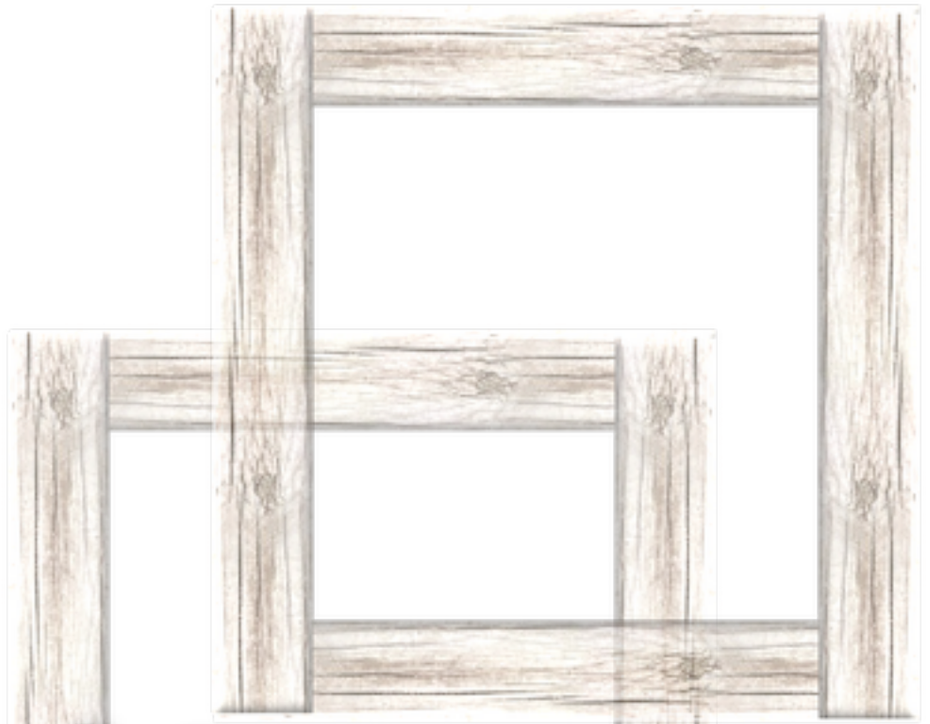


Interfaces Graphiques

Jean-Baptiste.Yunes@u-paris.fr
Université Paris Cité

- GUI Graphical User Interface, Interface Utilisateur Graphique, un type d'interface :
  - plus riche que le simple clavier et l'affichage en mode texte
  - un paradigme
    - une représentation picturale
    - un mode d'interaction gestuel

Un paradigme

- c’est un paradigme très répandu aujourd’hui
  - mais qui n’a pas que des avantages...
  - Neal Stephenson
    - In the beginning was the command line

