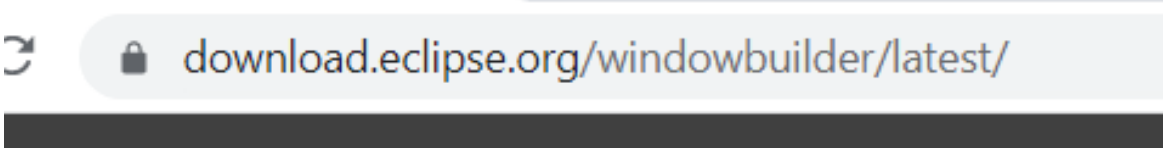
Labo 4 Labo : Environnement Graphique en Java

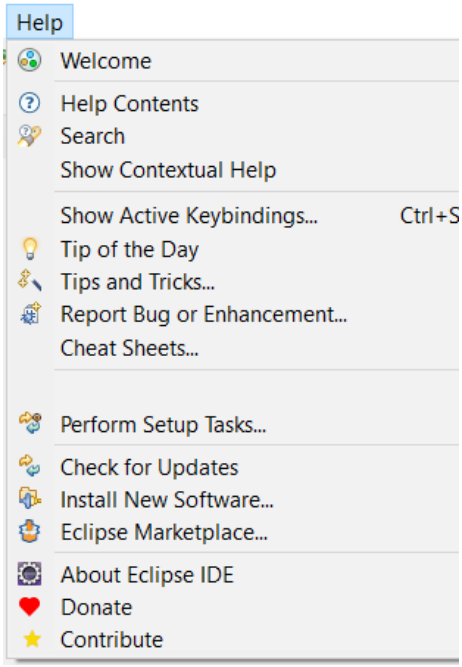
Objectifs :
Rappel du Design Pattern MVC
Intégration de l'environnement graphique, installation plug-in windowBuilder
Programmation événementielle

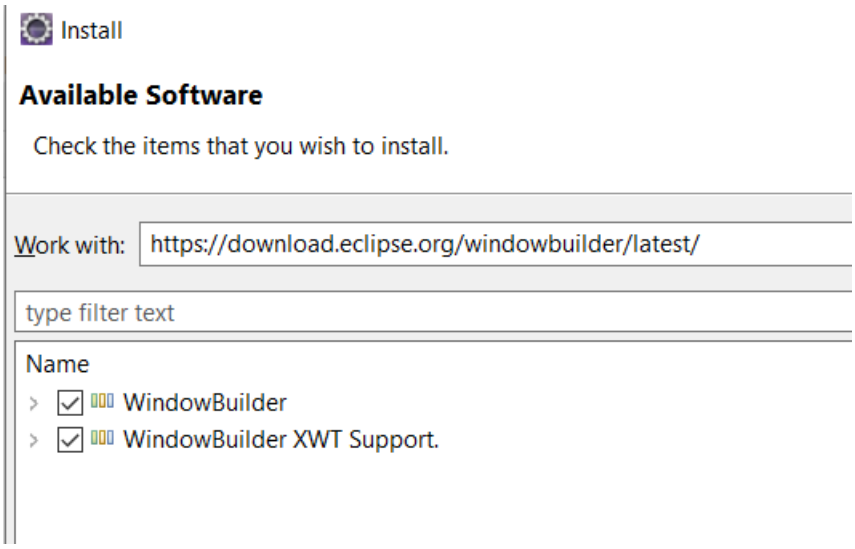
Dans le cadre du développement d'applications utilisateurs, la quasi-totalité des développements aboutie à une application « graphique », type « windows ».
La plateforme de développement Eclipse (IDE) est a priori destinée actuellement à développer des applications en mode caractères / console elle ne dispose pas actuellement de concepteur d'interface graphique.
Afin de pouvoir répondre à cette attente il nous faut installer un composant à Eclipse qui nous permettra de disposer de ces outils, il s'agit de WindowBuilder.

