Java Avancé - Cours 3 - Swing

Boîtes de Dialogue, Menus, Barres d'outils, Fenêtres modales

V. DESLANDRES, I. GUIDARA

veronique.deslandres@univ-lyon1.fr

Boites de dialogue

Les boîtes de dialogue standard

- On utilise la classe `JOptionPane` pour les boîtes de dialogue standard, prêtes à l'emploi
  - Et des méthodes **statiques : `showXXXDialog()`**

- Quatre types de boîtes
  - `MessageDialog` pour afficher un message
  - `ConfirmDialog` pour une réponse de l'utilisateur avec `Yes`, `No` et `Cancel`
  - `InputDialog` pour une invite de saisie
  - `OptionDialog` qui rassemble les caractéristiques des 3 autres types de boîtes de dialogue

Message Dialog

Caractéristiques :

- Un texte, un bouton OK et éventuellement un icône prédéfini

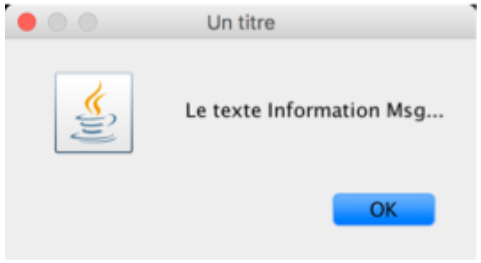
```
JOptionPane.showMessageDialog(fen, "Le texte Information Msg...", "Un titre", JOptionPane.INFORMATION_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Warning Msg...", "Un titre", JOptionPane.WARNING_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Error Msg...", "Un titre", JOptionPane.ERROR_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Question Msg...", "Un titre", JOptionPane.QUESTION_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Plain Msg...", "Un titre", JOptionPane.PLAIN_MESSAGE);
```

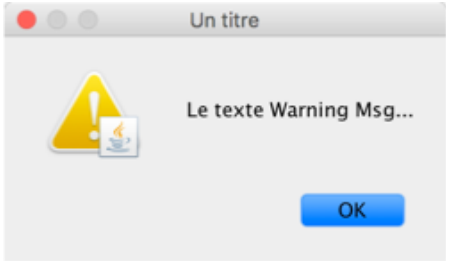
Fenêtre parent

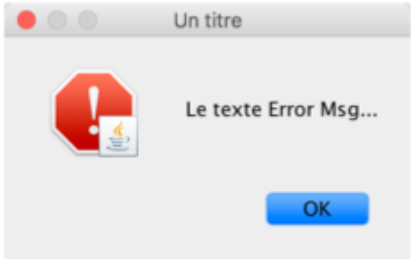
le texte

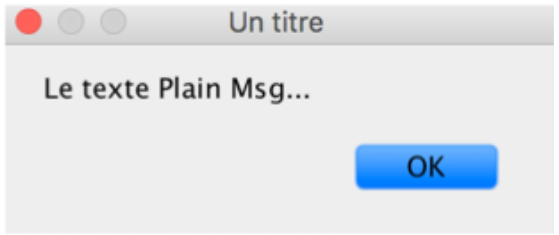
le titre

type d'icône (énumeration)



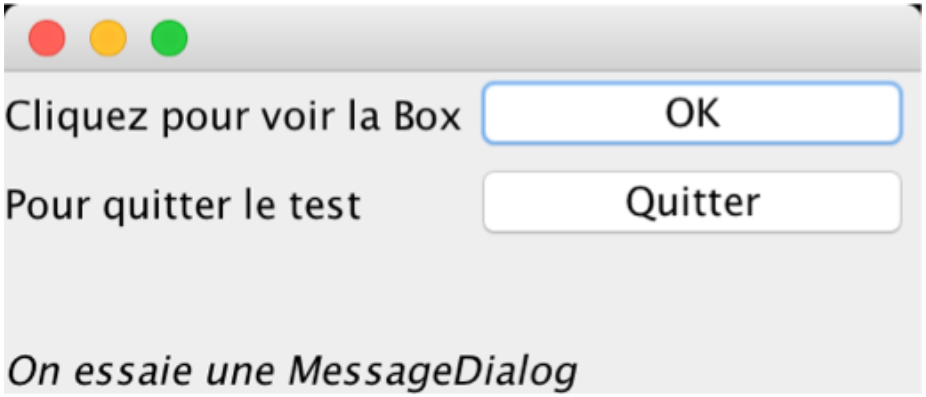






Message Dialog (2)

- Le 1er argument est une `JFrame`
- Si on met **null**, ça marche aussi et la fenêtre sera **centrée sur l'écran**
- Si on met une fenêtre parent, la fenêtre Dialog sera centrée sur cette dernière :



- Ceci est valable pour toutes les fenêtres de Dialog

Confirm Dialog

Caractéristiques :

- Une question, de 1 à 3 boutons (Oui/OK, Non, Annuler) et un icône

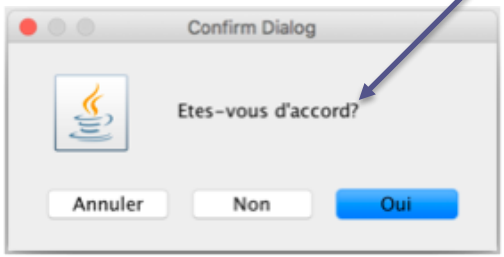
```
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.YES_NO_CANCEL_OPTION);
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.YES_NO_OPTION);
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.OK_CANCEL_OPTION);
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.DEFAULT_OPTION);
```