http://fr.wikipedia.org/wiki/Neal_Stephenson

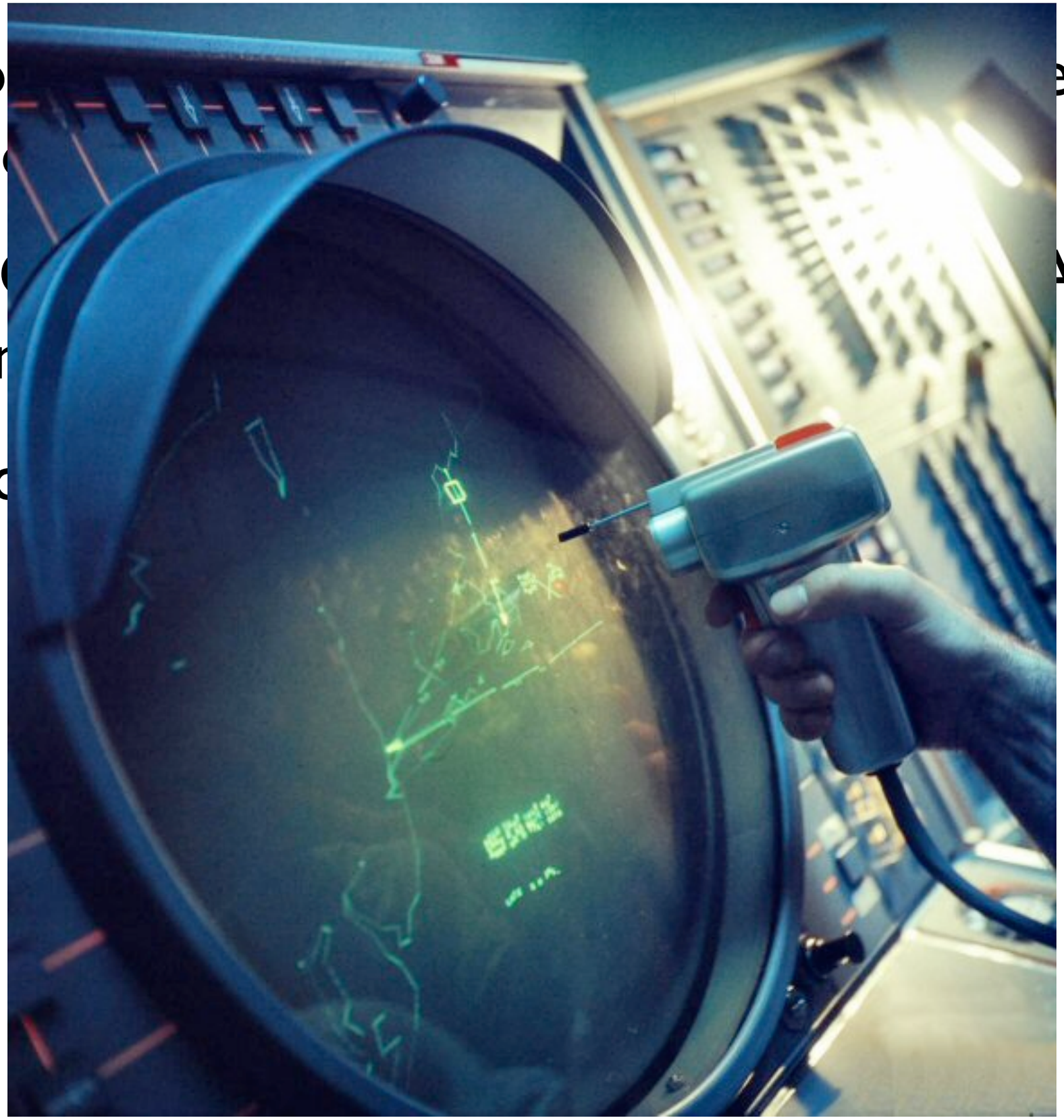
Un paradigme

- on considère habituellement que le père des GUI est Douglas Engelbart (1962)
  - AUGMENTING HUMAN INTELLECT: A Conceptual Framework
- mais que dire du projet SAGE 1950-1983 ?

http://www.williamson-labs.com/480_cpu.htm

Un paradigme

- on co…es GUI est D…
  - AU…A Cor…
- mais c…



http://www.williamson-labs.com/480_cpu.htm

- qu'est ce que ce type d'application a de particulier ?
  - son architecture
    - c'est l'utilisateur qui décide...
      - ...ou plutôt ce sont les évènements qui décident
- event-driven programming, event-driven design

Event-driven

- Event-driven programming ?
- une boucle principale :
  - (event detection) retirer un évènement
  - (event handling) distribuer l'évènement à la partie logicielle concernée

- La clé de tout lorsqu'on conçoit un programme utilisant une interface graphique c'est de suivre le conseil suivant :

  - une interface est bien conçue lorsque le programme se comporte comme l'utilisateur pense qu'il doit : user centric

  - ceci nécessite une conception soignée

- Quelle architecture ?
  - un système de fenêtrage (windowing system)
  - un gestionnaire de fenêtres (window manager)
  - des outils et bibliothèques d'objets (tools & API)

Architecture

- système de fenêtrage
  - X Window System (monde Unix) *aka* X11R7.7
  - Wayland
  - Quartz (Apple)
  - Windows (Microsoft)

- fournit aux applications un espace (graphique) d'accueil propre

Architecture

- gestionnaire de fenêtre (window manager)
  - kde
  - gnome
  - twm
  - motif
  - etc.
  - aqua
  - dwm (Desktop Window Manager)
- gère l'ensemble des fenêtres et uniformise leur manipulation

Architecture

- outils et bibliothèque d'objets (toolkits)
  - Xt, Xaw, Motif
  - GTK+
  - FLTK
  - Qt
  - wxWidgets
  - Cocoa
- uniformise la représentation visuelle des objets standards

Aqua/Quartz (MacOSX)

Photo d'écran, source privée

kde/X11 (Linux)

Photo d'écran KDE, source Wikipédia

gnome/X11 (Linux)