I) Installation de WindowBuilder dans Eclipse… (windowbuilder)



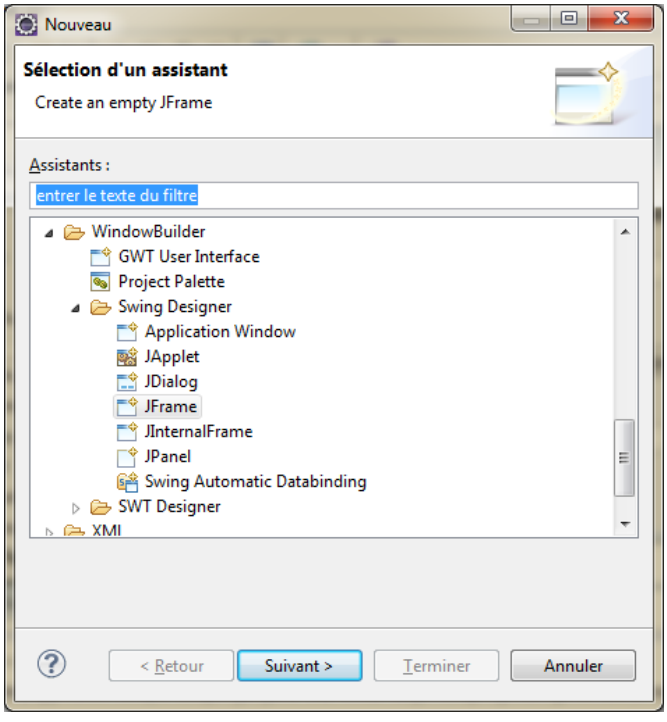
Pour ajouter WindowBuilder à votre eclipse, il vous suffit de vous rendre dans le menu help, puis de sélectionner « install new software ».
Dans la barre d'adresse, indiquez le chemin du upload, puis cliquez sur le bouton add.
Vous n'avez plus alors qu'à sélectionner les composants qui apparaissent dans la zone centrale de l'écran, puis laissez travailler.

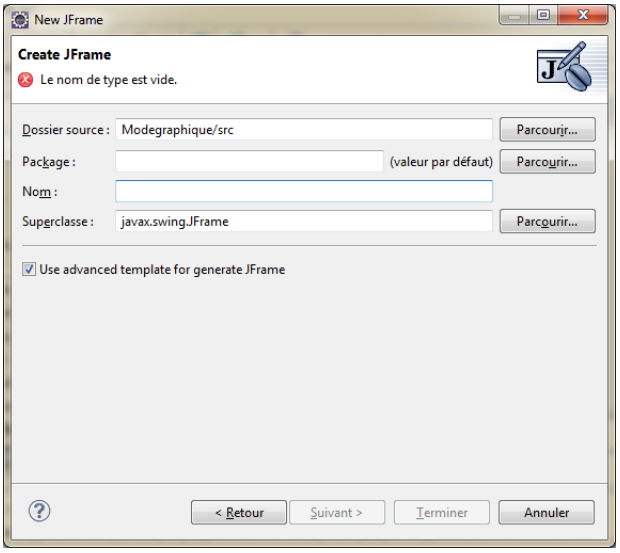




<u>Pour finaliser l'installation il vous faudra relancer Eclipse.</u>

Construire des IHM ?
En fait il faut se baser sur une chose simple, une interface graphique est une classe fille de la classe Jframe. Partant de ce concept, il vous reste à utiliser les méthodes héritées et le concepteur graphique proposé par WindowBuilder

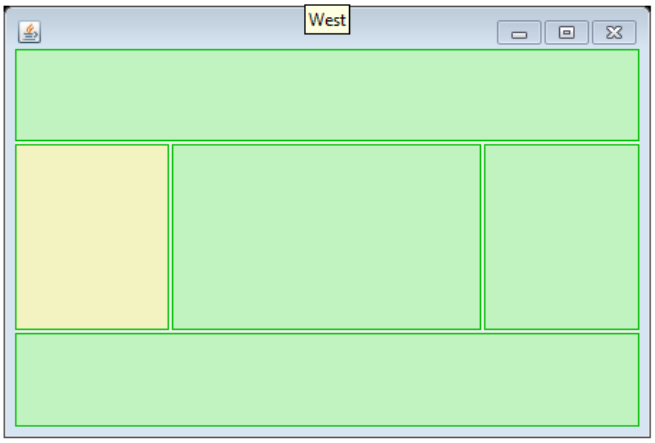


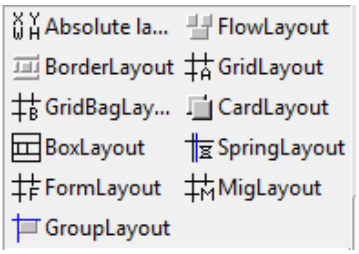


Notion de base à retenir, le mode graphique suppose de « jouer » avec des composants graphiques que l'on positionne sur une Jframe comme si on collait des gommettes sur une feuille de papier.