Fenêtre parent

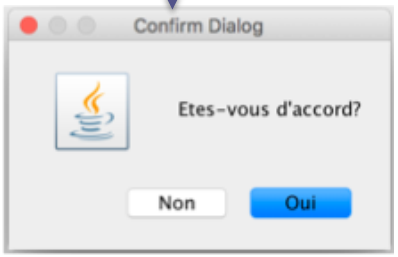
message

titre

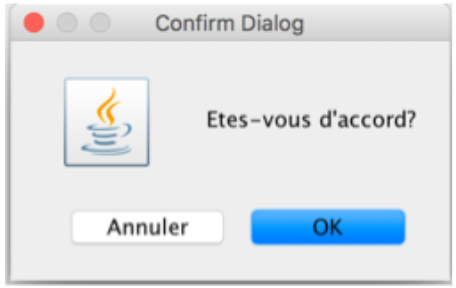
Option Bouton (énumeration)



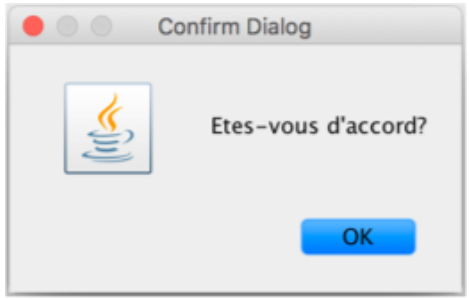
YES_NO_CANCEL_OPTION



YES_NO_OPTION



OK_CANCEL_OPTION



DEFAULT_OPTION

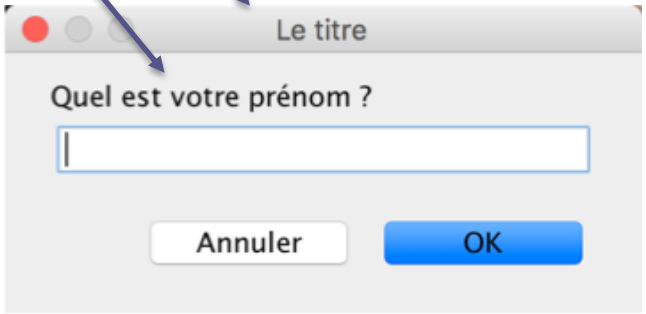
Input Dialog

Caractéristiques :

- Une question, une invite de saisie, 2 boutons (OK et annuler) et un icône

```
String message = "Quel est votre prénom ?";

String reponse = JOptionPane.showInputDialog(this, message, "Le titre", JOptionPane.PLAIN_MESSAGE);
// ou WARNING_MESSAGE, INFORMATION_MESSAGE
// ou ERROR_MESSAGE, PLAIN_MESSAGE);
```

Fenêtre parent



(avec ici *information* ou *question message*)

Option Dialog (1/2)

Caractéristiques :
- Une question avec des boutons représentant des choix différents et un icône

```
int showOptionDialog(Component parentComponent,
                     Object message,
                     String title,
                     int optionType,
                     int messageType,
                     Icon icon,
                     Object[] options,
                     Object initialValue)
```

liés

Soit *optionType* est à 0 et on donne des *options*

Soit un *optionType* est fourni (ex. YES_NO_OPTION), et *options* est **null**

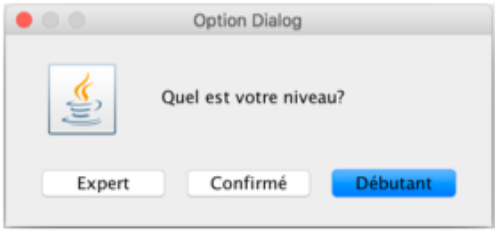
Ouvre une fenêtre de dialogue avec **l'icône proposé**, les **options** possibles, et le choix initial qui est déterminé par le paramètre initialValue.

La méthode renvoie l'indice de l'option (dans le tableau) choisie par l'utilisateur

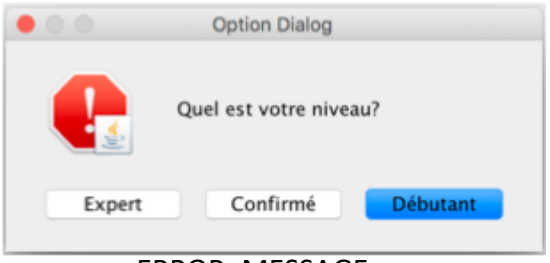
Option Dialog (2/2)

```
String[] choix={"Débutant", "Confirmé", "Expert"};
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.INFORMATION_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.ERROR_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.WARNING_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.QUESTION_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.PLAIN_MESSAGE, null, choix, choix[1]);
```

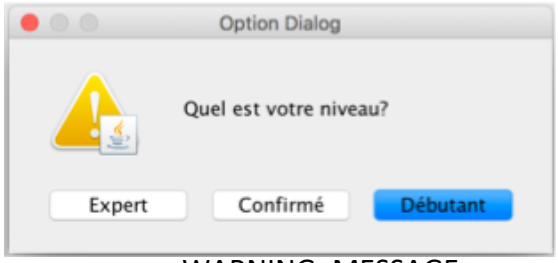
Fenêtre parent — Texte du msg — Titre — (Option Yes/No) — type icône — (icône) — liste des choix — Choix présélectionné