Photo d'écran Gnome, source Wikipédia

DWM (Windows 7)

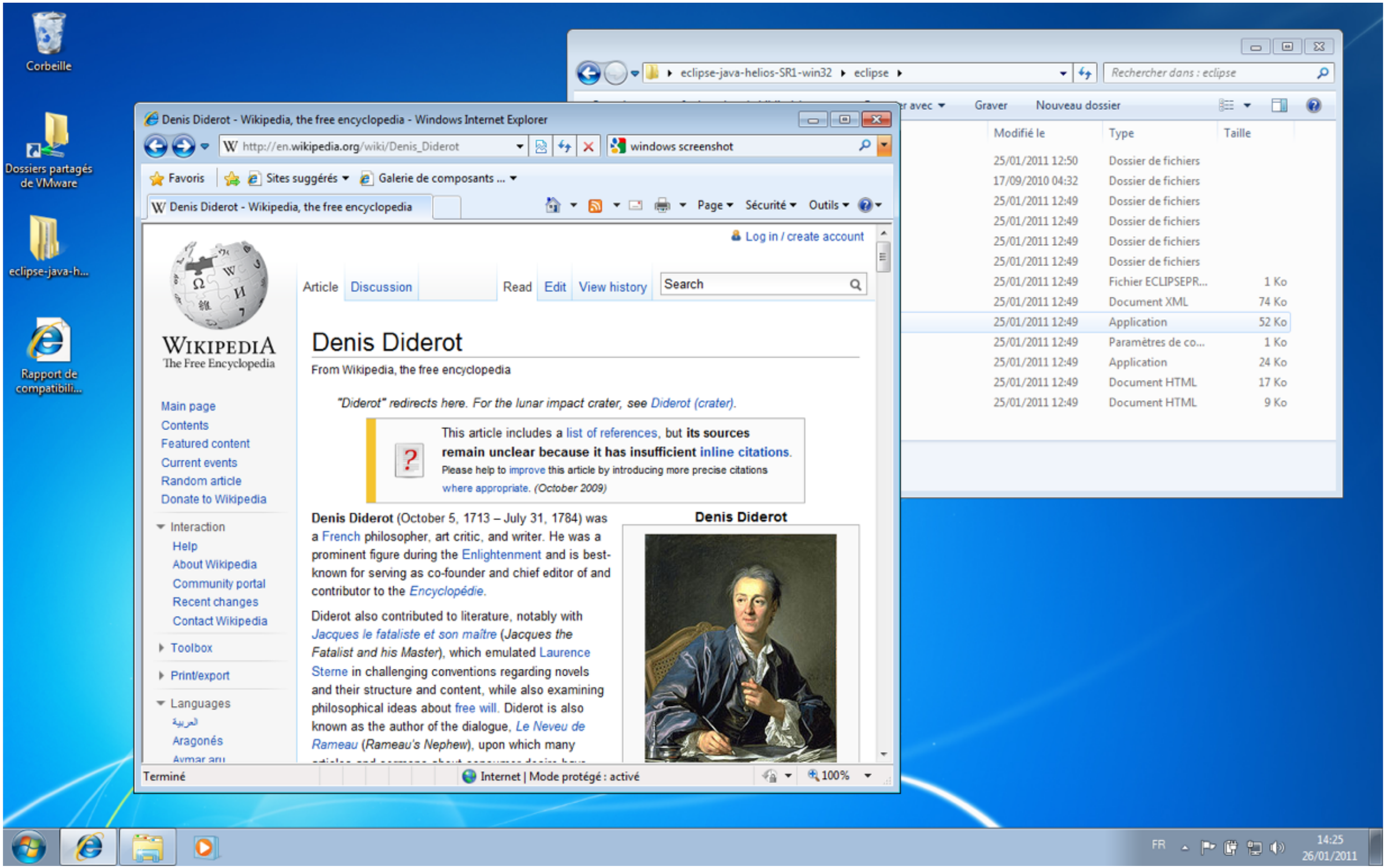


Photo d'écran Windows 7, source privée

Un exemple

- idée générale :
  - programmer sans de préoccuper des aspects graphiques

- toutefois il existe des règles à suivre pour créer une interface graphique dans un environnement donné
  - les guidelines

- KDE User Interface Guidelines
- Windows User Experience Interaction Guidelines
- Guide de l'Interface Utilisateur de Gnome
- Apple Human Interface Guidelines
- etc.