Pour cela il faut tout d'abord tenir compte du Layout, (**le gestionnaire de positionnement**), par défaut il organise la feuille selon les points cardinaux (BorderLayout).

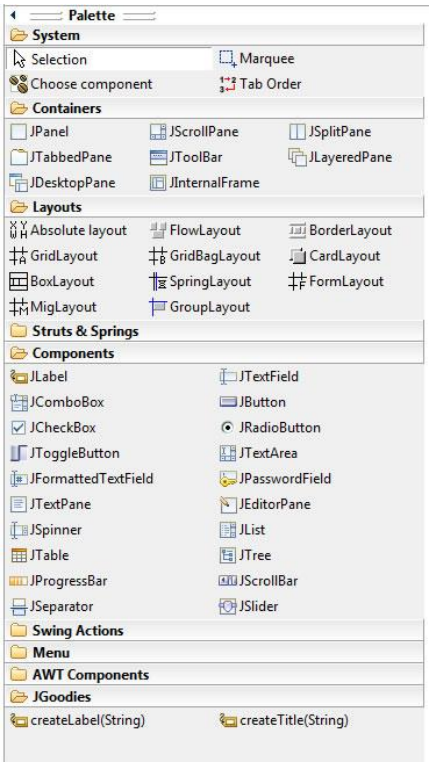
Première étape : changer de layout

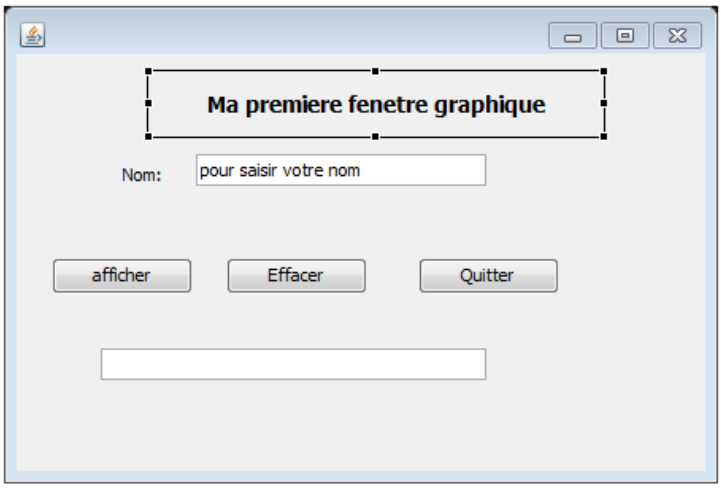




Mise en place des contrôles graphiques :

Il suffit de se servir dans la palette d'outils qui vous propose une multitude de contrôles graphiques et de les apposer et tracer sur la feuille.

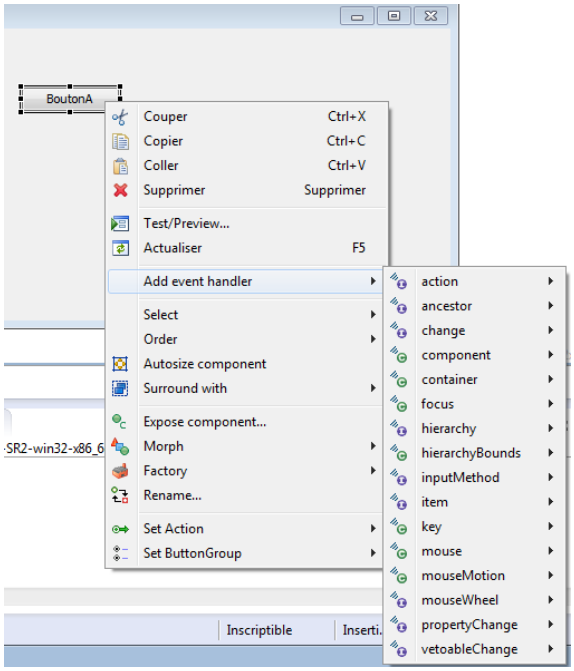




Mise en place d'actions spécifiques associées aux contrôles :

