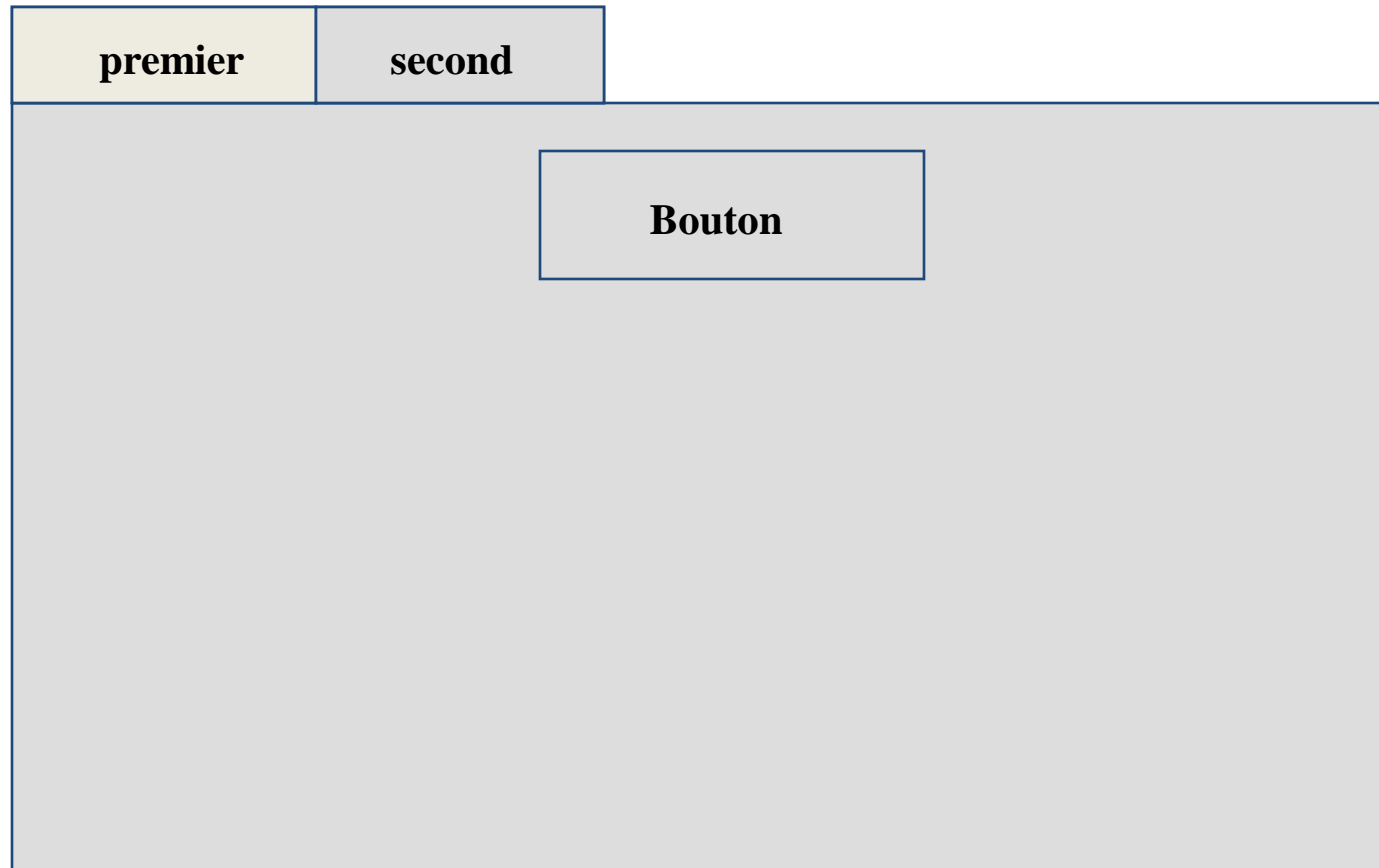
Autres gestionnaires de tracé:

GridBagLayout

CardLayout

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
- 6.6. Listes
- 6.7. Gestionnaires de tracé
- 6.8. Onglets (JTabbedPane)



```
public class Fenetre extends JFrame
{
    private Container cont;
    private JLabel label;
    private JButton bouton;
    private JPanel panneau1, panneau2;
    private JTabbedPane myTabbedPane;
```

```
    public Fenetre( )
    {
        ...
        cont = getContentPane( );
```

```
        panneau1 = new JPanel( );
        label = new JLabel("Bienvenue");
        panneau1.add(label);
```

—————→ **panneau du premier onglet**

```
        panneau2 = new JPanel( );
        bouton = new JButton("bouton");
        panneau2.add(bouton);
```

—————→ **panneau du second onglet**

```
        myTabbedPane = new JTabbedPane( );
        myTabbedPane.insertTab("premier ", null, panneau1, "ceci est le premier onglet", 0);
        myTabbedPane.insertTab("second ", null, panneau2, "ceci est le second onglet", 1);
```

```
        cont.add(myTabbedPane, BorderLayout.CENTER);
```

—————→ **JTabbedPane placé au centre**

```
        ...
```

titre de l'onglet

composant

index

```
myTabbedPane.insertTab( "premier ", null, panneau1, "ceci est le premier onglet", 0);
```

image
(Icon)

bulle d'aide

The diagram illustrates the parameters of the `insertTab` method. It shows a code snippet with five arguments: a string, `null`, a component, another string, and an integer. Colored arrows link each argument to a label: a purple arrow from the first string to 'titre de l'onglet', a green arrow from `null` to 'image (Icon)', a blue arrow from `panneau1` to 'composant', a black arrow from the second string to 'bulle d'aide', and a red arrow from the integer `0` to 'index'.

```
System.out.println("nombre d'onglets : " + myTabbedPane.getComponentCount( ));
```



nombre d'onglets

attention au casting

```
Component comp = (JPanel) (myTabbedPane.getComponentAt(1));
```



retourne le second composant

```
comp.setBackground(java.awt.Color.RED);
```



couleur rouge

6. Les composants Swing

- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
- 6.6. Listes
- 6.7. Gestionnaires de tracé
- 6.8. Onglets (JTabbedPane)
- 6.9. JSplitPane

bienvenue



bouton

```
public class Fenetre extends JFrame
{
    private Container cont;
    private JLabel label;
    private JButton bouton;
    private JPanel panneau1, panneau2;
    private JSplitPane mySplitPane;
```

```
    public Fenetre( )
    {
        ...
        panneau1 = new JPanel( );
        label = new JLabel("Bienvenue");
        panneau1.add(label);
```

```
        panneau2 = new JPanel( );
        bouton = new JButton("bouton");
        panneau2.add(bouton);
```

extension verticale:



```
        mySplitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT,true,panneau1, panneau2);
```

```
        mySplitPane.setOneTouchExpandable(true);
```