- suivre les règles permet d'obtenir une application qui se comporte comme attendu

- les API permettant en grande partie de garantir la conformité vis-à-vis de ces règles

- il n'y a donc qu'à se focaliser sur les aspects abstraits ou logiques

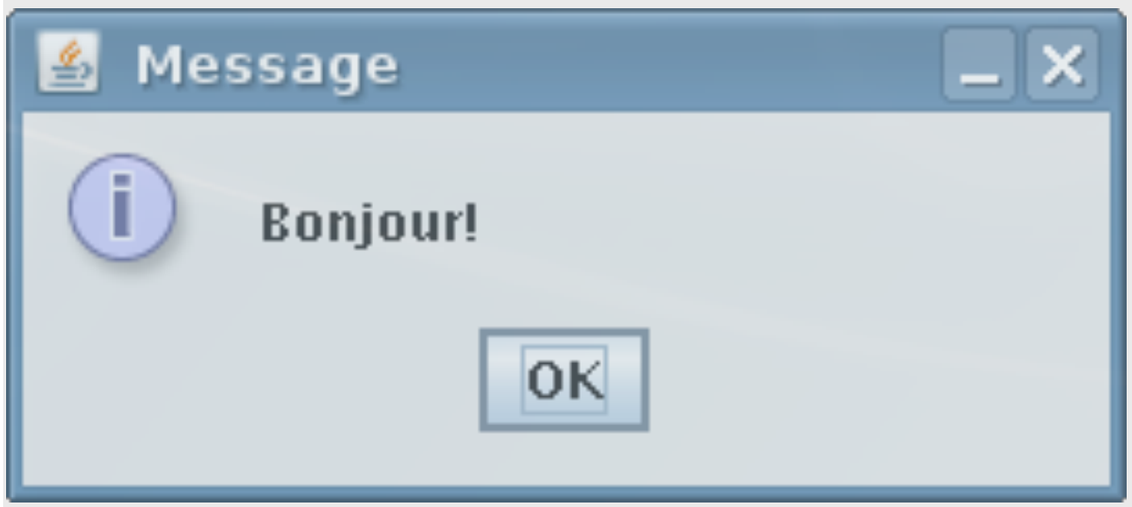
- un programme utilisant une interface graphique compliquée n'est pas nécessairement complexe

- les API offrent en général des objets tout faits permettant de rendre la plupart des services généraux attendus...

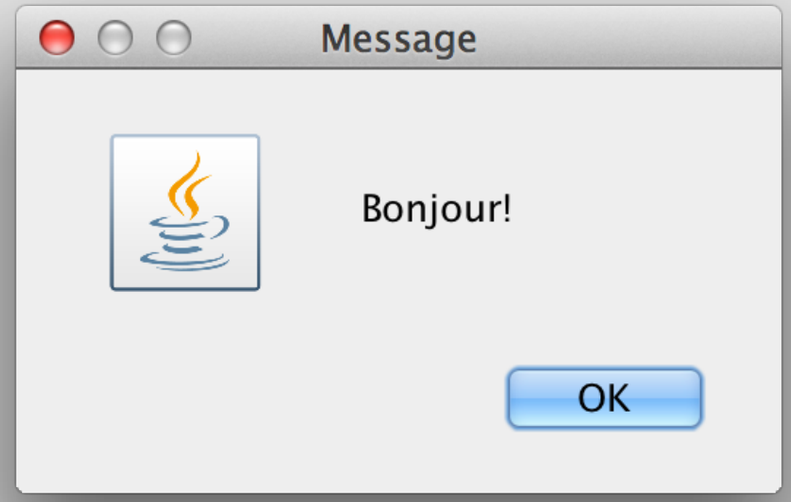
- afficher un message dans une fenêtre, par exemple...

Bonjour (minimal)

```
import javax.swing.JOptionPane;                                    Bonjour.java

public class Bonjour {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog( null, "Bonjour!" );
    }
}
```

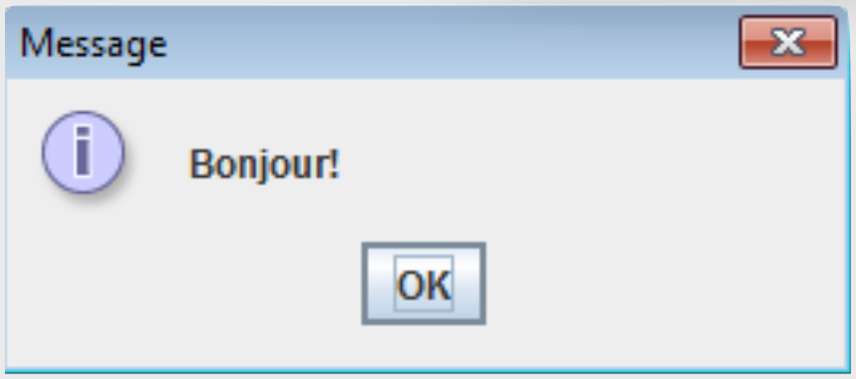


Linux (kde)