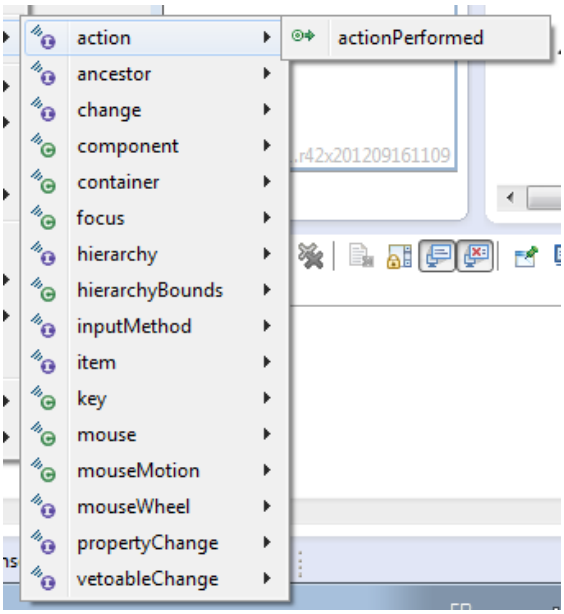
Cela revient à faire de la programmation événementielle tout simplement, voilà comment faire.

- Soit double cliquer sur l'élément (événement par défaut)
- Soit choisir l'événement dans la liste proposée suite au clic droit (Add event handler)



Tout comme en Visual Basic ou C#, à chacun des contrôles correspond une liste d'événements et un type d'écouteur. (Nous reviendrons plus tard sur la notion d'écouteur)

CHOISIR DES EVENEMENTS PARTICULIERS
Toutefois il vous est possible de rechercher d'autres événements que celui ou ceux par défaut, pour cela il vous suffit de choisir l'option Add event handler pour disposer d'autres événements.



Au travers des versions actuelles d'éclipse ou de Netbeans, lorsque vous changez d'événements l'écouteur associé l'est également. Toutefois il vous est toujours possible de définir votre organisation d'écouteurs.

Extrait de code généré à compléter :

```
/**
 * Create the frame.
 */
public Essai() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JButton btnBouton = new JButton("BoutonA");
    btnBouton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {


        }
    });
    btnBouton.setBounds(49, 52, 89, 23);
    contentPane.add(btnBouton);
}
```

Liste des événements et écouteurs

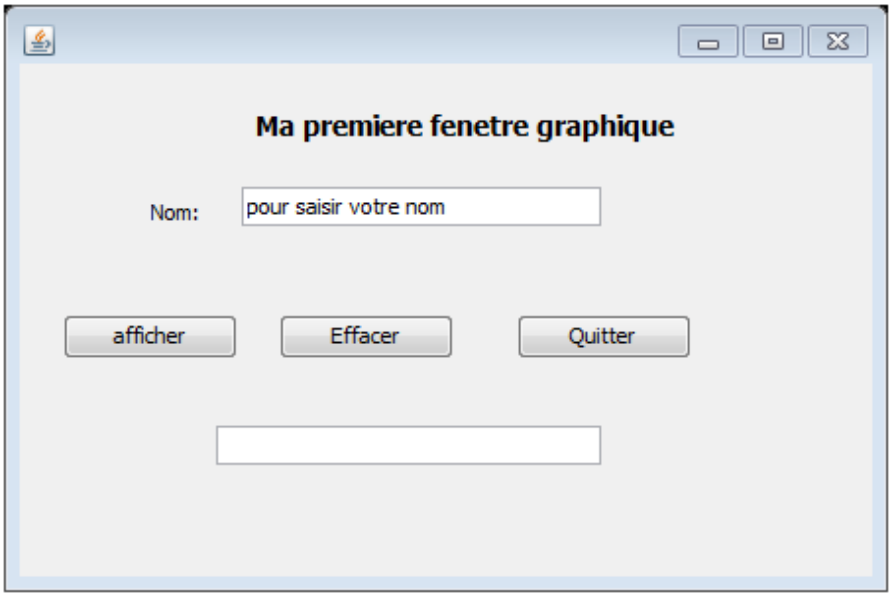
Ecouteur	Méthode écouteur	Type événement	Méthodes événement	Composant ou contrôle concerné
MouseListener	mouseClicked mousePressed mouseReleased mouseEntered mouseExited	MouseEvent	getClickCount getComponent getModifiers getSource getX getY getPoint isAltDown isAltGraphDown isControlDown isMetaDown isPopupTrigger isShiftDown	Component
MouseMotionListener	mouseDragged mouseMoved			
KeyListener	keyPressed keyReleased keyTyped	keyEvent	getComponent getSoruce getKeyChar getKeyCode getKeyModifiersText getKeyText getModifiers isAltDown isAltGraphDown isControlDown isMetaDown isActionKey	Component
FocusListener	focusGained focusLost	FocusEvent	getComponent getSource isTemporary	Component
WindowsListener	windowOpened windowClosing windowClosed windowActivated windowDeactivated windowIconified windowDeiconified	WindowEvent	getComponent getSource getWindow	Window
ActionListener	actionPerformed	ActionEvent	getSource getActionCommand getModifiers	Boutons Menus JTextField
ItemListener	itemStateChanged	ItemEvent	getSource getItem getStateChange	Boutons Menus JList JComboBox
ListSelectionListener	valueChanged	ListSelectionEvent	getSource getValuesAdjusting	JList
DocumentListener	changeUpdate insertUpdate removeUpdate	DocumentEvent	getDocument	Document
MenuListener	menuCanceled menuSelected menuDeselected	MenuEvent	getSource	JMenu
PopupMenuListener	popupMenuCanceled popupMenuWillBecomeVisible popupMenuWillBecomeInvisible	PopupMenuEvent	getSource	JPopupMenu

