ProgIHM - Cours 4

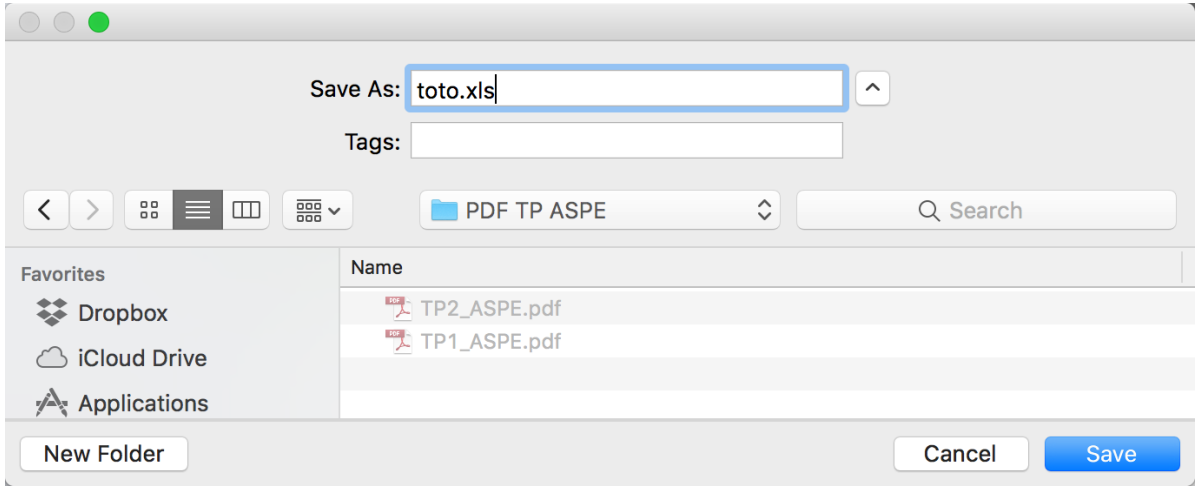
Les Fenêtres modales et de Sélection de Fichiers, les Menus, la barre d'outils

V. DESLANDRES

veronique.deslandres@univ-lyon1.fr

Sommaire de ce cours

Les fenêtres de sélection de fichiers

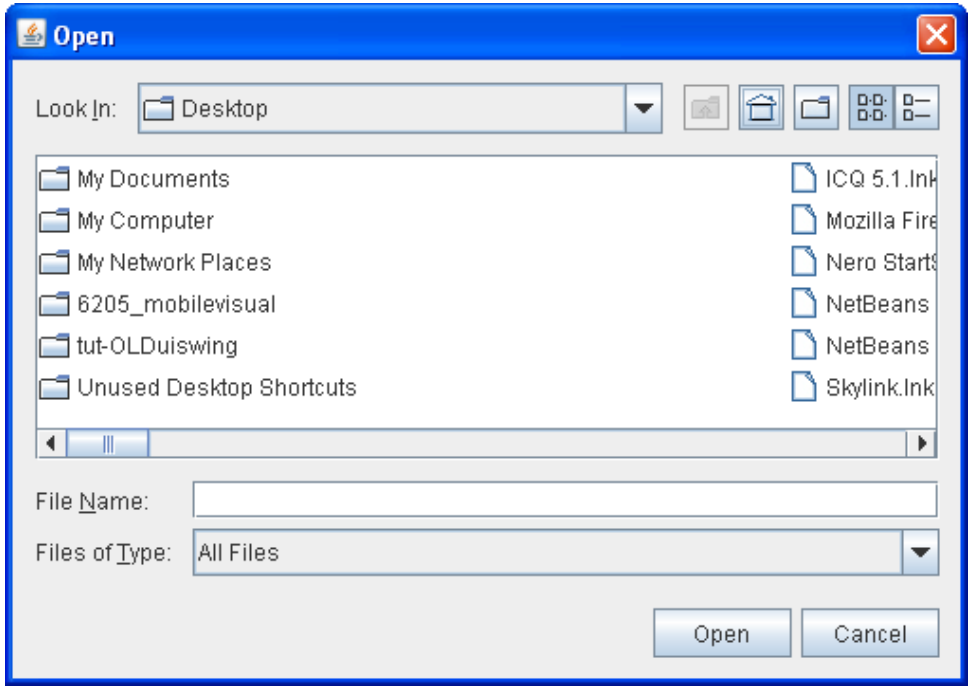


FileDialog

2 sortes de fenêtre pour sélectionner les Fichiers

- La **FileDialog** d'AWT : fenêtre de base permettant d'ouvrir ou d'enregistrer un fichier
  - Hérite de `java.awt.window`
  - Simple d'utilisation

- Le **JFileChooser** de SWING : fenêtre plus élaborée avec notamment la possibilité de filtrer les fichiers



JFileChooser

FileDialog d'AWT

Choix du fichier pour ouvrir

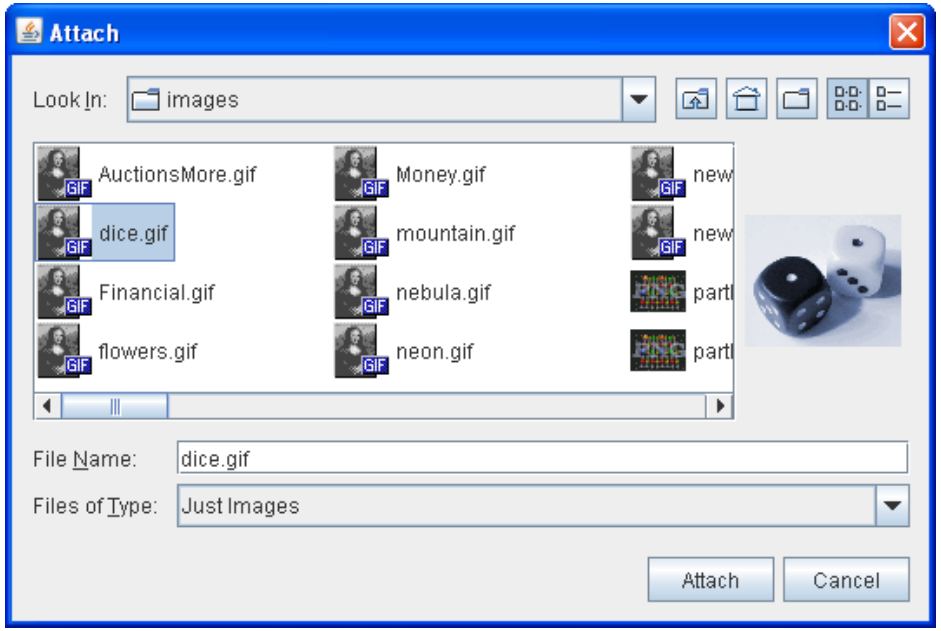
```
String nomFic = new String("");
try {
        // ouvrir un fichier
        FileDialog fd = new FileDialog(this , "Sélectionnez votre fichier...", FileDialog.LOAD);
        fd.setVisible(true);
        nomFic = ((fd.getDirectory()).concat(fd.getFile()));
}
catch (NullPointerException e) {
        System.out.println("Erreur ouverture dossier !");
}
```

Choix du fichier pour enregistrer

```
String nomFic = new String("");
try {
        FileDialog fd = new FileDialog(this , "Sélectionnez votre  fichier...", FileDialog.SAVE);
        fd.setVisible(true);
        nomFic = ((fd.getDirectory()).concat(fd.getFile()));
}
catch(NullPointerException e) {
    System.out.println("Erreur ouverture dossier !");
}
```

JFileChooser

- Beaucoup plus complet
  - Permet de définir des **filtres** : types de fichiers, fichier ou répertoires,…
  - Ici filtre sur fichiers images



- Regarder le tutoriel sur le site Oracle :

