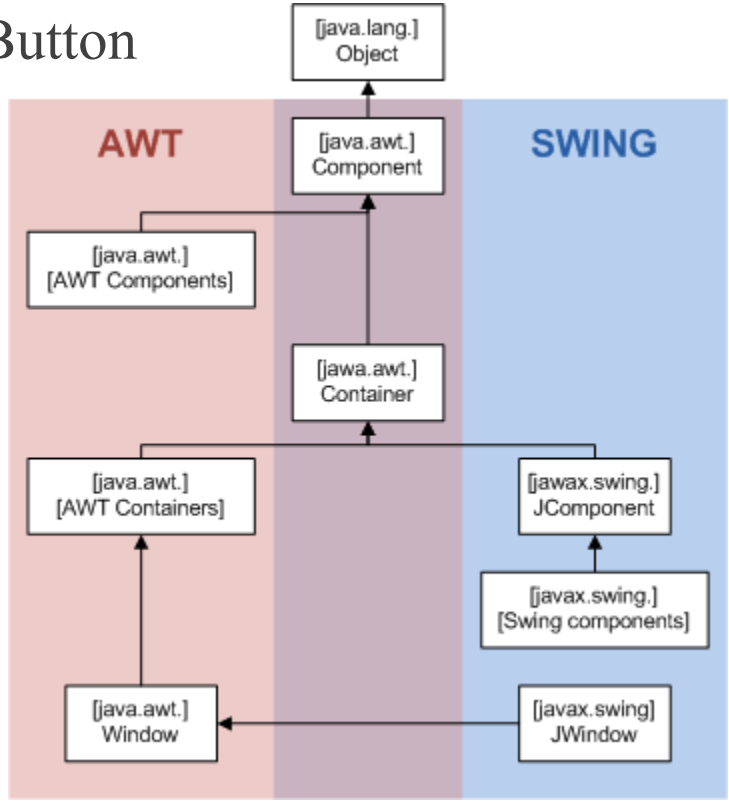
INTERFACE GRAPHIQUE (GUI)

Introduction

- Au début, les GUI étaient codés avec Abstract Window Toolkit (AWT) qui fournit une interface indépendante des OS, mais utilise leur composante native
  - Chaque composant AWT dois avoir un équivalent (peer component) sur chaque OS
  - Malheureusement, leurs comportements diffèrent parfois d'un OS à l'autre
- J2SE, JAVA 1.2, a introduit la library SWING qui permet de créer des IU identiques quel que soit l'OS
- SWING dessine lui-même les composantes et il est possible de choisir le « look and feel »

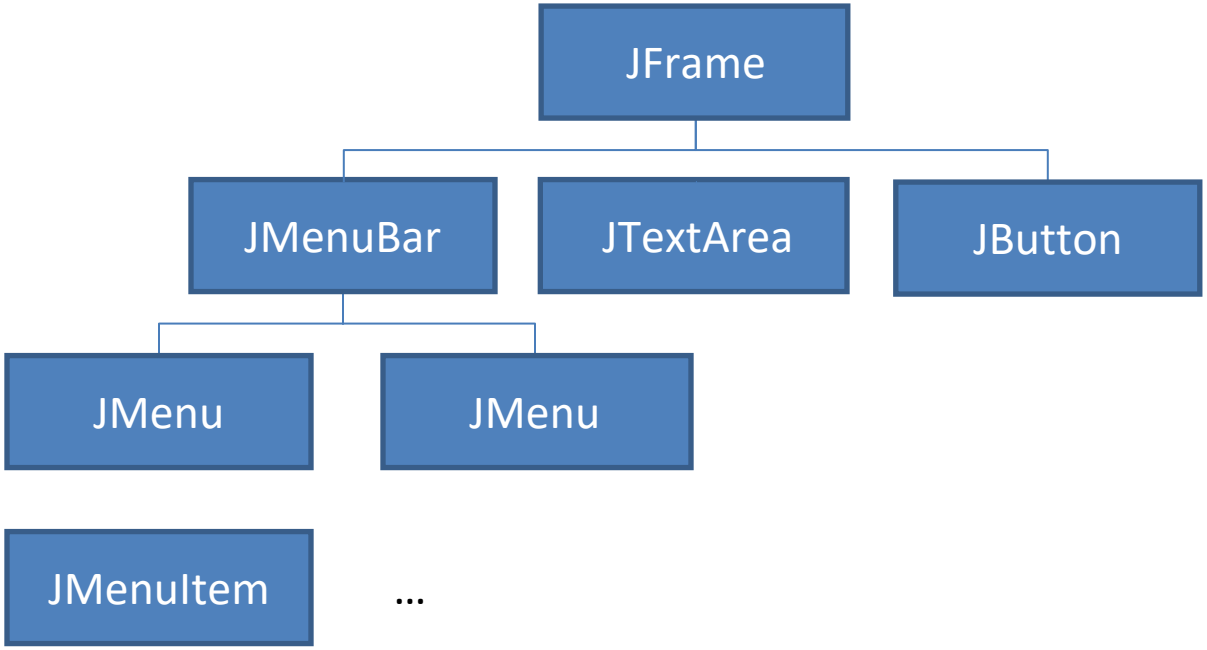
Introduction

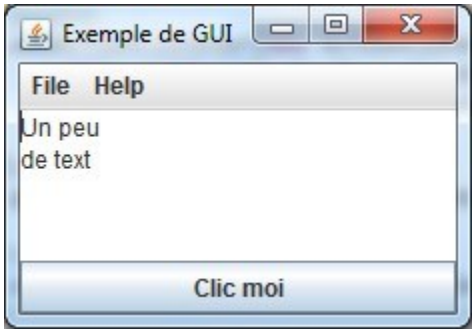
- SWING se trouve dans le package javax.swing
- Ses classes commencent par J
  - Frame
    java.awt.Frame, javax.swing.Jframe
    java.awt.Button, javax.swing.JButton
- Liens entre AWT et SWING