Les termes Boutons et Menus englobent les contrôles suivants :

Boutons : JButton, JCheckBox, JRadioButton,
Menus : JMenu, JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem

Bilan

En se basant sur l'exemple suivant :



Partie construction IHM

Composition de la fenetre

```
/**
 * Create the frame.
 */
public Mafenetre() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lbl_titre = new JLabel("Ma premiere fenetre graphique");
    lbl_titre.setHorizontalAlignment(SwingConstants.CENTER);
    lbl_titre.setFont(new Font("Tahoma", Font.BOLD, 14));
    lbl_titre.setBounds(83, 11, 287, 41);
    contentPane.add(lbl_titre);

    txtNom = new JTextField();
    txtNom.setText("pour saisir votre nom");
    txtNom.setBounds(113, 63, 183, 20);
    contentPane.add(txtNom);
    txtNom.setColumns(10);

    JLabel lblNom = new JLabel("Nom:");
    lblNom.setBounds(67, 63, 44, 27);
    contentPane.add(lblNom);

    textField = new JTextField();
    textField.setBounds(100, 185, 196, 20);
    contentPane.add(textField);
    textField.setColumns(10);
```

Comportements de certains éléments

```
    JButton btn_effacer = new JButton("Effacer");
    btn_effacer.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            txtNom.setText("");
        }
    });
    btn_effacer.setBounds(132, 128, 89, 23);
    contentPane.add(btn_effacer);

    JButton btn_quitter = new JButton("Quitter");
    btn_quitter.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            dispose();
        }
    });
    btn_quitter.setBounds(253, 128, 89, 23);
    contentPane.add(btn_quitter);

    JButton btnAfficher = new JButton("afficher");
    btnAfficher.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            textField.setText(txtNom.getText());
            }
    });
    btnAfficher.setBounds(22, 128, 89, 23);
    contentPane.add(btnAfficher);

}
```

Conseils : qui dit application graphique dit…

Conception d'interface professionnelle, respectant la plupart des consignes de conception graphique (UX et UI)
Homogénéité des contrôles graphiques utilisés (type, dimensions, couleurs, disposition)
Alignement des éléments de l'interface
Homogénéité des polices et tailles des polices
Une interface intuitive….

II) Mise en pratique : exercices authentification et devises

Exercice N°1 : basé sur l'intégration d'une IHM dans un projet JAVA
Construire un écran d'authentification pour une application.