INFORMATION_MESSAGE

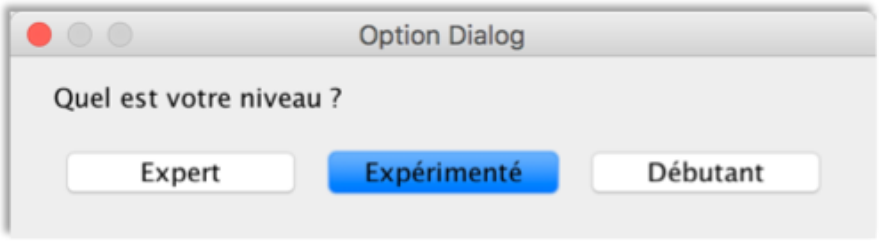


ERROR_MESSAGE



WARNING_MESSAGE

QUESTION_MESSAGE



PLAIN_MESSAGE

Exemples d'utilisation

Confirm Dialog

```
int response=JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",
        JOptionPane.YES_NO_CANCEL_OPTION);
if(response==JOptionPane.YES_OPTION)
    System.out.println("Yes Option !");
if(response==JOptionPane.NO_OPTION)
    System.out.println("No Option !");
if(response==JOptionPane.CANCEL_OPTION)
    System.out.println("Cancel Option !");
if(response==JOptionPane.CLOSED_OPTION)
    System.out.println("Closed Option !");
```

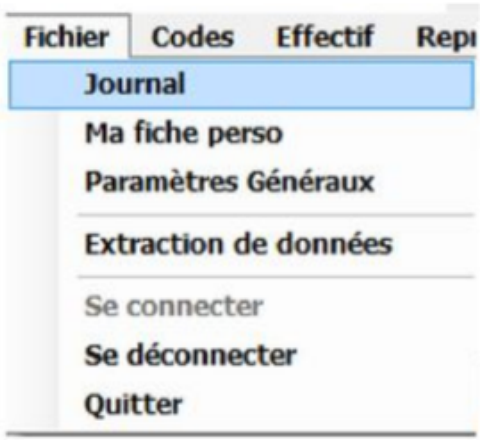
Input Dialog

```
String reponse = JOptionPane.showInputDialog(fen, "Quel est votre prénom ?", "Le titre",
        JOptionPane.INFORMATION_MESSAGE);
if (reponse.length() != 0) {
    System.out.println("Bonjour " + reponse);
}
```

Option Dialog

```
int response=JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,
        JOptionPane.INFORMATION_MESSAGE, null, choix, choix[0]);
if(response==JOptionPane.CLOSED_OPTION)
    System.out.println("Pas de formule choisie !");
else
    System.out.println("Vous avez choisi: "+choix[response]);
```