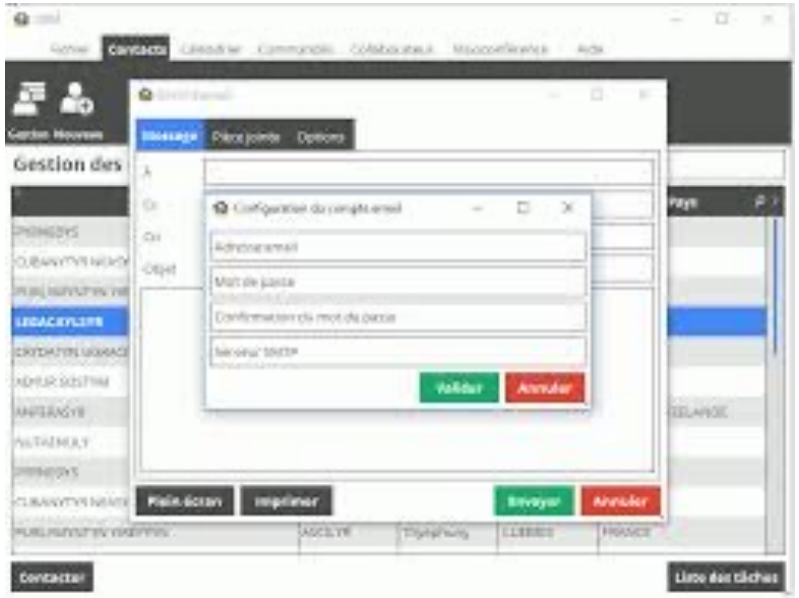
https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html

DEF Fenêtres modales

- Ces fenêtres de sélection de fichiers sont des « fenêtres modales »
- Càd des **fenêtres secondaires** lancées par une fenêtre **parent**

- La fenêtre **parent** restant alors inaccessible tant que l'utilisateur n'a pas terminé son action ou donné sa réponse

Construire ses propres Fenêtres Modales



Fenêtre Modale

- C'est une fenêtre qui **dépend** d'une autre fenêtre et qui **prend le contrôle** du clavier et de l'écran.
  - Par ex.: quand on pose une question à laquelle il est impératif que l'utilisateur réponde, avant de pouvoir à nouveau interagir avec le reste du programme

- La fenêtre modale permet :
  - **d'obtenir des données** de l'utilisateur,
  - de **fournir une information** à l'utilisateur

- Pour saisir des données de l'application, on utilise souvent la classe `JDialog`
  - Semblable à celle de `JFrame`
  - Utilisées pour des fenêtres modales et *non* modales

Fenêtre Modale (Exemple)

- Exemple:
  - On utilise la classe `Point` (d'AWT) avec 2 coordonnées réelles `X` et `Y`
  - On réalise un petit programme d'affichage ou de saisie/modification des coordonnées d'un point
  - Ici l'utilisateur doit fermer la boîte de dialogue avant de pouvoir à nouveau interagir avec le reste du programme

*Quand la fenêtre modale de saisie des nouvelles coordonnées du point s'ouvre, la fenêtre Parent devient **inactive***

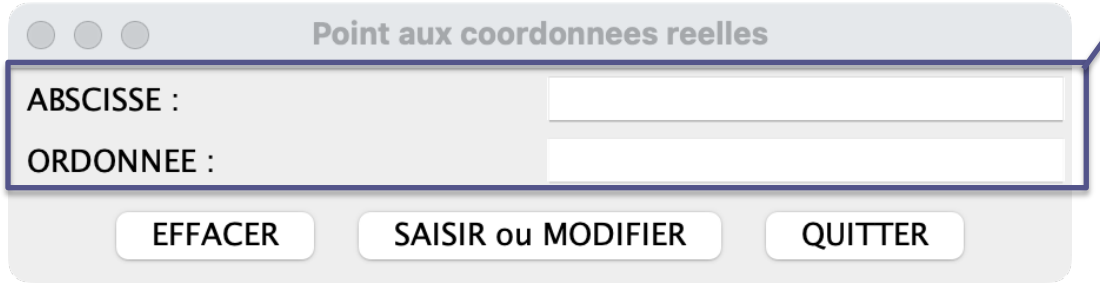
Fenêtre Parent (1/2)

```
public class AffichePoint extends JFrame implements ActionListener {

    JLabel lbl_abs, lbl_ord;
    JTextField tf_xa, tf_ya;
    JButton bt_Effacer, bt_Quitter, bt_Modifier;
    Point2D.Double pointA;          // classe Point d'AWT : coo
```

Classe Point aux coordonnées réelles

```
public AffichePoint() {

    // avec pack(), par défaut, la fen. est collée au coin HG de l'écran
    setBounds(500, 400, 600, 550);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("Fenêtre d'affichage d'un point");

    Container cc = getContentPane();
    cc.setLayout(new BorderLayout());
    cc.add(panneau_bas(), BorderLayout.SOUTH);
    cc.add(panneau_milieu(), BorderLayout.CENTER);

    pointA = new Point2D.Double(0.0, 0.0);
    setVisible(true);
}
```

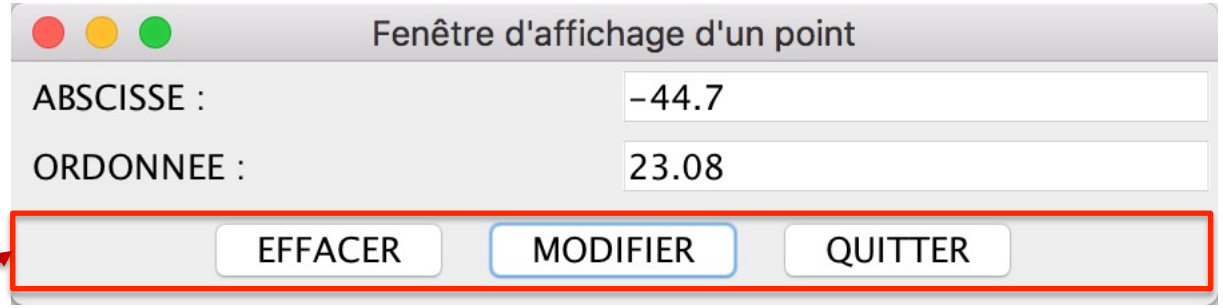
Ici on choisit de passer par des fonctions qui retournent le JPanel désiré




```
JPanel panneau_milieu() {
    JPanel milieu = new JPanel();
    milieu.setLayout(new GridLayout(2, 2));

    lbl_abs = new JLabel("  ABSCISSE : ");
    lbl_ord = new JLabel("  ORDONNEE : ");

    tf_xa = new JTextField(18);
    tf_ya = new JTextField(18);

    tf_xa.setEditable(false); // pas d'édition possible ici
    tf_ya.setEditable(false);

    milieu.add(lbl_abs);
    milieu.add(tf_xa);
    milieu.add(lbl_ord);
    milieu.add(tf_ya);

    return milieu;
}
```

Fenêtre Parent (2/2)

```
JPanel panneau_bas() {
    JPanel bas = new JPanel();

    bt_Effacer = new JButton("EFFACER");
    bt_Modifier = new JButton("MODIFIER");
    bt_Quitter = new JButton("QUITTER");

    bt_Quitter.addActionListener(this);
    bt_Modifier.addActionListener(this);
    bt_Effacer.addActionListener(this);

    bas.add(bt_Effacer);
    bas.add(bt_Modifier);
    bas.add(bt_Quitter);

    return bas;
}
```



*Si **MODIFIER** : on ouvre la fenêtre modale à qui on envoie le point en paramètre, et si l'utilisateur a validé, on affiche les nouvelles coordonnées modifiées*

```java
// Implémentation de l'écouteur des boutons MODIFIER, EFFACER, QUITTER
@Override
public void actionPerformed(ActionEvent e) {

    if (e.getSource() == bt_Modifier) {
        // on cree une fenetre modale liée a la fenetre courante:
        ModaleSaisiePoint fenSaisirUnPoint;
        fenSaisirUnPoint = new ModaleSaisiePoint(this, pointA);

        // si les coordonnees du pointA ont ete modifiees :
        if (fenSaisirUnPoint.OKchoisi) {
            tf_xa.setText(String.valueOf(pointA.getX()));
            tf_ya.setText(String.valueOf(pointA.getY()));
        }
    } else if (e.getSource() == bt_Effacer) {

        pointA.setLocation(0.0, 0.0);
        tf_xa.setText(String.valueOf(pointA.getX()));
        // ou tf_xa.setText(""+pointA.getX());
        // tf_xa.setText(pointA.getX());       // erreur compilation
        tf_ya.setText(String.valueOf(pointA.getY()));

    } else if (e.getSource() == bt_Quitter) {
        System.exit(0);
    }
```