→ affiche: ▲ ▼

```
        mySplitPane.setDividerLocation(300);
```

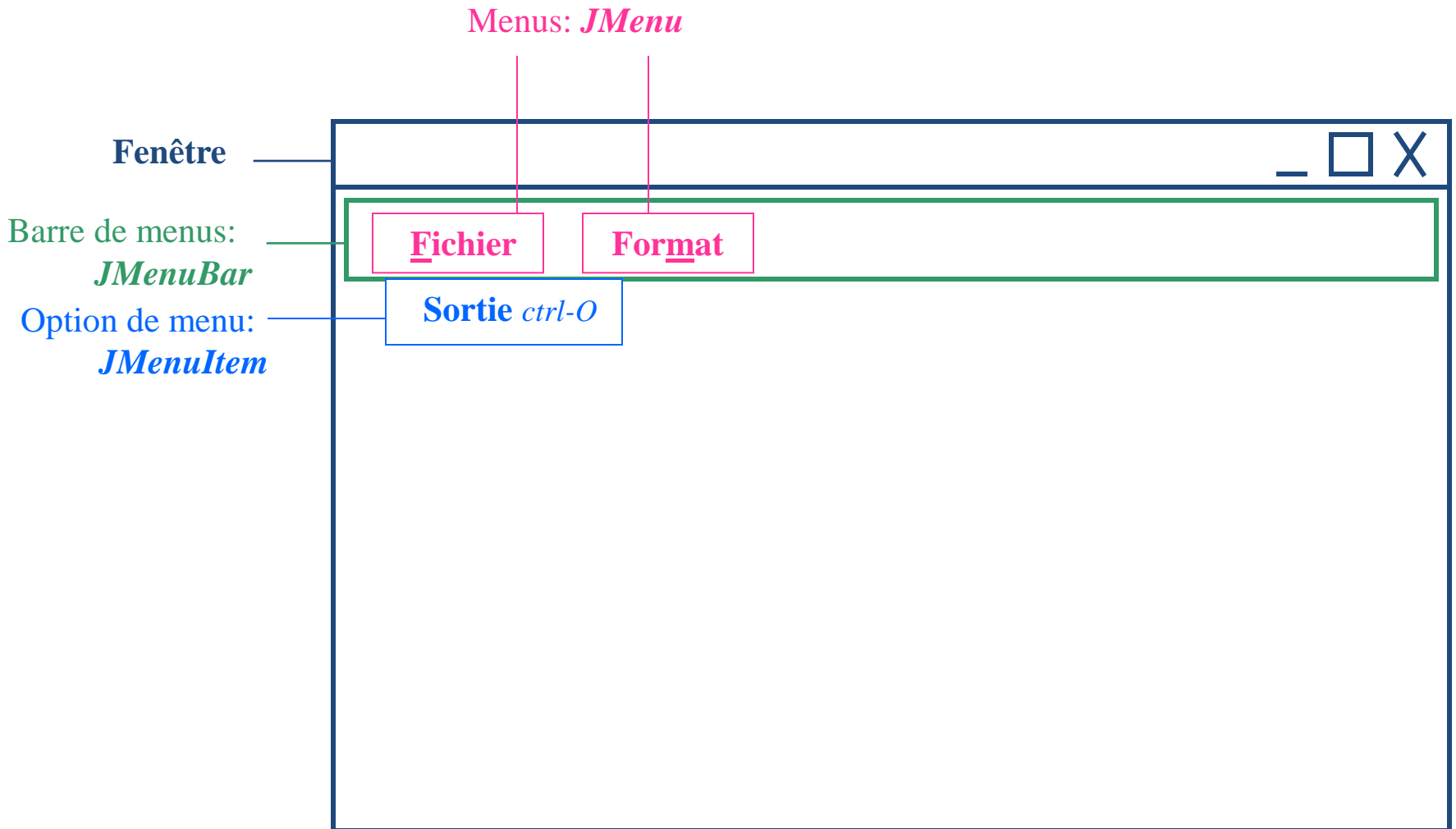
→ position de la séparation

```
        cont.add(mySplitPane, BorderLayout.CENTER);
```


```
        ...
```