Les Menus

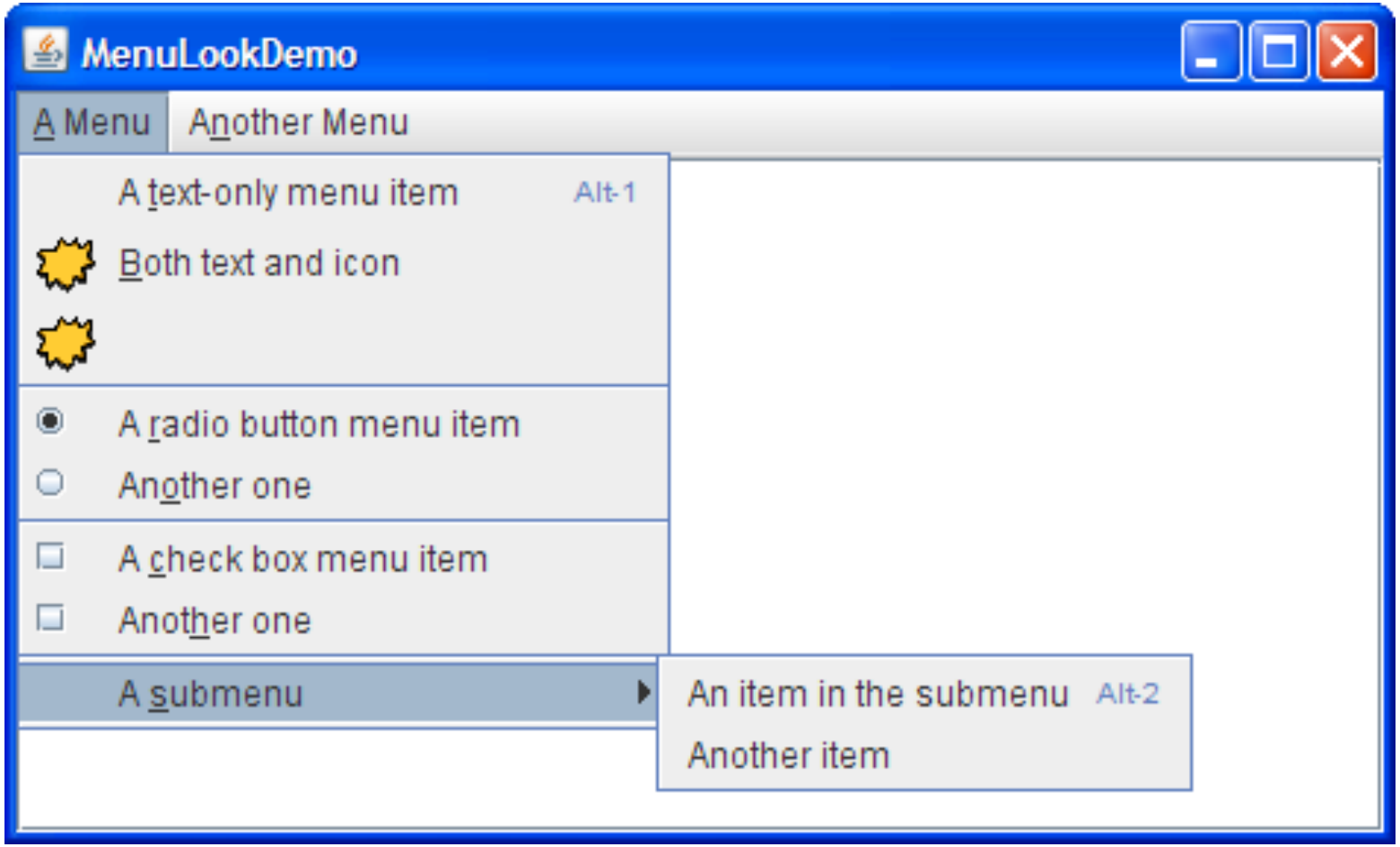

Fichier
Codes
Effectif
Repı
Journal
Ma fiche perso
Paramètres Généraux
Extraction de données
Se connecter
Se déconnecter
Quitter


Repro
Mise Bas
Mesures
Chantier
Lait
Campagne
Lot de Lutte
Choix des femelles
Choix des mâles
Paillettes semence
Embryons
Traitement Hormonal
Liste et MAJ des oestrus
MAJ du Plan d'accouplement
Liste et MAJ des saillies
Endoscopie
Gestation
Liste et Inventaire des reproducteurs

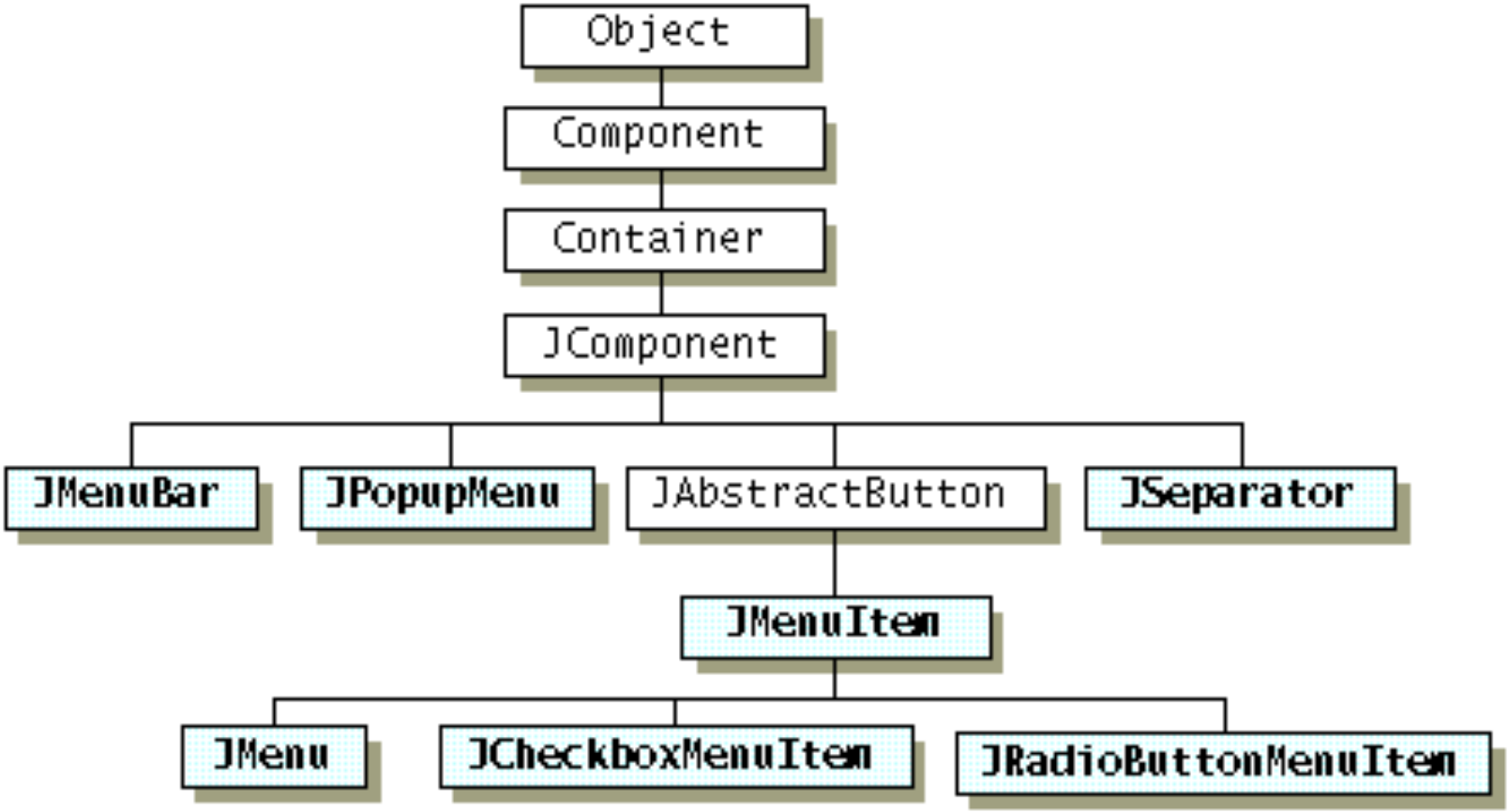
Les classes de menus

- 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**
  - soit dans un menu **déroulant**