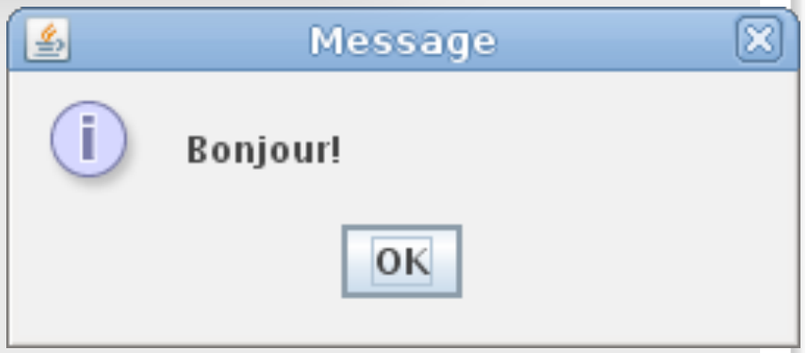


MacOSX

Linux (twm)



Windows 7

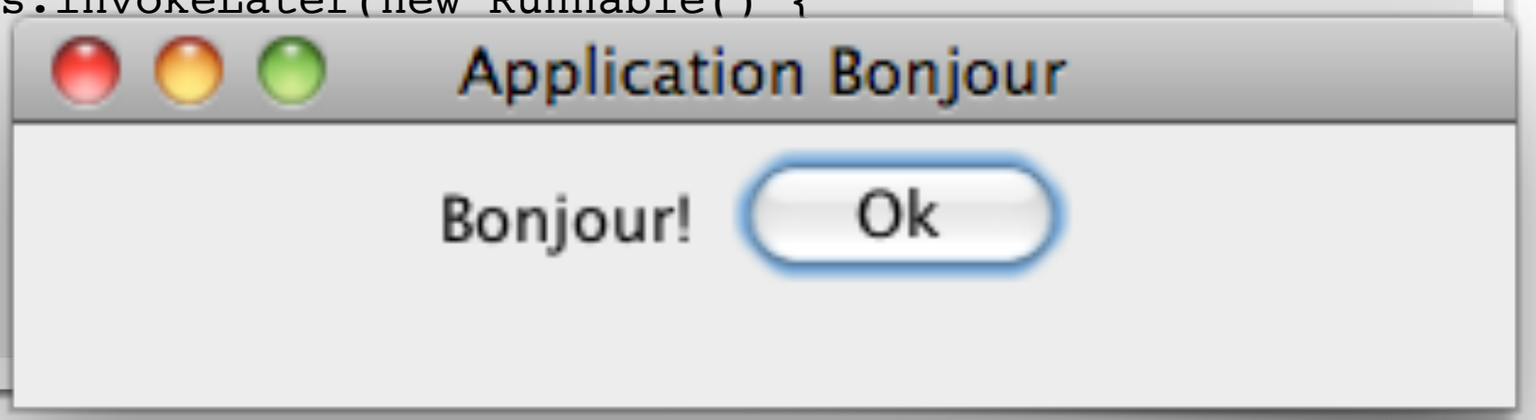


Linux (gnome)

Bonjour (à la main)