6. Les composants Swing




- 6.1. JOptionPane
- 6.2. Fenêtres (JFrame)
- 6.3. Panneaux (JPanel)
- 6.4. Zones de texte
- 6.5. Boutons
- 6.6. Listes
- 6.7. Gestionnaires de tracé
- 6.8. Onglets (JTabbedPane)
- 6.9. JSplitPane
- 6.10. Menus



```

public class Fenetre extends JFrame
{
    ...
    private JMenuBar barre;
    private JMenu menuFichier, menuFormat ...;
    private JMenuItem sortie;
    ...
    public Fenetre( )
    {
        ...
        barre = new JMenuBar( );
        setJMenuBar(barre);  ajoute la barre de menus à la fenêtre

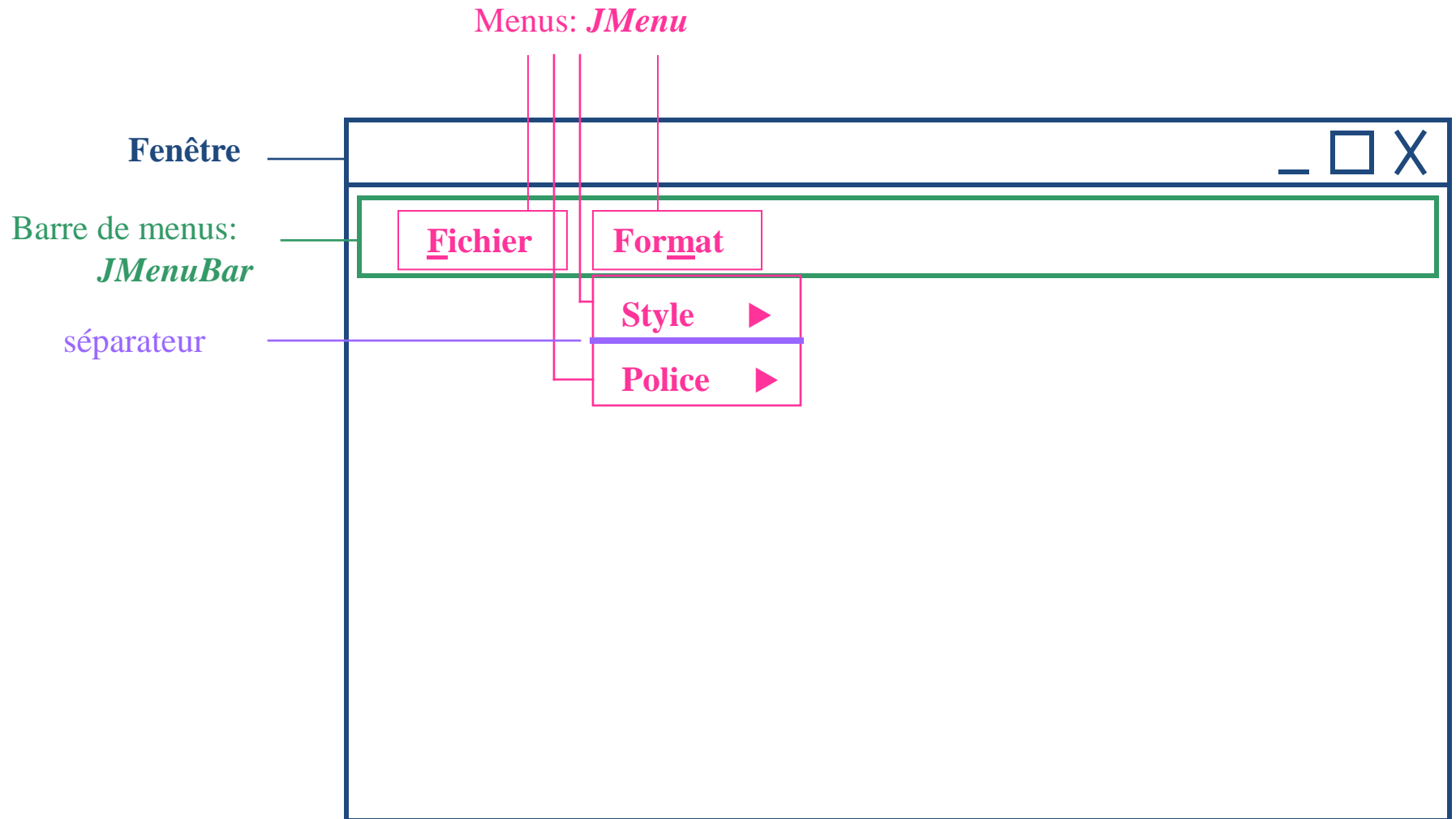
        menuFichier = new JMenu("Fichier");  crée un menu
        menuFichier.setMnemonic('F');  raccourci mnémonique: alt + F
        barre.add(menuFichier);  ajoute le menu Fichier à la barre de menus

        sortie = new JMenuItem("Sortie");  crée une option de menu
        sortie.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,InputEvent.CTRL_MASK));
        menuFichier.add(sortie);  Accélérateur clavier: ctrl + O
         ajoute l'option de menu Sortie au menu Fichier

        MonGestionnaireAction ga = new MonGestionnaireAction( );
        sortie.addActionListener(ga);

        menuFormat = new JMenu("Format");
        menuFormat.setMnemonic('m');
        barre.add(menuFormat);
    }
}