Le point saisi :
X, Y

*Si **EFFACER** : on redéfinit le point aux coordonnées 0,0 et on l'affiche*

*Si **QUITTER** : on ferme la fenêtre*

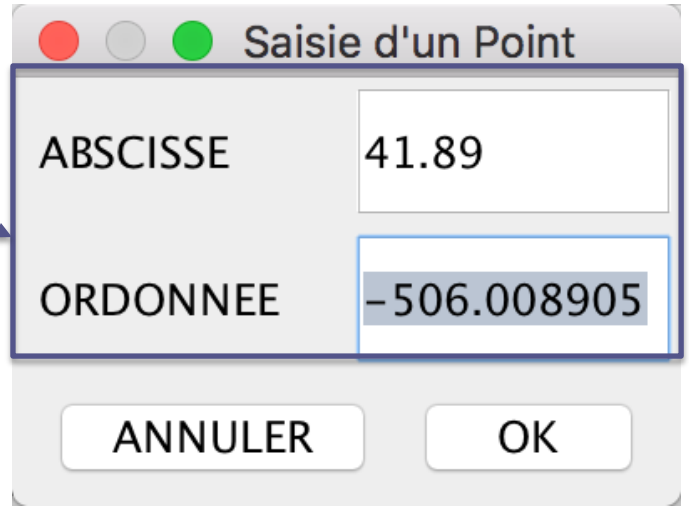
Fenêtre de saisie (1/4)

```
public class ModaleSaisiePoint extends JDialog implements ActionListener {

    Point2D.Double lePoint; // le point véhiculé entre les 2 fenêtres

    JLabel lbl_abs, lbl_ord;
    JTextField tf_x, tf_y;
    JButton btOK, btANNUL;
    boolean OKchoisi; // flag pour savoir si une saisie a été validee

    ModaleSaisiePoint(JFrame f, Point2D.Double p) {

        super(f, "Saisie d'un point", true); // true : construction de la fenêtre modale
        // sans 'true' : la fenêtre parent reste active
        this.lePoint = p;                    // on récupère le point envoyé lors de l'appel

        this.setBounds(570, 420, 200, 150);
        // on laisse gerer la fermeture par la fenetre mere :
        this.setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new BorderLayout());
        c.add(panneau_milieu(), BorderLayout.CENTER);
        c.add(panneau_bas(), BorderLayout.SOUTH);

        OKchoisi = false; // est à FALSE tant qu'on n'a pas validé une saisie
        this.setVisible(true);
    }
```

// true pour modale

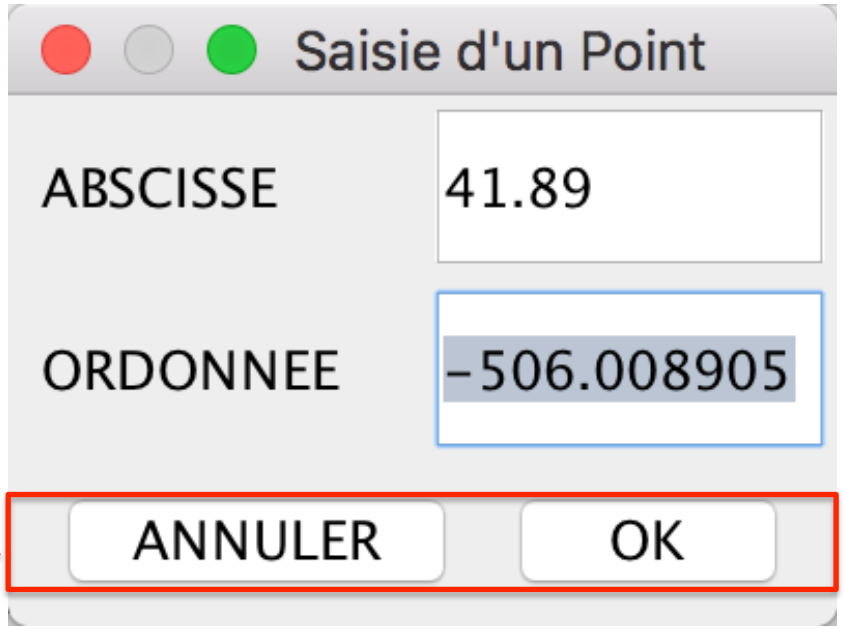
Fenêtre de saisie (2/4)

```
JPanel panneau_milieu() {
    JPanel milieu = new JPanel();
    milieu.setLayout(new GridLayout(2, 2));

    lbl_abs = new JLabel("  ABSCISSE");
    lbl_ord = new JLabel("  ORDONNEE");

    tf_x = new JTextField(20);
    tf_y = new JTextField(20);

    milieu.add(lbl_abs);
    milieu.add(tf_x);
    milieu.add(lbl_ord);
    milieu.add(tf_y);

    tf_x.setText(String.valueOf(lePoint.getX()));
    tf_y.setText(String.valueOf(lePoint.getY()));

    return milieu;
}
```



Fenêtre Saisie (3/4)

```
JPanel panneau_bas() {
    JPanel pan = new JPanel();

    btOK = new JButton("OK");
    btANNUL = new JButton("ANNULER");

    pan.add(btANNUL);
    pan.add(btOK);

    btOK.addActionListener(this);
    btANNUL.addActionListener(this);

    return pan;
}
```



Fenêtre Saisie (4/4)

```
// Implémentation de l'écouteur sur les boutons OK et ANNULER
@Override
public void actionPerformed(ActionEvent e) {
    double vx, vy;
    if (e.getSource() == btOK) {
        // lecture de l'abscisse :
        try {
            vx = Double.parseDouble(tf_x.getText());
            // ou vx = new Double(tf_x.getText());

        } catch (RuntimeException ex) {
            // en cas d'erreur sur la saisie, reaffecte les coordonnees d'origine
            vx = lePoint.getX();
            tf_x.setText(String.valueOf(lePoint.getX()));
        }

        // idem pour l'ordonnee :
        try {
            vy = Double.parseDouble(tf_y.getText());

        } catch (RuntimeException ex) {
            vy = lePoint.getY();
            tf_y.setText(String.valueOf(lePoint.getY()));
        }

        // affecte nouvelles valeurs au Point lePoint :
        lePoint.setLocation(vx, vy);
        OKchoisi = true;
```

On récupère la valeur des champs X et Y

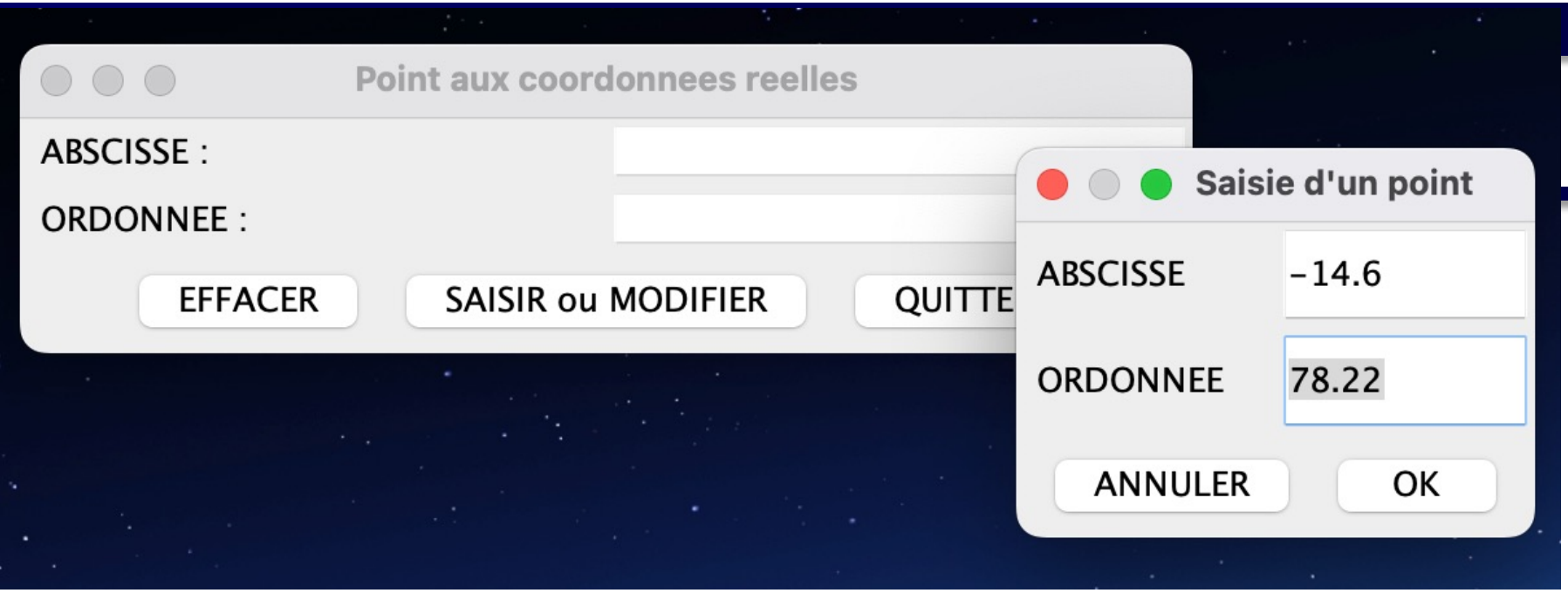
En cas de pb (si une exception est levée), on prend les anciennes valeurs du point