Les menus (exemple)

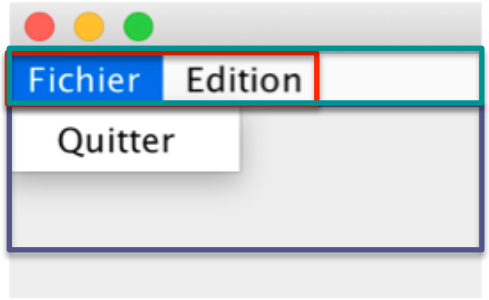
```
JMenuItem exit=new JMenuItem("Quitter");
JMenuItem option1=new JMenuItem("Option-1");
JMenuItem option2=new JMenuItem("Option-2");
JMenu fichier=new JMenu("Fichier");
fichier.add(exit);
JMenu edition=new JMenu("Edition");
edition.add(option1);
edition.add(option2);
JMenuBar mb=new JMenuBar();
mb.add(fichier);
mb.add(edition);
setJMenuBar(mb);
```

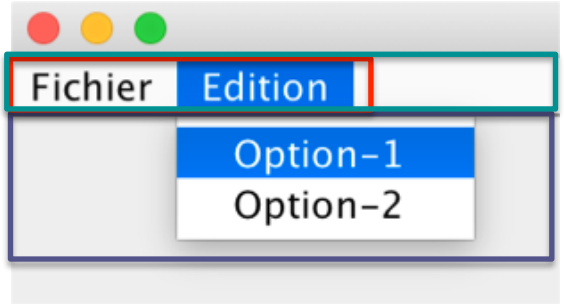
Créer les 3 MenuItems

Créer les 2 Menus (Fichier et Edition) et ajouter les MenuItems aux Menus

Créer la barre de menu et y ajouter les 2 Menus

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





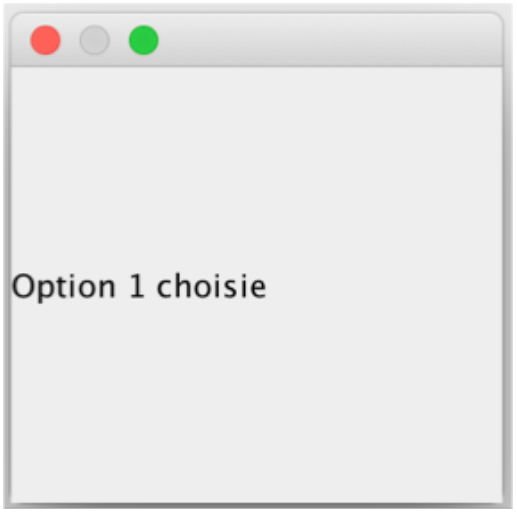
Les menus (suite)

```
exit.addActionListener(new MyActionListener());
option1.addActionListener(new MyActionListener());
option2.addActionListener(new MyActionListener());
```

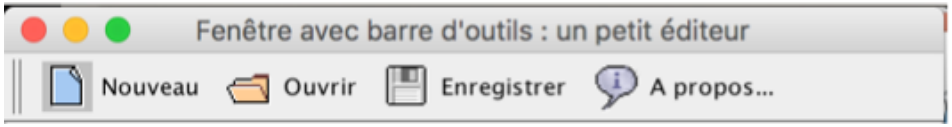
Ajouter un écouteur à chaque item de menu

```
JDialog jd=new JDialog();
JLabel jl=new JLabel();
if(e.getSource()==exit)
    System.exit(0);
if(e.getSource()==option1)
    jl=new JLabel("Option 1 choisie");
else if(e.getSource()==option2)
    jl=new JLabel("Option 2 choisie");

jd.setSize(200,200);
jd.getContentPane().add(jl);
jd.setVisible(true);
```

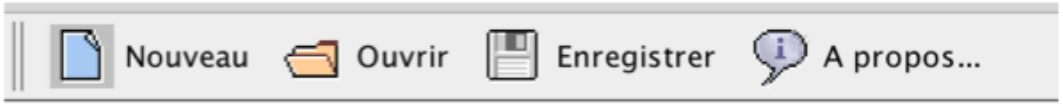
Exemple de traitement quand on clique sur chaque item de menu