```



private JMenu ..., menuPolice, menuStyle;

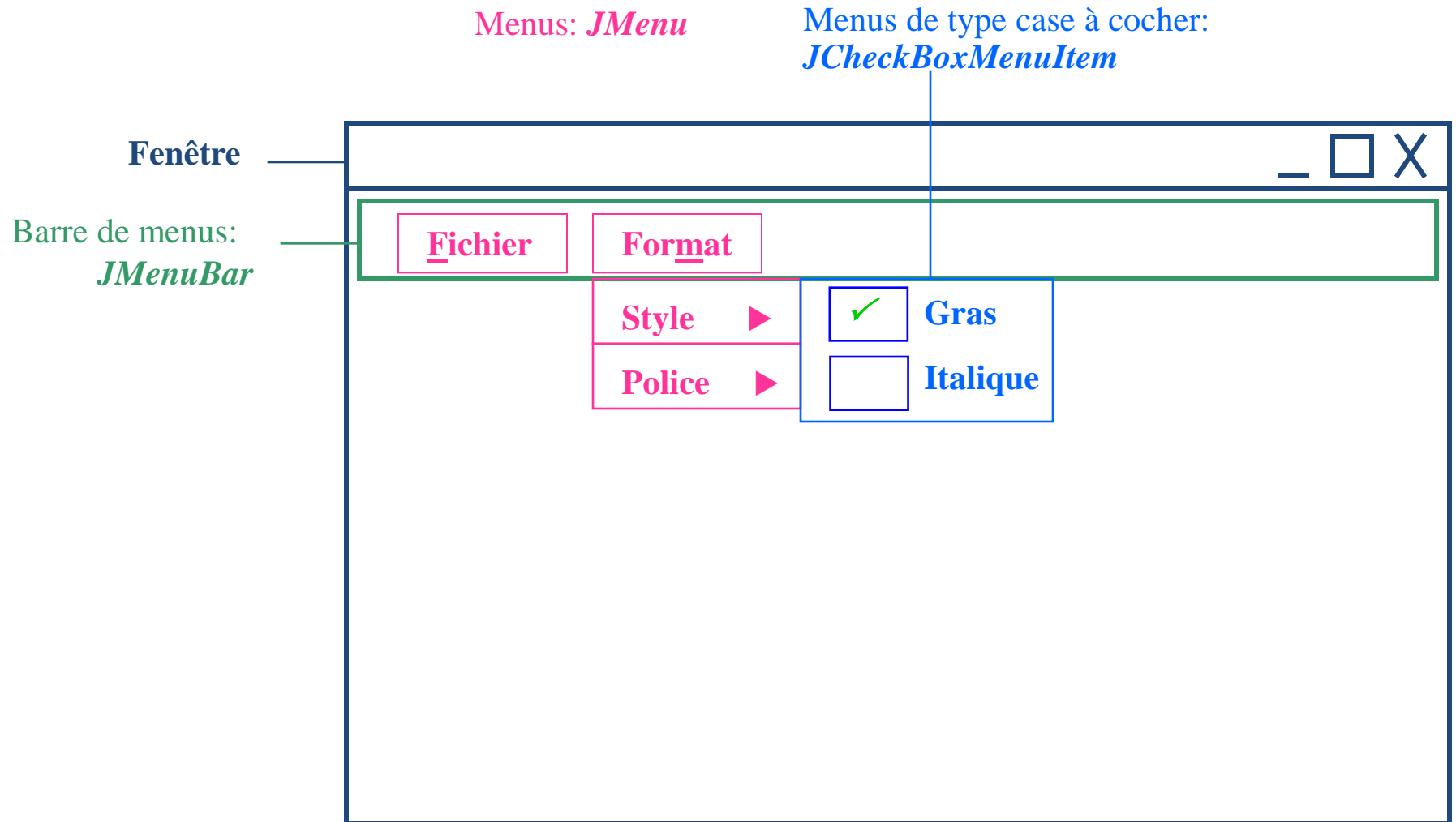
menuStyle = new JMenu("Style");