Wikipedia

Arborescence





- Suit une arborescence partant d'une composante système JFrame (ou Jwindow) qui contient tout les autres composants

Fonctions importantes

- setVisible() : rend le frame visible
- pack() ajuste la taille, sinon :
- setDefaultCloseOperation() :
  - DO_NOTHING_ON_CLOSE
  - HIDE_ON_CLOSE (default)
  - DISPOSE_ON_CLOSE (même chose que hide, mais libère les ressources)
  - EXIT_ON_CLOSE

- setResizable()
- add() et remove() des container pour ajouter des composants

Composants

- Hérite tout de JComponent
  - JPanel se comporte comme un composant normal, mais peut contenir d'autres composants
  - Il peut avoir un Layout Manager différent que son parent
- Plusieurs composants (label, bouton,etc) supportent du formatage HTML
- JLabel : texte non éditable
- JButton : bouton normal
  - Événements : avec addActionListener
    ActionListener contient actionPerformed()
- JToggleButton : bouton qui toggle
  - JRadioButton, JCheckBox
- ButtonGroup avec JToggleButton :
  un seul bouton peut être activer par groupe

Composants

- JTextComponent
  - JTextFeild, JPassWordFeild
  - JTextArea
  - JEditorPane, JTextPane
    (peuvent afficher du HTML)
- JMenu, JMenuBar , JMenuItem
- JTable

Événements

- Les événements d'IU traduisent des actions de l'utilisateur
- Les sources sont les composants avec lesquelles l'utilisateur a interagi
- ***MouseEvent*** : occasionnés par la souris
- ***KeyEvent*** : actions sur le clavier
- ***ActionEvent*** : clic sur un bouton, choix dans un menu, …
- ***WindowEvent*** : actions sur le cadre de Frame: ouverture, changement de taille, iconification…
- ***PaintEvent*** : événement notifiant à une fenêtre qu'elle doit redessiner son contenu
- ***FocusEvent*** : composant a « obtenu le focus »
- Etc..

Événements

- Les composants qui sont capables de générer des événements exposent des fonctions qui ont la forme
  - addXXXListener(XXXListener) Ex.: addMouseListener(MouseListener l)
- MouseEvent a les fonctions suivantes:
  - getButton()
  - getPoint() , getX(), getY()
  - getMouseModifiersText()

```
public interface MouseListener extends EventListener {
    void mouseClicked(MouseEvent e);
    void mouseEntered(MouseEvent e);
    void mouseExited(MouseEvent e);
    void mousePressed(MouseEvent e);
    void mouseReleased(MouseEvent e);
}
```

Exemple 1: GUI simple

```
class SimpleGUI1 extends JFrame{
    SimpleGUI1(){
      setSize(400,400);//la taille de la fenêtre en pixels
      //activer le bouton X pour fermer la fenêtre
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      this.setVisible(true);//activer la fenêtre
    }

    public static void main(String[] args){
       SimpleGUI1 gui = new SimpleGUI1();
    }
}
```

Exemple 2: GUI avec un bouton

```
public class SimpleGUI2 extends JFrame{
   SimpleGUI2(){
      setSize(400,400);//la taille de la fenêtre en pixels
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      //Créer un bouton
      JButton but1 = new JButton("Bouton");
      //Obtenir le contrôle du paneau pour ajouter le bouton
      Container cp = getContentPane();
      cp.add(but1);
      setVisible(true);
   }

   public static void main(String[] args){
      SimpleGUI2 gui = new SimpleGUI2();
   }
}
```

Exemple 3: GUI avec contrôle d'affichage

```
public class SimpleGUI3 extends JFrame{
    SimpleGUI3(){
       setSize(400,400); //la taille de la fenêtre en pixels
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       //Créer deux bouton
       JButton but1 = new JButton("Bouton1");
       JButton but2 = new JButton("Bouton2");
       //Obtenir le contrôle du paneau pour ajouter les boutons
       Container cp = getContentPane();
       //Définir le style d'affichage
       cp.setLayout(new FlowLayout(FlowLayout.CENTER));
       //Ajouter les boutons au paneau
       cp.add(but1);
       cp.add(but2);
       setVisible(true);
    }

    public static void main(String[] args){
       SimpleGUI3 gui = new SimpleGUI3();
    }
}
```

Exemple 4: GUI avec le contrôle d'événement

```
public class SimpleGUI4 extends JFrame{
    SimpleGUI4(){
       setSize(400,400); //la taille de la fenêtre en pixels
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       JButton but1 = new JButton("Bouton");
       //Obtenir le contrôle du paneau pour ajouter le bouton
      Container cp = getContentPane();
       cp.setLayout(new FlowLayout(FlowLayout.CENTER));
       //Ajouter le contrôle d'événement au boutton
       but1.addActionListener((ActionListener) new MyActionListener());
       //Ajouter le bouton au paneau
       cp.add(but1);
       setVisible(true);
    }
    public static void main(String[] args){
       SimpleGUI4 gui = new SimpleGUI4();
    }

    //Définir une classe interne pour saisir les événements
    class MyActionListener implements ActionListener{
        public void actionPerformed(ActionEvent ae){
           JOptionPane.showMessageDialog(null, "Cliqué!!!");
       }
    }
}
```