La même chose *à la main*

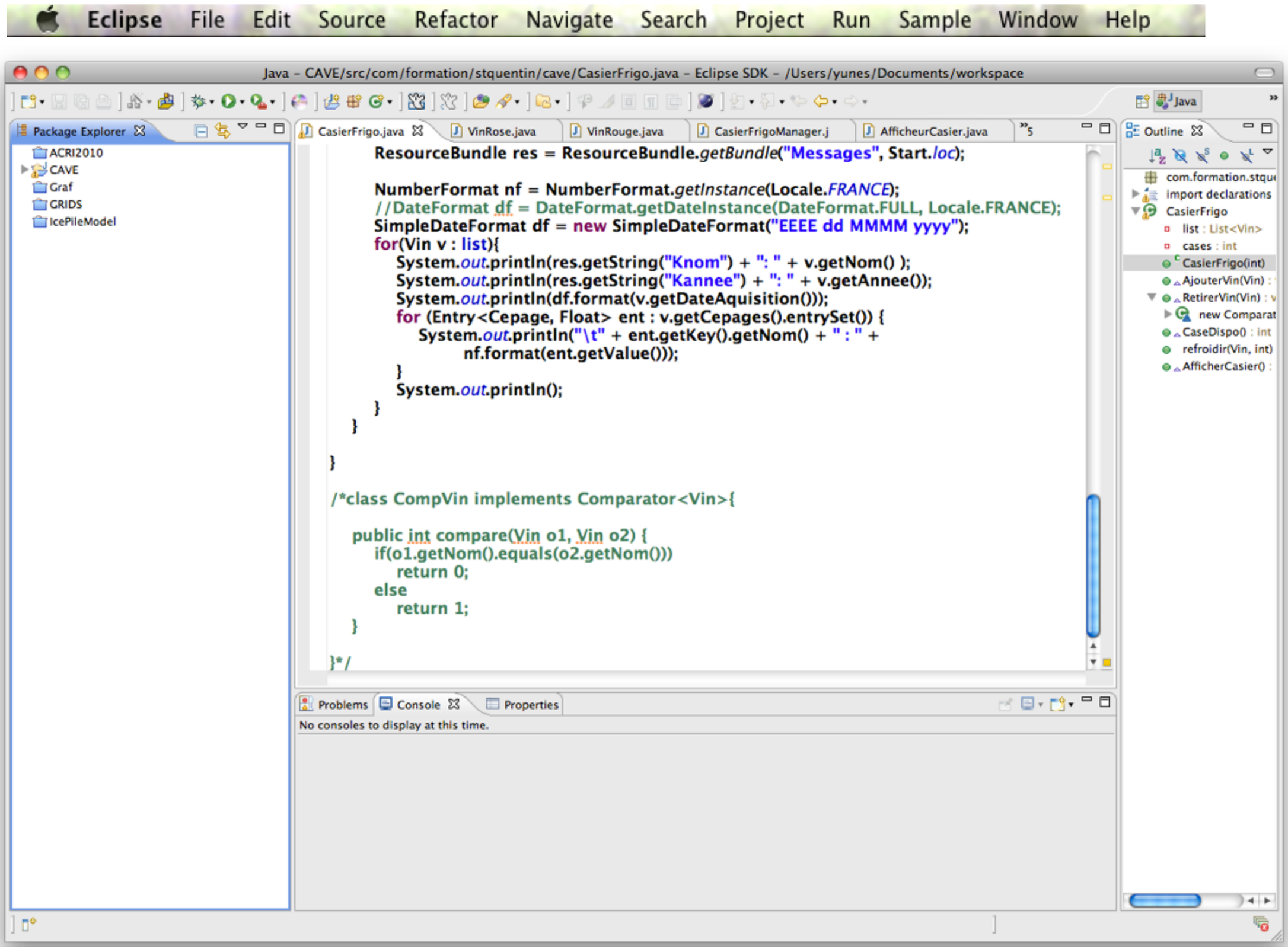
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Bonjour {
    private static void creeInterface() {
        JFrame frame = new JFrame("Application Bonjour");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Bonjour!");
        JButton button = new JButton("Ok");
        button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
          System.exit(0);
      }
        });
        frame.getContentPane().setLayout(new FlowLayout());
        frame.getContentPane().add(label);
        frame.getContentPane().add(button);
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                creeInterface();
            }
        });
    }
}
```



- Que sont/font tous ces objets ?
- Comment interfèrent-ils les uns avec les autres ?
  - C'est le sujet central de ce cours
  - Comprendre la structure d'une API d'une interface graphique
    - nous étudierons celle de Java : Swing
      - mais les autres sont assez semblables au moins dans leur principe

- En préliminaire on peut dire que les interfaces graphiques :
  - constituent un domaine pour lequel le paradigme objet est assez efficace
  - font une utilisation intensive de divers *design pattern*
- si vous avez déjà tout oublié : révisez…

Décomposition d'une interface



- GUI Java
- Historiquement :
- AWT Abstract Window Toolkit
  - son utilisation directe est considérée comme obsolète
  - sa disponibilité est conservée car :
- Swing, basé sur AWT...
  - il existe aussi SWT...

- AWT
  - les composants sont *lourds* (heavyweight), *i.e.* ils ont tous un pair natif attaché...
    - la conception d'AWT est thread-safe, trop lourd...
- Swing
  - sauf les composants racine, ils sont *légers* (lightweight)

- JFC Java Foundation Classes
  - composants Swing
  - look-and-feel
  - accessibilité
  - API 2D
  - internationalisation

- 12 packages AWT