menuFormat.add(menuStyle);  ajoute le menu *Style* au menu *Format*

menuFormat.addSeparator();  ajoute un séparateur

menuPolice = new JMenu("Police");

menuFormat.add(menuPolice);



```
private JCheckBoxMenuItem gras, italic ;
```

```
MonGestionnaireCheckBox gcb = new MonGestionnaireCheckBox();
```

```
gras = new JCheckBoxMenuItem("Gras");
```

```
gras.addItemListener(gcb);
```

—————→ associe un écouteur d'événement à l'option de menu *gras*

```
menuStyle.add(gras);
```

—————→ ajoute l'option de menu *gras* au menu *Style*

```
italic = new JCheckBoxMenuItem("Italique");
```

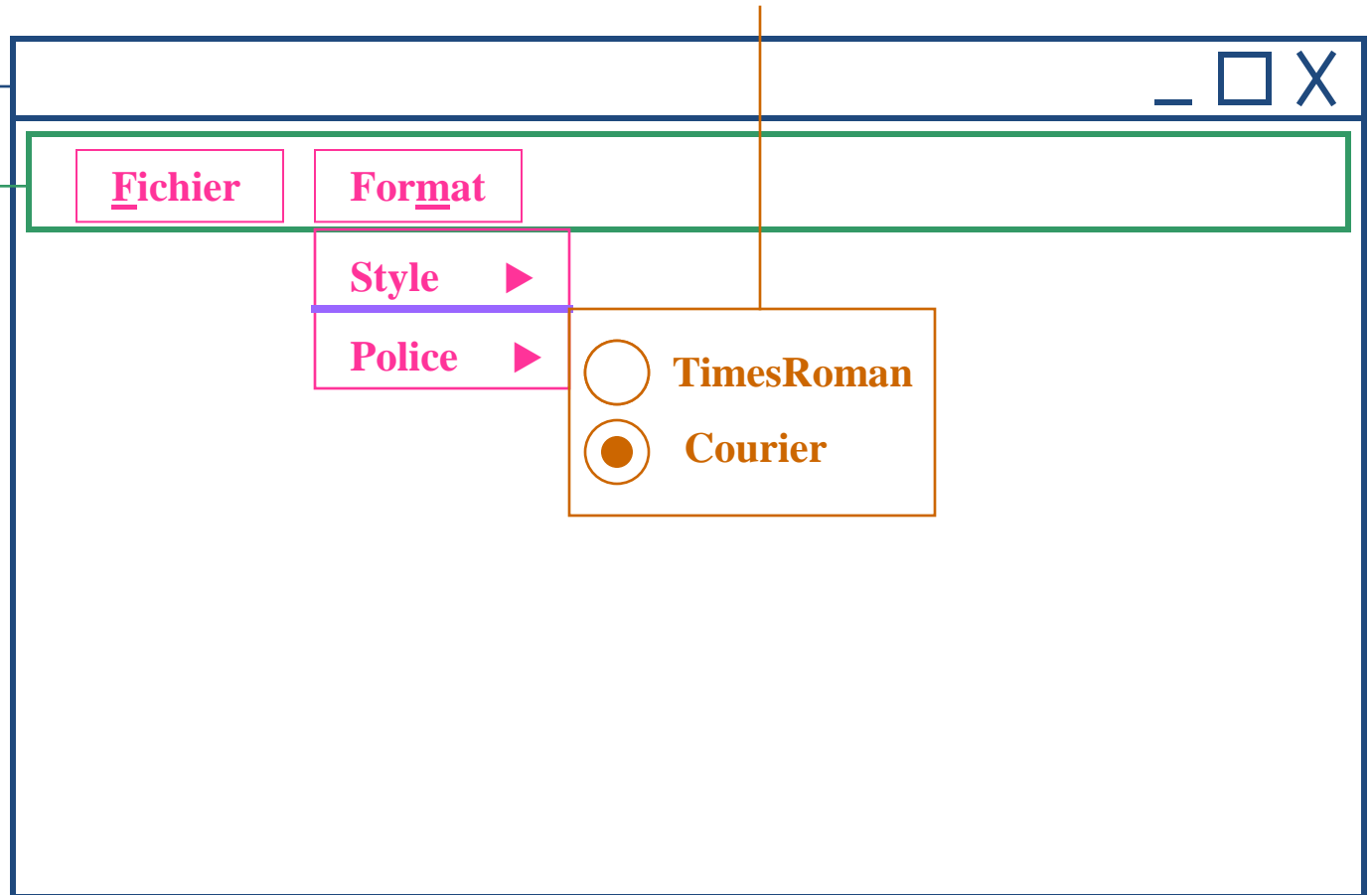
```
italic.addItemListener(gcb);
```

```
menuStyle.add(italic);
```