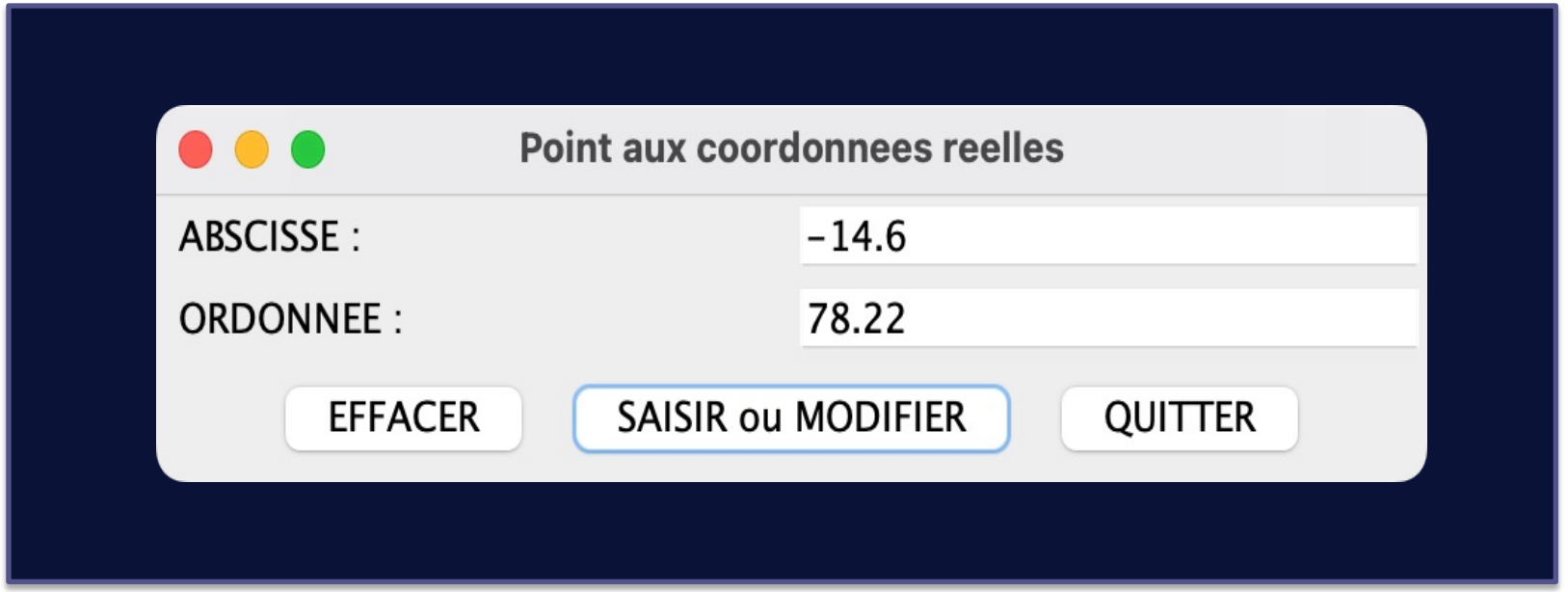
On les transmet au point

(fin de ActionPerformed() de la fenêtre modale)

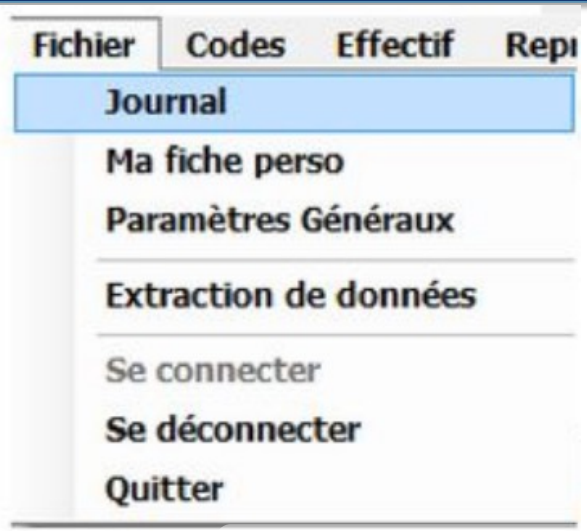
```
        } else {  // clic bt_ANNUL : on indique que le point n'a pas été modifié
            OKchoisi = false;
        }
        dispose(); // on rend la main à la fenetre parent
    } // fin écouteur

} // fin classe ModaleSaisiePoint
```

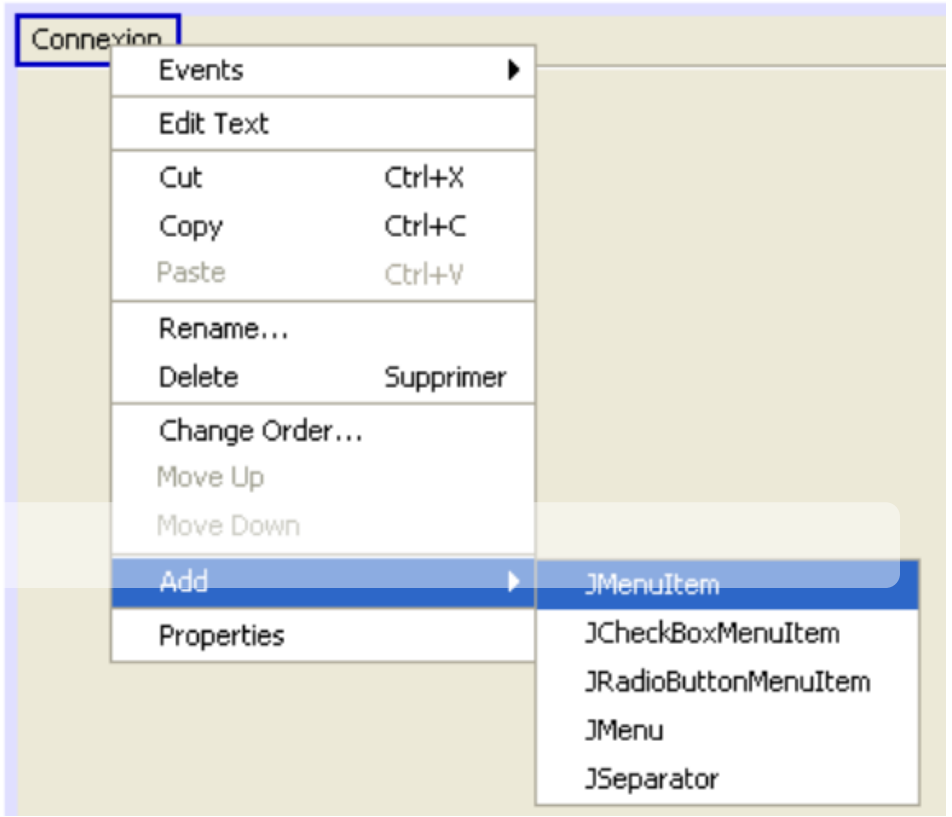
***dispose**() de JFrame ferme la fenêtre modale et toutes les autres fenêtres éventuellement ouvertes à partir d'elle, et rend la main à la fenêtre parent*


Point aux coordonnees reelles
ABSCISSE :
ORDONNEE :
EFFACER
SAISIR ou MODIFIER
QUITTE
Saisie d'un point
ABSCISSE
-14.6
ORDONNEE
78.22
ANNULER
OK


Point aux coordonnees reelles
ABSCISSE :
-14.6
ORDONNEE :
78.22
EFFACER
SAISIR ou MODIFIER
QUITTER