```
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
```

- 18 packages Swing (JSE 7)

```
javax.swing                    javax.swing.plaf.nimbus
javax.swing.border             javax.swing.plaf.synth
javax.swing.colorchooser       javax.swing.table
javax.swing.event              javax.swing.text
javax.swing.filechooser        javax.swing.text.html
javax.swing.plaf               javax.swing.text.html.parser
javax.swing.plaf.basic         javax.swing.text.rtf
javax.swing.plaf.metal         javax.swing.tree
javax.swing.plaf.multi         javax.swing.undo
```

JFrame

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

public class Bonjour {
  private static void creeInterface() {
    JFrame frame = new JFrame("Application Bonjour");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel label = new JLabel("Bonjour!");
    JButton button = new JButton("Ok");
    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        System.exit(0);
      }
    });
    frame.getContentPane().setLayout(new FlowLayout());
    frame.getContentPane().add(label);
    frame.getConte
    frame.pack();
    frame.setVisibl
  }
  public static vo
    javax.swing.S
      public voi
        creeInte
      }
    });
  }
}
```

- JFrame ?
  - un des quatres conteneurs racine (top-level containers) `JFrame`, `JDialog`, `JWindow`, ~~`JApplet`~~
  - il faut au moins un conteneur pour afficher quelque chose...

JLabel

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

public class Bonjour {
  private static void creeInterface() {
    JFrame frame = new JFrame("Application Bonjour");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel label = new JLabel("Bonjour!");
    JButton button = new JButton("Ok");
    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        System.exit(0);
      }
     });
     frame.getContentPane().setLayout(new FlowLayout());
     frame.getContentPane().add(label);
     frame.getContentPane().add(button);
     frame.pack();
     frame.setVisible(true);
    }
    public static vo
       javax.swing.S
         public voi
           creeInter
         }
       });
    }
}
```

- JLabel ?
  - un composant permettant d'afficher un texte court, une image ou les deux
  - il est passif

JButton

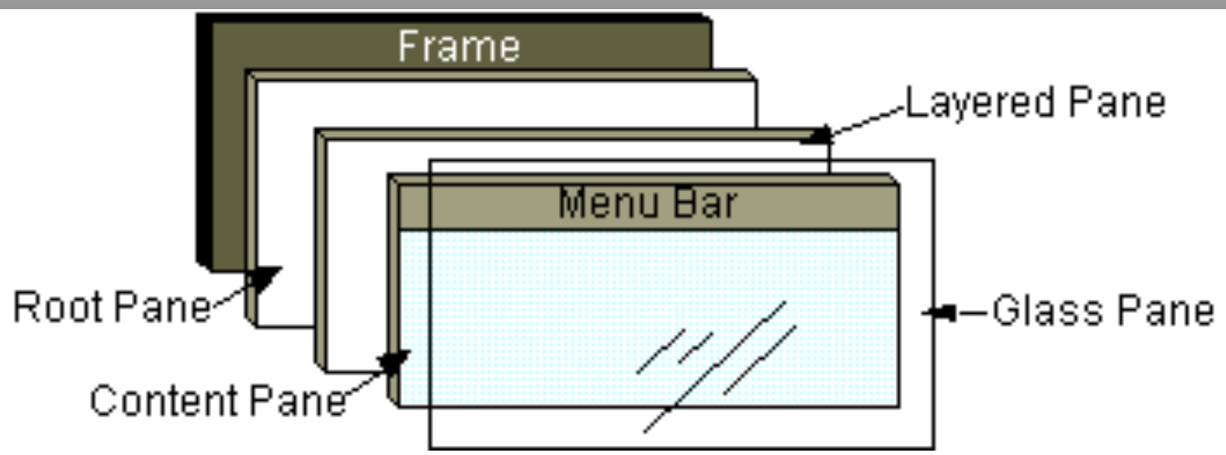
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Bonjour {
  private static void creeInterface() {
    JFrame frame = new JFrame("Application Bonjour");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel label = new JLabel("Bonjour!");
    JButton button = new JButton("Ok");
    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        System.exit(0);
      }
     });
     frame.getContentPane().setLayout(new FlowLayout());
     frame.getContentPane().add(label);
     frame.getContentPane().add(button);
     frame.pack();
     frame.setVisibl
    }
   public static vo
      javax.swing.S
        public voi
          creeInter
        }
      });
   }
}
```