Menus: *JMenu*

Menus de type bouton radio:
JRadioButtonMenuItem

Fenêtre
Barre de menus:
JMenuBar



private JRadioButtonMenuItem times, courier;

private ButtonGroup groupeRadio;

MonGestionnaireRadio gr = new MonGestionnaireRadio();

times = new **JRadioButtonMenuItem**("TimesRoman");

times.**setSelected**(true);  **option par défaut**

menuPolice.add(times);  **ajoute l'option de menu *TimesRoman* au menu *Police***

times.**addItemListener**(gr);  **associe un écouteur d'événement à l'option de menu *TimesRoman***

courier = new JRadioButtonMenuItem("Courier");

menuPolice.add(courier);

courier.**addItemListener**(gr);

groupeRadio = new **ButtonGroup**(); 

groupeRadio.add(times);

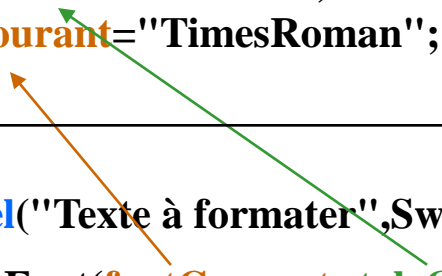
groupeRadio.add(courier);

Rappel: ButtonGroup:

un seul bouton radio coché à la fois

```
private JLabel texte;  
private int styleCourant=Font.PLAIN;  
private String fontCourant="TimesRoman";
```

```
texte = new JLabel("Texte à formater",SwingConstants.CENTER);  
texte.setFont(new Font(fontCourant,styleCourant,16));  
c.add(texte,BorderLayout.CENTER);
```



private class MonGestionnaireAction implements **ActionListener**

{ public void **actionPerformed**(ActionEvent e)

{ System.exit(0); }

}

private class MonGestionnaireRadio implements **ItemListener**

{ public void **itemStateChanged**(**ItemEvent** e)

{ if (**e.getSource**() == **times**)

fontCourant = "**TimesRoman**";

 else

fontCourant = "**Courier**";

texte.setFont(new Font(**fontCourant**,**styleCourant**,16));

 }

}

private class MonGestionnaireCheckBox implements ItemListener

{ public void **itemStateChanged**(**ItemEvent** e)

{ **styleCourant** = Font.**PLAIN**;

PLAIN = 0

if (**gras**.isSelected())

BOLD = 1

styleCourant += Font.**BOLD**;

ITALIC = 2

if (**italic**.isSelected())

styleCourant += Font.**ITALIC**;

texte.setFont(new Font(**fontCourant**,**styleCourant**,16));

}