Les Menus

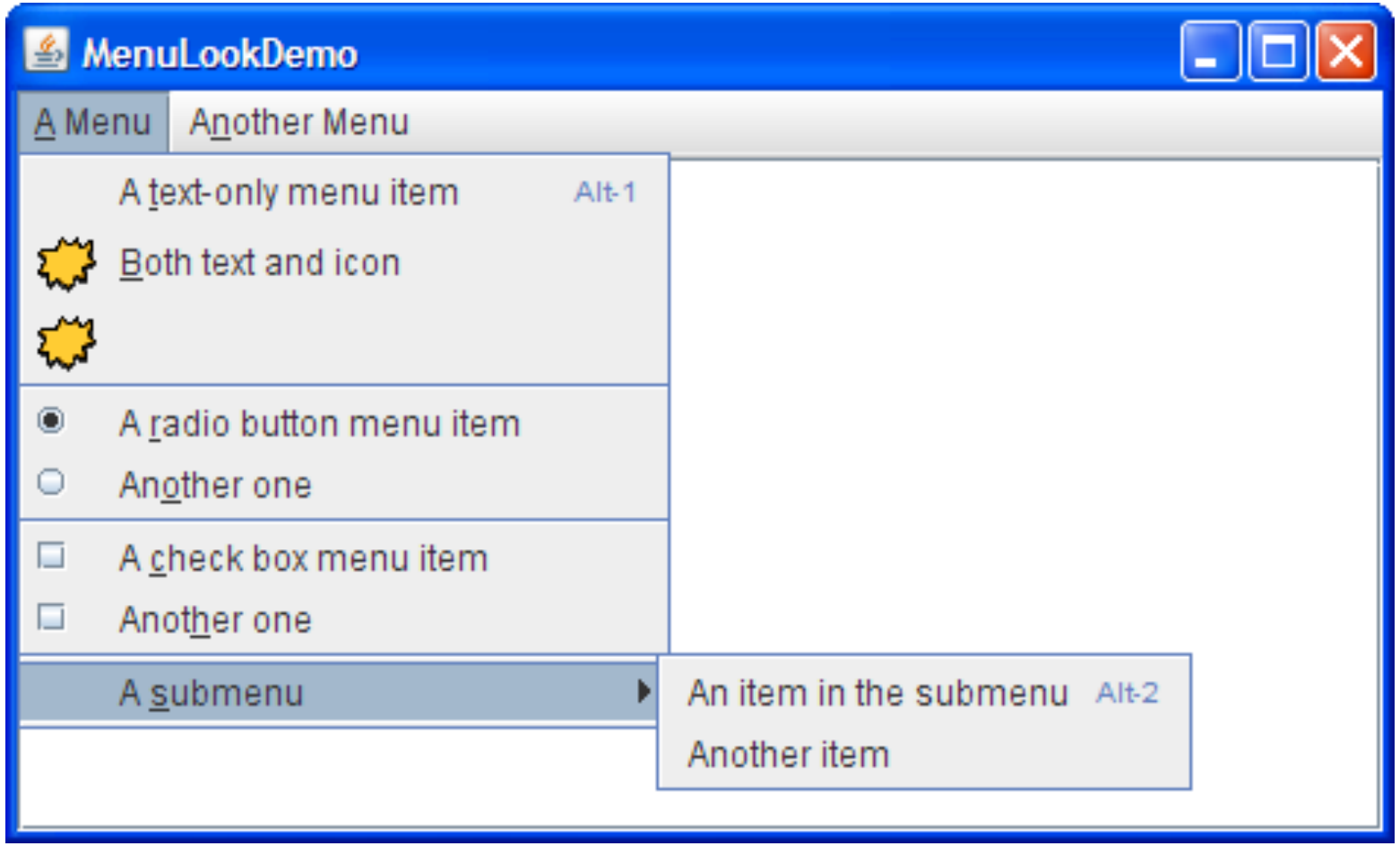

Fichier
Codes
Effectif
Repı
Journal
Ma fiche perso
Paramètres Généraux
Extraction de données
Se connecter
Se déconnecter
Quitter


Connexion
Events
Edit Text
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Rename...
Delete Supprimer
Change Order...
Move Up
Move Down
Add
Properties
JMenuItem
JCheckBoxMenuItem
JRadioButtonMenuItem
JMenu
JSeparator

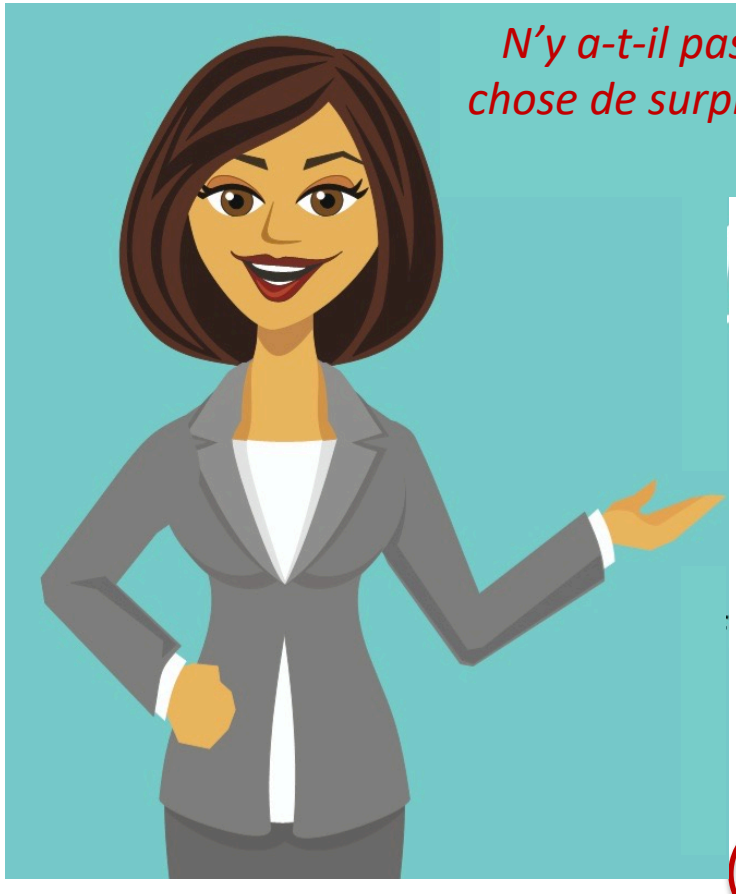
Les classes de menus

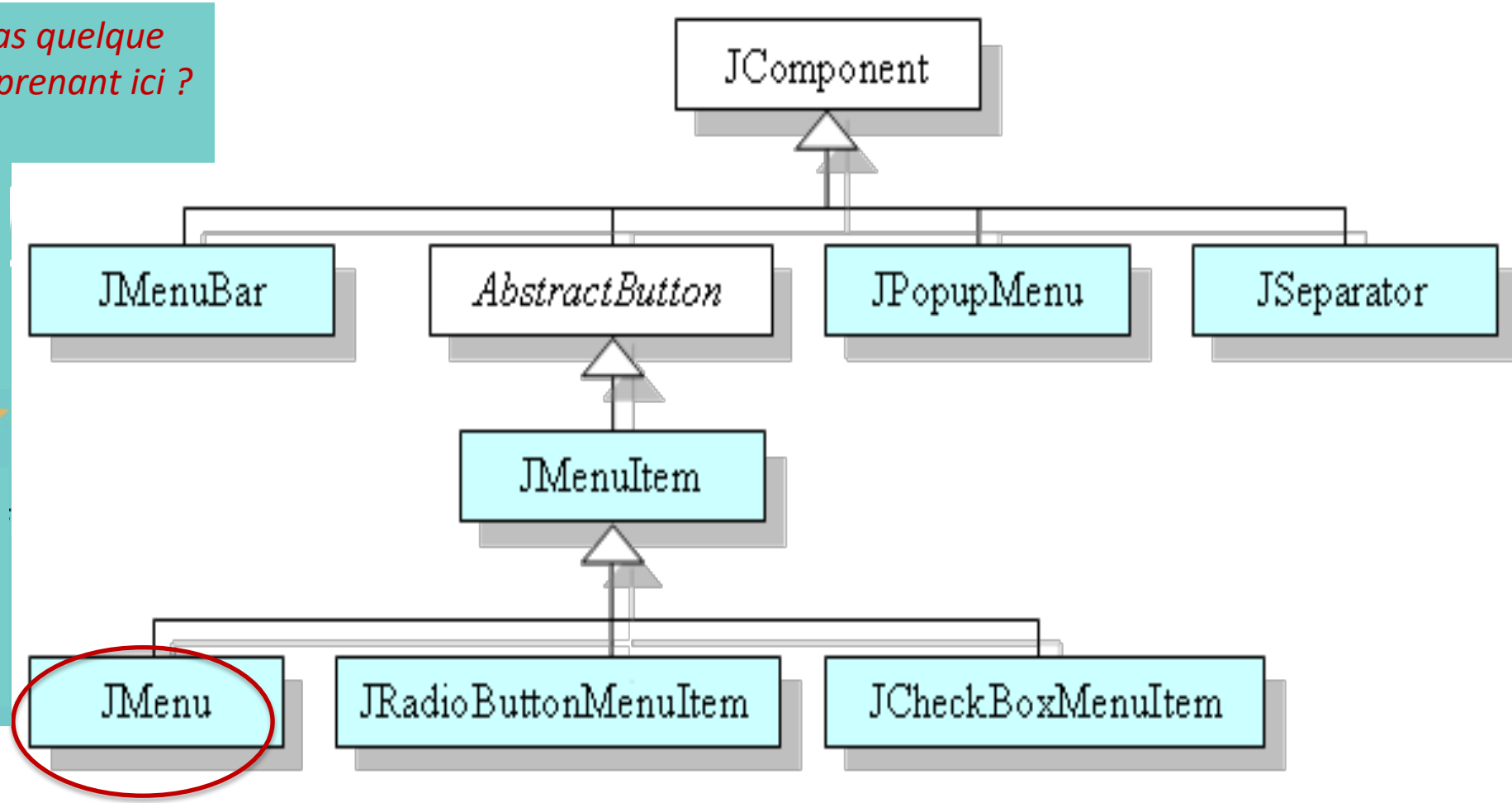
- Il existe plusieurs classes pour créer des menus en Java
  - `JMenu` qui sert à créer un **menu**
  - `JMenubar` sert à créer une **barre** de menus
  - `JPopupMenu` sert à créer un menu **déroulant**
  - `JMenuItem` est un **item** d'un menu
  - `JCheckBoxMenuItem` est un item de menu, à **cocher**
  - `JRadioButtonMenuItem` est un item **Radio**