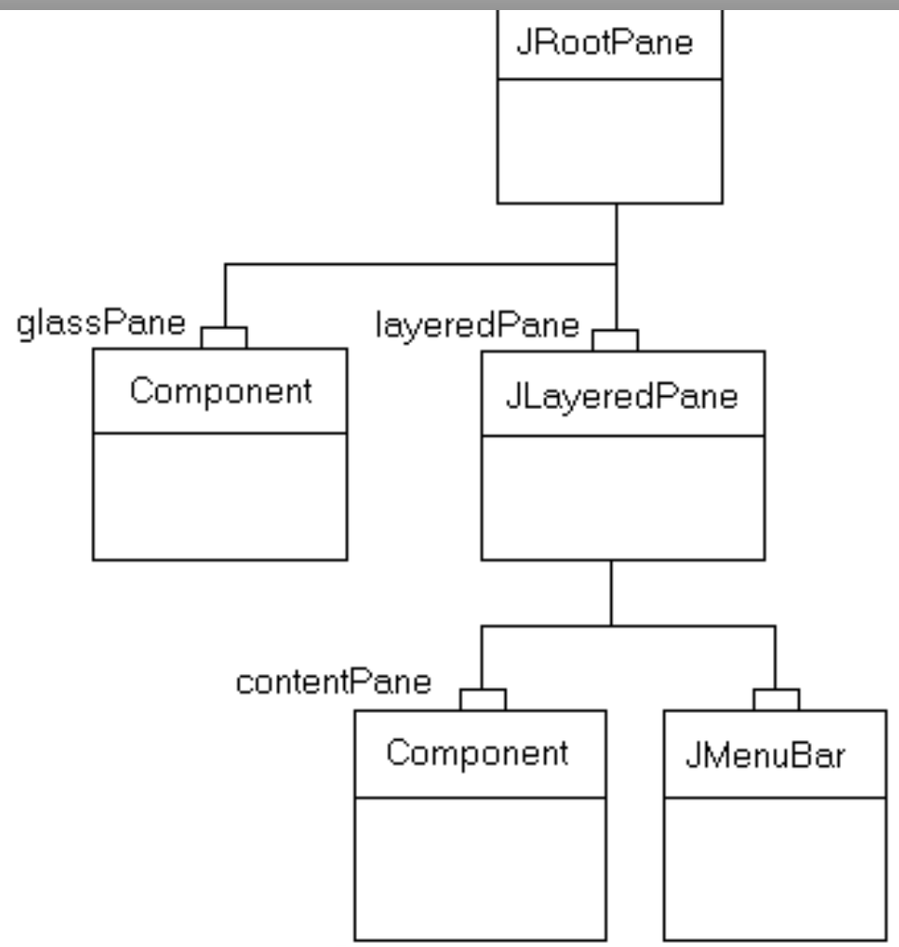
- JButton ?
  - un composant permettant d'obtenir une interaction basique avec l'application

Listener

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

public class Bonjour {
  private static void creeInterface() {
    JFrame frame = new JFrame("Application Bonjour");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel label = new JLabel("Bonjour!");
    JButton button = new JButton("Ok");
    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        System.exit(0);
      }
     });
     frame.getContentPane().setLayout(new FlowLayout());
     frame.getContentPane().add(label);
     frame.getConter
     frame.pack();
     frame.setVisibl
    }
    public static vo
       javax.swing.S
         public voic
           creeInter
         }
       });
    }
}
```

- JButton ?
  - `addActionListener` permet d'associer une *réaction* au clic, ici on termine l'application





- `JFrame` contient :
- un `JRootPane` qui lui-même est composé :
  - d'un `glassPane` (`Component`) qui recouvre un `JLayeredPane` lequel contient :
    - un `contentPane` (`Container`)
    - éventuellement une `JMenuBar`

- le `contentPane` par défaut est
  - descendant de `JComponent`
    - lequel utilise un `BorderLayout`
      - note : les composants Swing sont des Containers AWT...

Tapez pour saisir le texte

- `pack()` :
  - calcule les tailles adéquates permettant une représentation raisonnable à l'écran de la hiérarchie construite