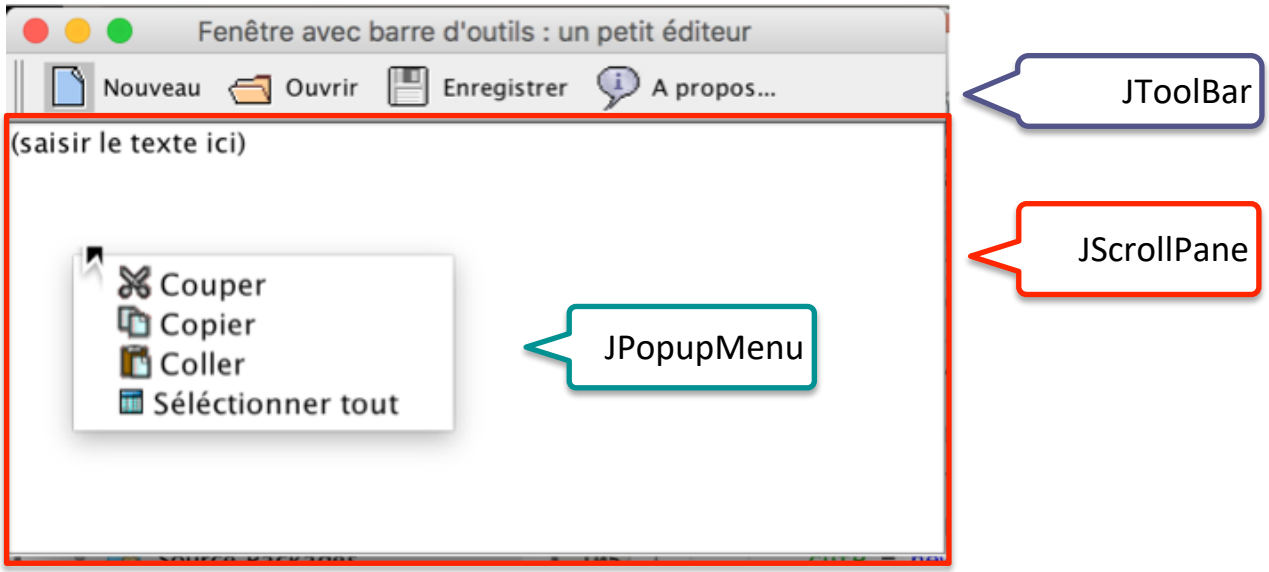
Les barres d'outils



Barre d'outils

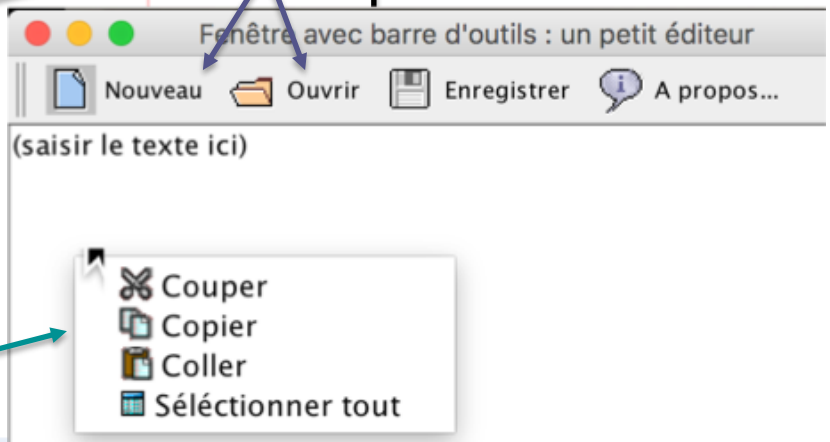


- Illustration sur un **petit éditeur**
- La fenêtre est munie d'**ascenseurs** : `JScrollPane` est un conteneur disposant de barres de defilement, verticale et horizontale, uniquement quand c'est nécessaire
  - Ceci permet de visualiser des composants plus grands que l'espace dans lequel ils sont visualisés.
- 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)



Ajouter la zone texte au ScrollPane

Méthodes déf. 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 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