Les menus


MenuLookDemo
A Menu
Another Menu
A text-only menu item
Alt-1
Both text and icon
A radio button menu item
Another one
A check box menu item
Another one
A submenu
An item in the submenu
Alt-2
Another item

Hiérarchie de composants des menus





Les menus

- Par convention, les menus ne sont pas placés dans d'autres composants de l'interface

  → **pas d'ajout au contentPane**

- Ils apparaissent :
  - soit dans une **barre de menus** (associée à la fenêtre avec `setMenuBar()`)
  - soit dans un menu **déroulant (JPopupMenu)**

- Le menu **déroulant lui-même** n'a pas besoin **d'écouteur**
  - C'est géré automatiquement par l'API Swing
  - On place juste des écouteurs **sur les items de menu**

Les menus (exemple)

Dans la fenêtre

```
protected JMenuItem quitter = new JMenuItem("Quitter");
protected JMenuItem option1 = new JMenuItem("Option-1");
protected JMenuItem option2 = new JMenuItem("Option-2");
protected JMenuItem option3 = new JMenuItem("Option-3");
```

On crée les 4 items de menu

Dans le constructeur

```
JMenu fichier = new JMenu("Fichier");
fichier.add(quitter);
JMenu edition = new JMenu("Edition");
edition.add(option1);
edition.add(option2);
edition.addSeparator();
edition.add(option3);
```

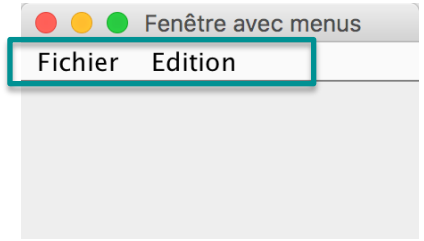
On crée les 2 menus (Fichier et Edition) et on ajoute les items aux menus avec une ligne séparatrice

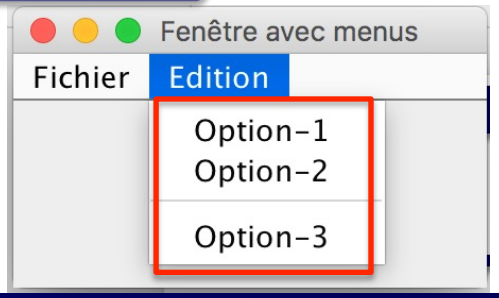
Dans le constructeur

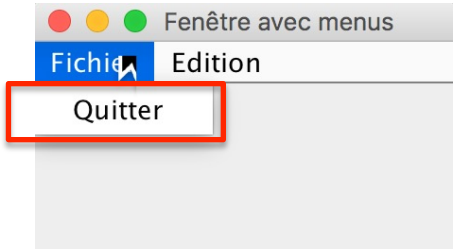
```
JMenuBar mb = new JMenuBar();
mb.add(fichier);
mb.add(edition);
this.setJMenuBar(mb);
```

On crée la barre de menu, on y ajoute les 2 menus

Méthode de JFrame : ajoute la barre de menu à la fenêtre







Les écouteurs des items de menu

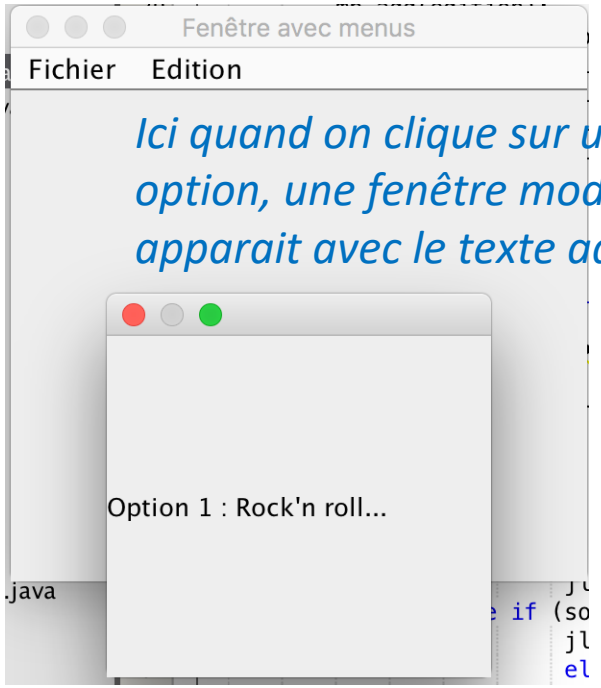
Dans le constructeur

```
quitter.addActionListener( new MonActionListener());
option3.addActionListener( new MonActionListener());
option2.addActionListener( new MonActionListener());
option1.addActionListener( new MonActionListener());
```

On ajoute l'écouteur à chaque item

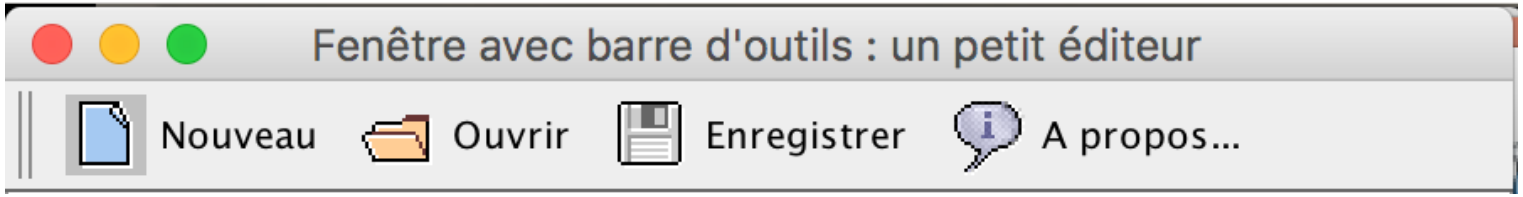
Exemple d'écouteur (ici en classe interne)

```
class MonActionListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {

            Object source = e.getSource();
            if (source == quitter) System.exit(0);

            JDialog jd = new JDialog();
            JLabel jl;
            if (source == option1)
                    jl = new JLabel("Option 1 : Rock'n roll...");
            else if (source == option2)
                    jl = new JLabel("Option 2 : Choubidou waa...");
                    else
                    jl = new JLabel("Option 3 : Lalala...");
            jd.setSize(200,200);
            jd.setLocation(250,300);
            jd.getContentPane().add(jl); // on ajoute le JLabel ici
            jd.setVisible(true);
    }
} // fin classe ecouteur
```

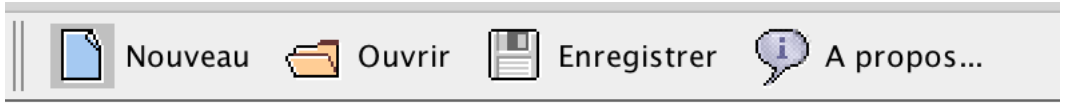


Ici quand on clique sur une option, une fenêtre modale apparait avec le texte ad-hoc :