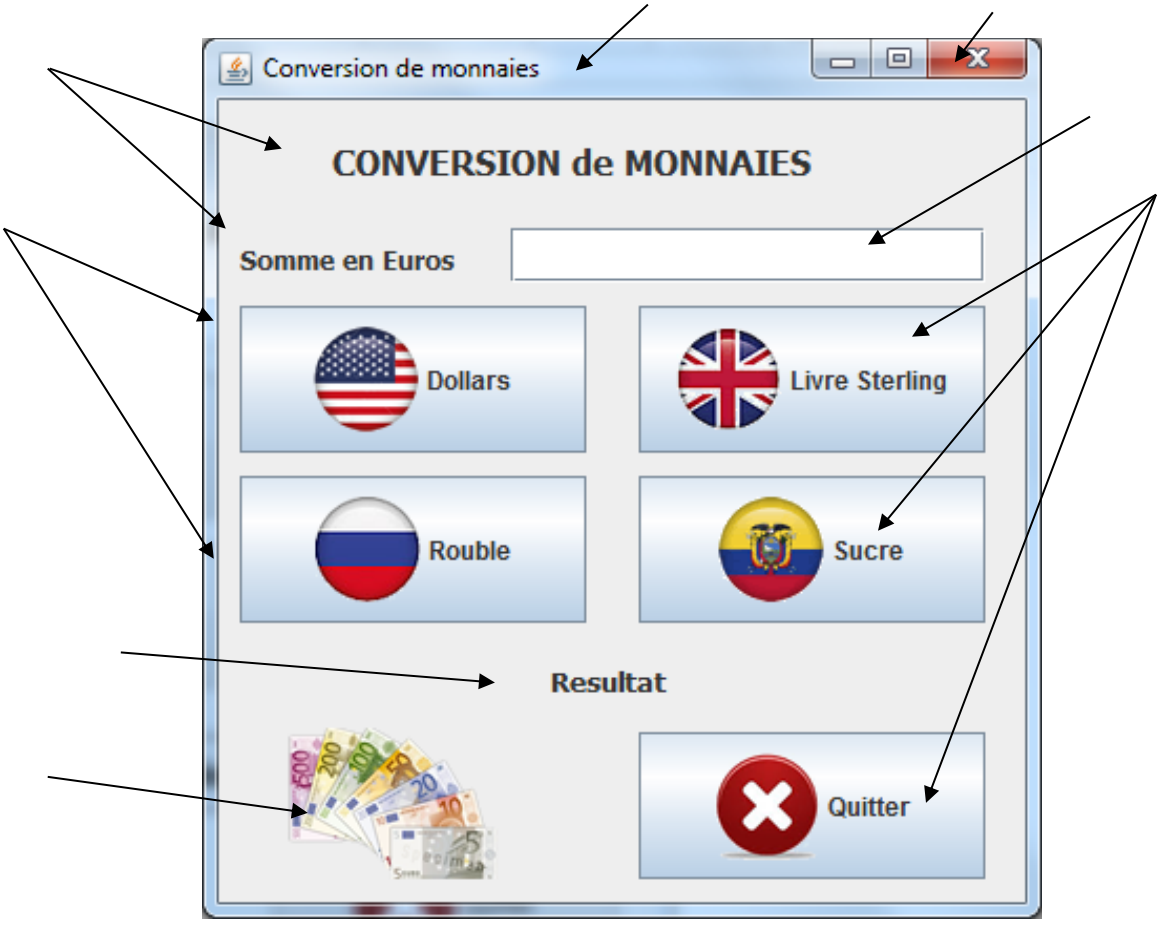
Consignes

- Vous créerez un paquetage « Vue » préalablement dans votre projet
- Il faut systèmatiquement renommer les controles utilisés (de manière normalisée)
- Ecran d'authentification classique avec un champ login en clair et un champ mot de passe en caché. (pour la validation des données saisies, **exceptionnellement** on verifiera avec des valeurs en dur dans le code)

Exercice N°2 : basé sur l'intégration d'une IHM dans un projet JAVA
Construire un écran d'authentification pour une application.

Consignes

- Reproduire l'interface ci-dessous



Il faut penser également à coder la croix de sortie située en haut à droite de la fenêtre

T.A.F :
1. Identifier les composants graphiques à utiliser.
2. Concevoir l'interface proposée ci-dessus dans le paquetage « Vue ».
3. Il faut systèmatiquement renommer les controles utilisés (de manière normalisée)
4. Mettre en place le code de l'application, qui rendra l'application foncitonnelle.