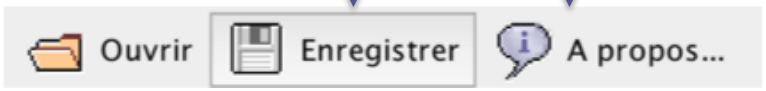
Ajouter les menuItem au menu Popup

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

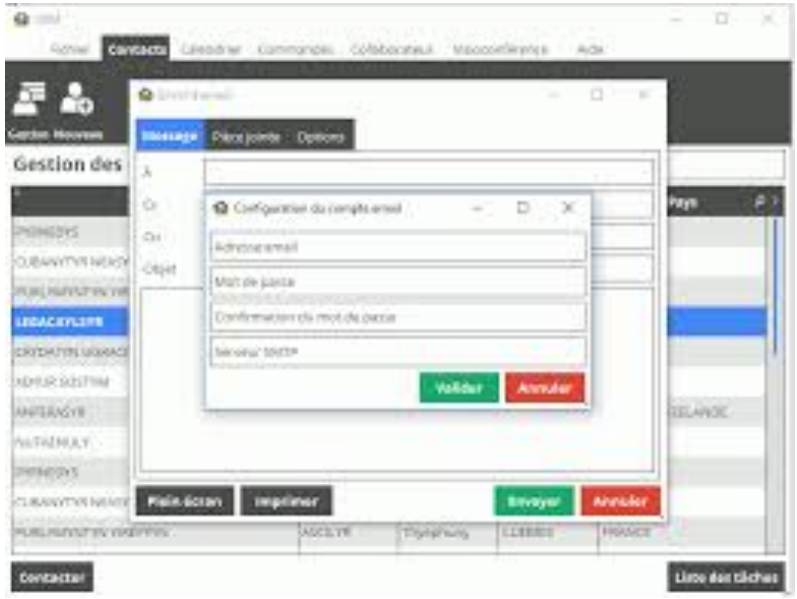


```
//-------------------------------------------------------------------------------------
//  Les méthode utilitaires
//-------------------------------------------------------------------------------------

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 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;
    } // fin de sauver()

    //-------------------------------------------------------------------------------------------

    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) {
        }

    } // fin de ecrire()

    //-------------------------------------------------------------------------------------------
    public void aPropos() {  // Boite d'information
        JOptionPane.showMessageDialog(this, "VDe Corp. 2004", "A propos", JOptionPane.INFORMATION_MESSAGE);
    }
} //fin de la classe Fenetre
```

```
class TestBarreOutils {
    public static void main(String[] args) {
        //Création et affichage de la fenêtre
        JFrame frame = new BarreOutil();
        frame.setSize(700, 400);
        frame.setTitle("Fenêtre avec barre d'outils : un petit éditeur");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Construire ses propres Fenêtres Modales



Fenêtre Modale

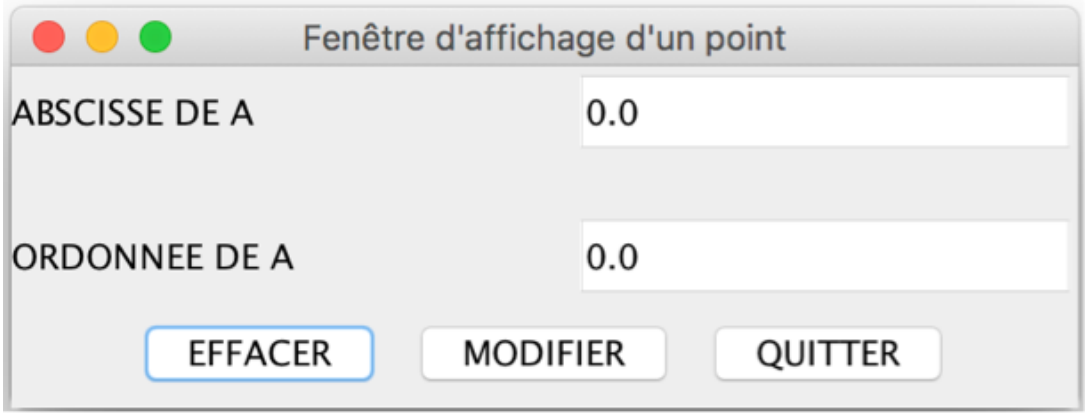
- DEF C'est une fenêtre qui **dépend** d'une autre fenêtre et qui **prend le contrôle** du clavier et de l'écran.
- Associée généralement à une question à laquelle il est impératif que l'utilisateur réponde, avant de pouvoir à nouveau interagir avec le reste du programme

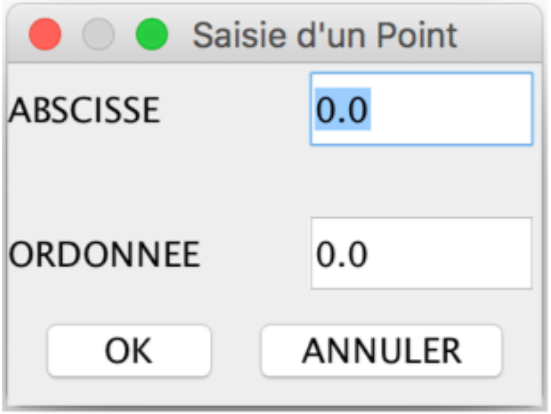
- La fenêtre modale permet :
  - d'obtenir des informations de l'utilisateur qui sont nécessaires pour réaliser une opération
  - de fournir une information à l'utilisateur qui doit en prendre connaissance avant de pouvoir continuer à utiliser l'application

- Pour saisir des données de l'application, on utilise souvent la classe `JDialog`
  - semblable à celle de `JFrame`
  - permet la création de fenêtres modales ou non modales

Fenêtre Modale (Exemple)

- Exemple:
  - on a la classe `Point` avec 2 coordonnées réelles `X` et `Y`
  - on réalise un petit programme de saisie, d'affichage ou de 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



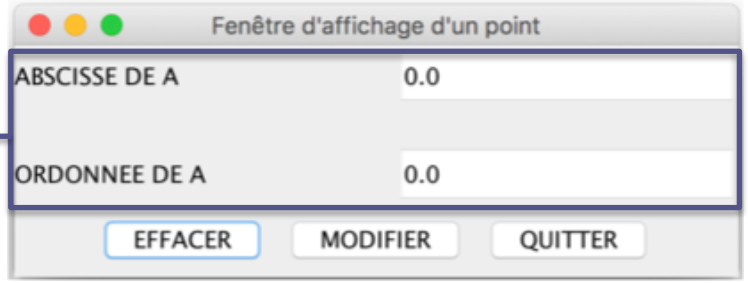


Fenêtre Parent (1/2)

```
public class FenetreModale extends JFrame implements ActionListener{
    JTextField xa, ya;
    JLabel abscisse, ordonnée, d;
    JButton btnE, btnM, btnQ;
    Point pointA=new Point(0,0);
    JPanel pan;
    FenetreModale(){  //Constructeur
        pan=(JPanel) getContentPane();
        pan.setLayout(new BorderLayout());
        pan.add(panneau_bas(), BorderLayout.SOUTH);
        pan.add(panneau_milieu(), BorderLayout.CENTER);
    }