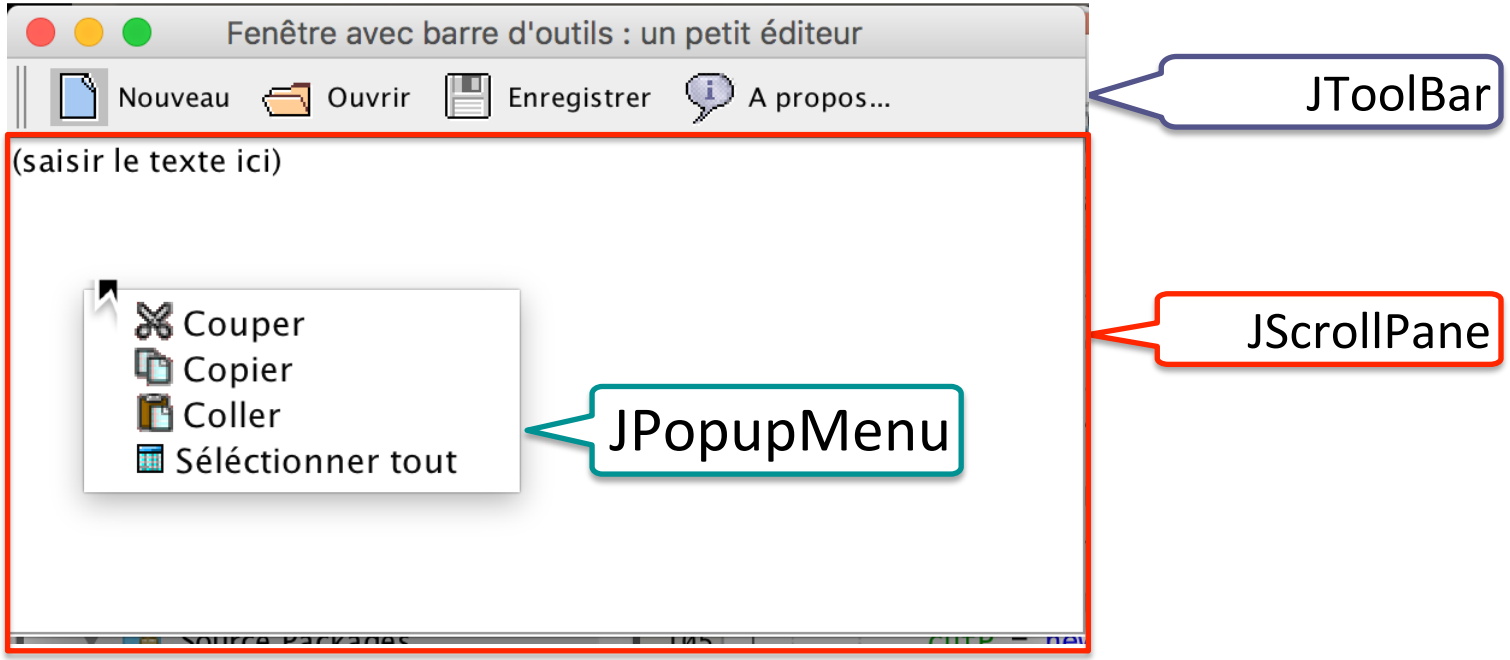
Les barres d'outils et le menu déroulant



Barre d'outils

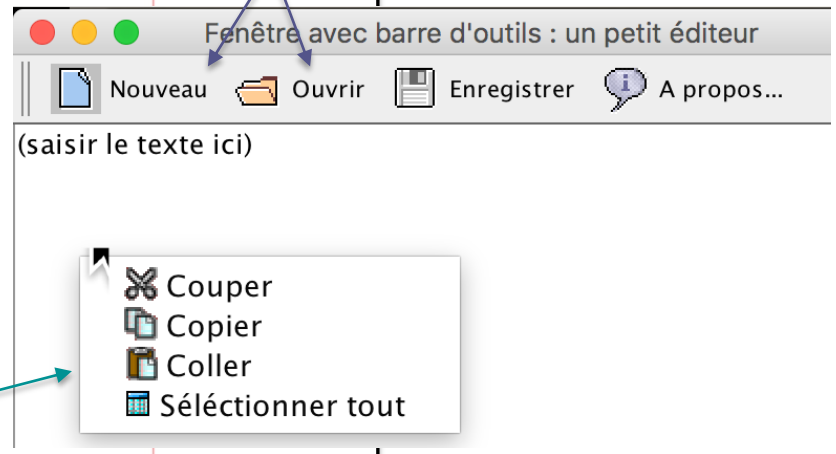


- Illustration sur un **petit éditeur de texte**
- La fenêtre est munie d'**ascenseurs** : `JScrollPane` est un conteneur disposant de barres de défilement, verticale et horizontale, uniquement quand c'est nécessaire
  - Ceci permet de visualiser des composants plus grands que l'espace dans lequel ils sont visualisés
- Un **menu déroulant** `JPopupMenu` apparaît quand on fait un clic droit



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

public class BarreOutil extends JFrame {
    private JTextArea zone_texte;
    private JPopupMenu popup;
    // les items du menu contextuel se terminent par la lettre "P"
    private JMenuItem cutP;
    private JMenuItem copyP;
    private JMenuItem pasteP;
    private JMenuItem allP;
    // les boutons requis pour la barre de menu
    private JButton nouv;
    private JButton ouvre;
    private JButton sauve;
    private JButton apropos;

    BarreOutil() {      // constructeur
        getContentPane().setLayout(new BorderLayout());

        zone_texte = new JTextArea("(saisir le texte ici)");
        popup = ajouterPopUp();
        zone_texte.setComponentPopupMenu(popup); // gère l'écoute automatique du popup
                                                 // pas besoin d'installer un écouteur associé
        JToolBar barreOutils = ajouter_barre_outils();
        JScrollPane scroll = new JScrollPane(zone_texte);

        getContentPane().add(barreOutils, BorderLayout.NORTH);
        getContentPane().add(scroll, BorderLayout.CENTER);
    } // fin du constructeur
```

(boutons sans bord)



(méthode de JComponent)

Ajouter la zone texte au ScrollPane

Méthodes données ci-après pour définir la **fenêtre Popup** et la **barre d'outils** de l'éditeur

```java
//-----------------------------------------------------------------------
// methode ajouter_barre_outils de construction de la barre d'outils
//-----------------------------------------------------------------------
public JToolBar ajouter_barre_outils() {
    // définition d'une nouvelle barre
    JToolBar outil = new JToolBar();

    // définition des boutons avec les images :
    nouv = new JButton("Nouveau", new ImageIcon("src/gif/new.gif"));
    ouvre = new JButton("Ouvrir", new ImageIcon("src/gif/open.gif"));
    sauve = new JButton("Enregistrer", new ImageIcon("src/gif/save.gif"));
    apropos = new JButton("A propos...", new ImageIcon("src/gif/about.gif"));

    // propriétés des boutons de la barre d'outil : sans bordure
    nouv.setBorderPainted(false);
    ouvre.setBorderPainted(false);
    sauve.setBorderPainted(false);
    apropos.setBorderPainted(false);

    // enregistrement auprès de la classe écouteur pour la souris :
    nouv.addMouseListener(new PassageDeLaSouris());
    ouvre.addMouseListener(new PassageDeLaSouris());
    sauve.addMouseListener(new PassageDeLaSouris());
    apropos.addMouseListener(new PassageDeLaSouris());

    // enregistrement auprès de la classe écouteur pour les clics souris :
    nouv.addActionListener(new EcouteurMenu());
    ouvre.addActionListener(new EcouteurMenu());
    sauve.addActionListener(new EcouteurMenu());
    apropos.addActionListener(new EcouteurMenu());

    // on ajoute les boutons créés à la barre d'outils :
    outil.add(nouv);
    outil.add(ouvre);
    outil.add(sauve);
    outil.add(apropos);

    // on retourne la barre créée :
    return outil;
} // fin de la methode ajouter_barre_outils()
```

Créer les 4 boutons

Ajouter des écouteurs d'action aux boutons

Ajouter les boutons à la barre d'outils

```java
//-------------------------------------------------------------
// creation d'un pop-up menu
//-------------------------------------------------------------
public JPopupMenu ajouterPopUp() {

    // on instancie le menu contextuel
    JPopupMenu pop = new JPopupMenu("Menu contextuel");

    // initialisation des items du menu contextuel
    cutP = new JMenuItem("Couper", new ImageIcon("cut.gif"));
    copyP = new JMenuItem("Copier", new ImageIcon("copy.gif"));
    pasteP = new JMenuItem("Coller", new ImageIcon("paste.gif"));
    allP = new JMenuItem("Séléctionner tout", new ImageIcon("datePicker.gif"));

    // enregistrement des items au près de l'écouteur approprié
    cutP.addActionListener(new EcouteurMenu());
    copyP.addActionListener(new EcouteurMenu());
    pasteP.addActionListener(new EcouteurMenu());
    allP.addActionListener(new EcouteurMenu());

    // ajout des items au menu contextuel
    pop.add(cutP);
    pop.add(copyP);
    pop.add(pasteP);
    pop.add(allP);

    // on retourne le nenu contextuel
    return pop;
}
```

Créer les menuItem

Associer les écouteurs aux items de menu

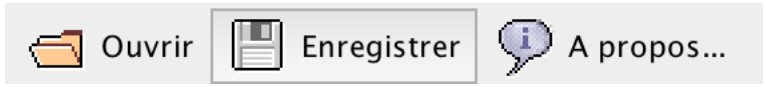
Ajouter les menuItem au menu Popup

```
//-----------------------------------------------------------------------
//   Les classes internes ecouteur
//-----------------------------------------------------------------------
class EcouteurMenu implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        if (source == nouv) {
            zone_texte.setText(new String());  // on vide la zone de texte
            setTitle("Nouveau fichier"); // on modifie le titre de la fenêtre
        }
        if (source == ouvre) {
            String nomfic = ouvrir(); // ouvrir le fichier sélectionné
            if (nomfic != null) {
                zone_texte.setText(new String());
                setTitle(nomfic);
                afficher(nomfic); // afficher le contenu du fichier
            }
        }
        if (source == sauve) {
            String nomfic = sauver();
            ecrire(nomfic);
        }
        if (source == apropos) {
            aPropos();
        }

        if (source == cutP) {
            zone_texte.cut(); // methode de TextArea
        }
        if (source == copyP) {
            zone_texte.copy(); // idem
        }
        if (source == pasteP) {
            zone_texte.paste();
        }
        if (source == allP) {
            zone_texte.selectAll();
        }
    }
}
```

Traitement des actions sur les boutons de la barre d'outils

Traitement des actions sur les items de la fenêtre Popup

```
//-------------------------------------------------------------------------------------
// classe interne écouteur pour encadrer les boutons lorsque l  on passe dessus avec la souris
//-------------------------------------------------------------------------------------

class PassageDeLaSouris extends MouseAdapter {
    public void mouseEntered(MouseEvent e) { // lorsque la souris "entre" sur un des composants écoutés

        if (e.getSource() instanceof JButton) { // le MouseEvent peut provenir d'autres composants
            ((JButton) e.getSource()).setBorderPainted(true); // on fait apparaitre le bord du bouton
        }
    }

    public void mouseExited(MouseEvent e) { // lorsque la souris 'quitte' le composant
        if (e.getSource() instanceof JButton) {
            ((JButton) e.getSource()).setBorderPainted(false);
        }
    }
} // de classe interne PassageDeLaSouris
```



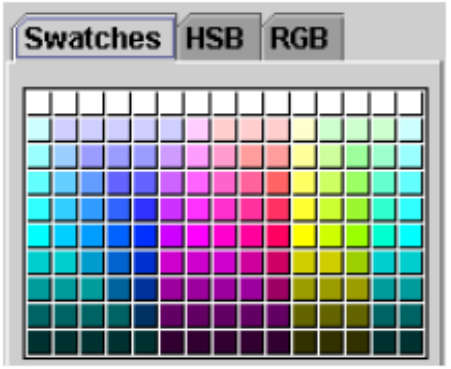
```
//-------------------------------------------------------------------------------------
//  Les méthode utilitaires
//-------------------------------------------------------------------------------------

public String ouvrir() {   // choix du fichier en lecture
    String nomFic = new String("");
    try {
        // chargement de fichier
        FileDialog fd = new FileDialog(this, "Sélectionnez votre fichier...", FileDialog.LOAD);
        fd.setVisible(true);
        nomFic = ((fd.getDirectory()).concat(fd.getFile()));
    } catch (NullPointerException e) {
        System.out.println("Erreur ouverture dossier !");
    }
    return nomFic;
} // fin de ouvrir()
//-------------------------------------------------------------------------------------
// méthode de chargement de la zone de texte depuis le fichier sélectionné en lecture
// lecture en flux caracteres

public void afficher(String nom) {
    try {
        FileReader fichier = new FileReader(nom);
        LineNumberReader lecteur = new LineNumberReader(fichier);
        String ligne = new String("");
        setTitle(nom);
        do {
            ligne = lecteur.readLine();
            zone_texte.append(ligne);
            zone_texte.append("\n\r");
        } while (ligne != null);
        fichier.close();
    } catch (FileNotFoundException e) {
    } catch (IOException e) {
    }
} // fin de afficher()
```

```
//-------------------------------------------------------------------------

public void ecrire(String nom) {    // Ecriture mode caractere dans le fichier
    try {
        FileWriter fic = new FileWriter(nom);
        BufferedWriter buff = new BufferedWriter(fic);
        buff.write(zone_texte.getText());
        buff.close();
        fic.close();
    } catch (IOException e) {
    }

} // de ecrire()
//-------------------------------------------------------------------------

public String sauver() {            // choix du fichier en ecriture
    String nomFic = new String("");
    try {
        FileDialog fd = new FileDialog(this, "Sélectionnez votre  fichier...", FileDialog.SAVE);
        fd.setVisible(true);
        nomFic = ((fd.getDirectory()).concat(fd.getFile()));
    } catch (NullPointerException e) {
    }

    return nomFic;
} // de sauver()
```

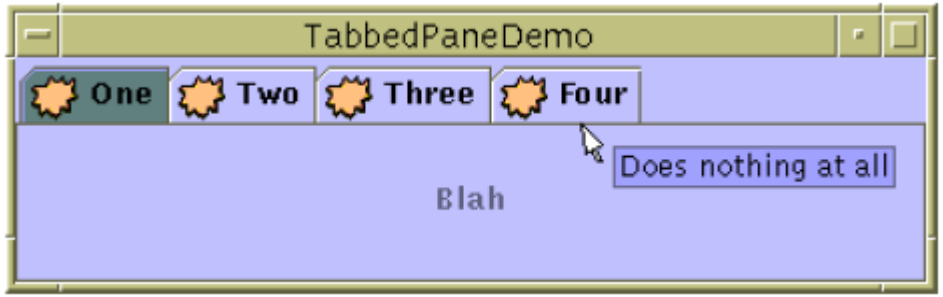
```
    //-----------------------------------------------------------------------
    public void aPropos() {  // Boite d'information
        JOptionPane.showMessageDialog(this, "VDe Corp. 2020", "A propos", JOptionPane.INFORMATION_MESSAGE);
    }
    //-----------------------------------------------------------------------

} // fin de la classe BarreOutil




class TestBarreOutil {

    public static void main(String[] args) {
        //Création de la fenêtre
        BarreOutil frame = new BarreOutil();

        //Affichage de la fenêtre
        frame.setVisible(true);
    }
}
```

Autres composants utiles

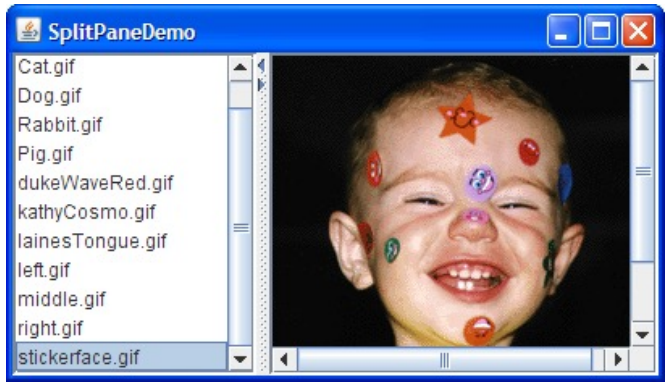
`JColorChooser` : fenêtre de dialogue permettant de choisir une couleur



`JTabbedPane` : permet de mettre plusieurs JPanel dans des onglets



`JPlitPane` : il s'agit d'un double conteneur permettant une interaction entre deux composants



`JScrollPane: un conteneur permettant le défilement (ascenseur) si nécessaire`