    JPanel panneau_milieu(){ //Définir le panneau au milieu
        JPanel milieu=new JPanel();
        milieu.setLayout(new GridLayout(2,2,20,20));

        abscisse=new JLabel("ABSCISSE DE A");
        ordonnée=new JLabel("ORDONNEE DE A");

        xa=new JTextField(20);
        ya=new JTextField(20);

        xa.setEditable(false);
        ya.setEditable(false);

        milieu.add(abscisse);
        milieu.add(xa);
        milieu.add(ordonnée);
        milieu.add(ya);

        return milieu;
    }
```

Classe Point du package awt

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



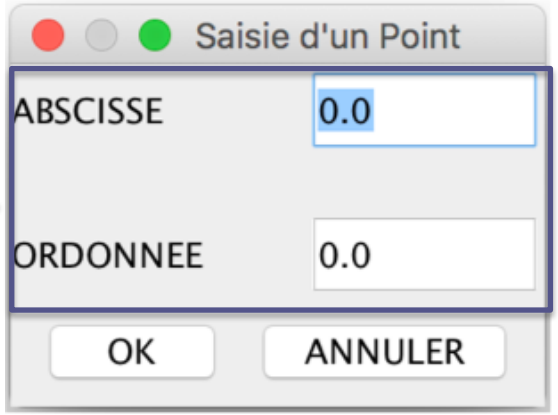
Fenêtre Parent (2/2)

```
    JPanel panneau_bas(){
        JPanel bas=new JPanel();
        btnE=new JButton("EFFACER");
        btnM=new JButton("MODIFIER");
        btnQ=new JButton("QUITTER");
        bas.add(btnE);
        bas.add(btnM);
        bas.add(btnQ);
        btnE.addActionListener(this);
        btnM.addActionListener(this);
        btnQ.addActionListener(this);
        return bas;
    }
    public void actionPerformed(ActionEvent e){
        if(e.getSource()==btnE){
            pointA.setLocation(0,0);
            xa.setText(String.valueOf(pointA.getX()));
            ya.setText(String.valueOf(pointA.getY()));
        }
        else if(e.getSource()==btnM){
            FenSaisiePoint fenSaisie= new FenSaisiePoint(this, pointA);
            if(fenSaisie.afficheModale()){
                xa.setText(String.valueOf(pointA.getX()));
                ya.setText(String.valueOf(pointA.getY()));
            }
        }
        else if(e.getSource()==btnQ){
            this.dispose();
        }
    }
    public static void main(String[] args){
        JFrame fen=new FenetreModale();
        fen.setBounds(10, 20, 400, 150);
        fen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fen.setTitle("Fenêtre d'affichage d'un point");
        fen.setVisible(true);
    }
}
```

Fenêtre Saisie (1/2)

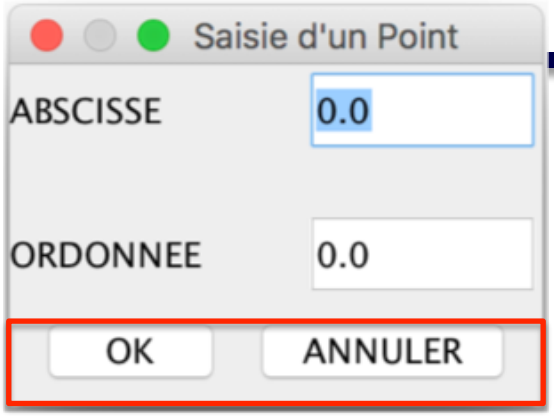
```
public class FenSaisiePoint extends JDialog implements ActionListener{
    Point p;
    JTextField x,y;
    JButton btOK, btQUIT;
    boolean OKchoisi;

    FenSaisiePoint(JFrame f, Point p){
        super(f, "Saisie d'un Point", true);
        this.p=p;
        setLocationRelativeTo(f);
        setBounds(200,300,200,150);
        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);
        //pour récupérer les coordinnées du point
        x.setText(String.valueOf(p.getX()));
        y.setText(String.valueOf(p.getY()));
    }
    JPanel panneau_milieu(){
        JPanel milieu=new JPanel();
        milieu.setLayout(new GridLayout(2,2,20,20));

        JLabel abscisse=new JLabel("ABSCISSE");
        JLabel ordonnée=new JLabel("ORDONNEE");

        x=new JTextField(20);
        y=new JTextField(20);
        x.setEditable(true);
        y.setEditable(true);

        milieu.add(abscisse);
        milieu.add(x);
        milieu.add(ordonnée);
        milieu.add(y);
        return milieu;
    }
```

// true pour modale



Fenêtre Saisie (2/2)

```
JPanel panneau_bas(){
    JPanel bas=new JPanel();
    btOK=new JButton("OK");
    btOK.addActionListener(this);
    bas.add(btOK);
    btQUIT=new JButton("ANNULER");
    btQUIT.addActionListener(this);
    bas.add(btQUIT);
    return bas;
}
boolean afficheModale(){
    setVisible(true);
    return OKchoisi;
}
public void actionPerformed(ActionEvent e){
    if(e.getSource()==btOK){
        double vx, vy, d1;
        try{
            vx=(new Double(x.getText())).doubleValue();
        }
        catch(RuntimeException ex){
            vx=p.getX();
            x.setText(String.valueOf(p.getX()));
        }
        try{
            vy=(new Double(y.getText())).doubleValue();
        }
        catch(RuntimeException ex){
            vy=p.getY();
            y.setText(String.valueOf(p.getY()));
        }
        p.setLocation(vx, vy);
        OKchoisi=true;
        dispose();
    }
    else{
        OKchoisi=false;
        dispose();
    }
}
// Fin de la classe principale
```



On récupère la valeur des champs X et Y et on les transmet au Point p

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

redonne la main à la fenêtre parent (Fenêtre Modale.java)